Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de una plataforma para la simulación, análisis y gestión de mensajes de incidencias asociados a escenarios CBRN virtuales

Grado en Ingeniería Mecánica

ALUMNO: Álvaro Campos Rivera

DIRECTOR: José Antonio González Prieto

CURSO ACADÉMICO: 2020-2021

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de una plataforma para la simulación, análisis y gestión de mensajes de incidencias asociados a escenarios CBRN virtuales

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Infantería de Marina

Universidade Vigo

RESUMEN

La llegada de tecnologías cada vez más avanzadas y su elevado costo de adquisición y mantenimiento, han llevado a las Fuerzas Armadas a la búsqueda de nuevas soluciones relacionadas con la instrucción y adiestramiento de su personal, principalmente mediante el uso de herramientas de simulación e interfaces de usuario avanzadas. Entre estas necesidades se encuentran los sistemas de adiestramiento para situaciones de alerta CBRN, que son el objetivo principal de este trabajo.

De esta forma, se ha implementado el código necesario que, tomando datos de escenarios simulados en una herramienta software, automatice la redacción de mensajes de informe CBRN conforme al formato reflejado en la doctrina NATO STANDARD ATP-45. La redacción, lectura, transmisión y gestión de los mensajes se ha construido empleando el lenguaje Python, utilizando como referencia la estructura del formato previamente indicado. La arquitectura relativa a las comunicaciones del sistema se ha concebido mediante servidor-cliente, en donde se ha empleado el protocolo MQTT para la transmisión segura y rápida de mensajes. Esto permitirá la adición de nuevos elementos en los escenarios (sensores, fuentes, ...), o usuarios que participan en una simulación, empleando un estándar ampliamente utilizado. De esta forma se facilita que, en el futuro, sea posible cumplir el objetivo de crear simulaciones que integren las propiedades específicas en la transmisión de mensajes relacionadas de las redes de mando con las que cuentan las Unidades desplegadas.

De esta manera, se ha conseguido dar un primer paso hacia la creación de una línea de desarrollo de gran interés para la Armada, creando software a la medida de nuestras técnicas, tácticas y procedimientos, que sirva para la consecución del mayor grado de adiestramiento de las Unidades de la Flota.

PALABRAS CLAVE

CBRN, NBQ, MQTT, Python, Qt.

AGRADECIMIENTOS

A mi tutor José Antonio, quien siempre mostró un compromiso y una implicación por su trabajo digna de admiración. Gracias por guiarme no solo en el ámbito del proyecto sino en aquello en lo que ha sido necesario.
Por supuesto, gracias a mi familia y a mi novia, que han sido y serán mi aliento en los momentos más difíciles.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Armas de destrucción masiva

Cuando se escuchan expresiones del tipo: arma de destrucción masiva, es natural pensar directamente en bombas nucleares como las lanzadas en Nagasaki e Hiroshima. Sin embargo, este concepto es el utilizado para referirse a una familia de armas, las denominadas CBRN[1] (nuclear, biológica, química y radiológica). Tal y como se indica en [1], son aquellas pertenecientes a este grupo aquellas que ***"cuyo empleo por la mano del hombre pueden llegar a tener resultados desastrosos tanto para el hombre, la vida animal y/o las infraestructuras según se apliquen unas u otras y en función de su intensidad y potencia"***.

De esta forma, la utilización de este tipo de armas ha sido tradicionalmente asociada a eventos que pueden generar un efecto psicológico en la población sin necesidad de ser empleadas. Esto se debe a que una gran parte de estas armas quedan asociadas unos efectos destructivos intensos y duraderos en el tiempo, pero por otra parte generan la idea de estar expuestos a un arma invisible, así como una aparente incapacidad de dirigir con precisión los ataques a objetivos concretos, con la consecuente proliferación de los "daños colaterales".

Otro factor a tener en cuenta en este tipo de eventos CBRN, es que no solo tiene efectos inmediatos en el personal que se encuentra en la zona, sino que su capacidad de dispersión hace que pueda tener alcances de decenas de kilómetros [2], amentando ampliamente los costes, en todos los aspectos del evento, así como los asociados a la futura descontaminación del área.

1.2 Amenazas CBRN

1.2.1 Terrorismo

La historia del terrorismo en España está marcada en su mayoría por la actuación de la banda terrorista ETA. Sin embargo, dicho grupo armado estuvo enfocado al ámbito nacional y se construye sobre la concepción del apoyo popular vasco. Es por ello, por el gran riesgo de causar daños a sectores económicos y sociales afines a la causa, por lo que en un principio se tomó como remota la probabilidad de utilización de agentes CBRN por parte de la banda terrorista y se veía el IED[2] como el arma por excelencia para causar gran número de muertes en lugares públicos [3].

Por otro lado, es con la aparición del terrorismo islámico cuándo todas las alarmas de amenaza CBRN tomaron protagonismo, sobre todo después de los acontecimientos relacionados con el atentado

[1] También denominada NBQ-R, o simplemente NBQ
[2] Artefacto Explosivo Improvisado

del 11 de septiembre de 2001. Los atentados perpetrados por Al Qaeda contra Estados Unidos dejaron patente el recrudecimiento del contexto terrorista, en el que se recurren a complejos planes, que persiguen generar una gran letalidad y dónde el riesgo de la vida del terrorista no es un impedimento para llevar a cabo el atentado, llegando incluso a la inmolación. Además, las amenazas de capacidad nuclear por parte de su líder Osama ben Laden [4], convirtieron en urgente y necesaria la actualización de las Fuerzas y Cuerpos de Seguridad del Estado (FCSE) en materia CBRN.

El principal factor que pone en alerta a los Gobiernos es el interés demostrado por el terrorismo islámico en orientar sus generosos recursos hacia la proliferación de armamento CBRN, tal y como hizo Osama ben Laden en una tirada en 1998 [5].

Como consecuencia, ya no nos encontramos ante la amenaza clásica de dos naciones en conflicto, en la que los frentes de batalla están definidos y las actuaciones reguladas por tratados internacionales.

La amenaza terrorista cuenta con la capacidad de actuación en todo el territorio nacional [2], desde zonas con alta densidad poblacional donde el impacto en la población sería catastrófico, a lugares remotos en los que se la actuación y detección puede difícil, afectando a suelos de cultivos o incluso reservas de agua. Por todo ello, **la respuesta CBRN debe ser flexible, rápida y confiable, y debe ser coordinada** entre los diferentes organismos territoriales relacionados con la gestión de amenazas de este tipo.

1.2.2 Estados fallidos

Un Estado fallido se define por aquel que no es capaz de servir y garantizar unos servicios básicos, así como su propio funcionamiento. Las causas pueden ser múltiples, sin embargo, el mínimo común múltiplo es siempre el mismo:

- El Estado no posee una estructura fuerte.
- Se ha producido un vacío de poder en la cúpula del Estado.
- El Estado no posee una legitimidad suficiente o sus instituciones claves son débiles.
- No tiene las herramientas suficientes para garantizar los servicios básicos a sus ciudadanos. [6]

La organización de un ataque terrorista a gran escala precisa de tiempo y un lugar estable desde donde operar. El peligro que suponen estos Estados fallidos viene de la mano de la incapacidad de control que tienen sus FCSE, convirtiéndolos en un refugio idóneo para estructuras terroristas, desde donde orquestar un posible atentado CBRN.

Pero este no es la única amenaza que suponen los Estados fallidos, sino que un gran número de dictaduras han basado su supervivencia y seguridad usando tenencia y amenaza de uso de armamento CBRN. Además, la caída de dictadores inmediatamente después de abandonar sus programas de adquisición y almacenamiento de armas CBRN, supone un aliciente para estos regímenes totalitarios. [1]

1.2.3 Guerra convencional

En este caso, dos o más países que se encuentren en una guerra declarada podrían verse tentados a usar armas CBRN para atacar a su adversario. No obstante, existen ciertas convenciones, tratados y pactos entre países, que prohíben el uso de armas CBRN [7] [8] [9] [10]. En particular, en el caso de las armas nucleares, **el Tratado de No Proliferación Nuclear** solo establece prohibiciones parciales y, explícitamente deja exentos de la prohibición de tenencia a cinco países: Reino Unido de Gran Bretaña e Irlanda del Norte, Estados Unidos de América, República Francesa, Federación Rusa y República Popular de China [11]. Es con el **Tratado sobre la Prohibición de las Armas Nucleares** con el que se pretende llegar a un acuerdo para el desarme nuclear, lo que entra en oposición con la política de disuasión nuclear establecida por la **Organización del Tratado del Atlántico Norte (NATO**, de sus siglas en inglés), según la cual, la disponibilidad de capacidad nuclear se mantiene como un medio

disuasorio [12]. Cabe destacar que ninguno de los estados nuclearmente armados se ha adherido por ahora al Tratado de Prohibición de las Armas Nucleares, sin embargo, la NATO ha expresado como objetivo la creación de un mundo sin armas nucleares, aunque mientras éstas existan, la alianza contará con ellas.

1.2.4 Accidentes

Una particularidad de la defensa CBRN es que en nuestra sociedad existen sistemas productivos en los que se emplea material de esta naturaleza, que podrían generar eventos CBRN de cierta gravedad. Es por ello, que un accidente en uno de estos sistemas podría llegar a tener las mismas consecuencias que un arma utilizada de manera dolosa. A diario se usan químicos de forma industrial (TIM[3]), se produce energía a gran escala en centrales nucleares cuyos residuos se transportan a cementerios nucleares y se trabaja con agentes biológicos en investigación. Por ello, **la prevención** es un tema crucial para evitar accidentes, de forma que en todos estos casos existan protocolos de seguridad y actuación frente a este tipo de incidentes. Además, tras la creación de la **Unidad Militar de Emergencias** (UME) se ha estrechado la coordinación entre el **Ministerio de Defensa** y autoridades civiles, constituyendo esta la primera línea de actuación al contar con cometidos y material especializados para este tipo de intervenciones [13].

1.3 Objetivos

Los objetivos principales que se persiguen con el desarrollo del presente TFG pueden resumirse de la siguiente forma:

- Implementar de una clase que procese automáticamente mensajes de la serie CBRN según formato [14], es decir, que genere los mensajes enviados desde los sensores virtuales y además sirva para procesar su información desde el software del centro de mando que recibe los mensajes de las alertas CBRN
- Dar apoyo a la generación de un simulador de adiestramiento en redacción de mensajes CBRN.
- Realizar un análisis de los contenidos integrados en [14], sobre todo en lo referido a la serie de mensajes CBRN enfocado al adiestramiento de personal de equipos CBRN.
- Abrir una línea de desarrollo que ofrezca un campo de proliferación tecnológica en contribución a la construcción de la Armada 4.0.

[3] Materiales Tóxicos Industriales

2 ESTADO DEL ARTE

2.1 Agentes CBRN

Los eventos CBRN son una familia de cuatro tipos de incidentes, que aun estando englobados juntos tienen muchas características diferenciadoras. Conocer las particularidades de cada tipo de incidente aumenta considerablemente la eficacia en la respuesta y gestión de eventos. En la siguiente tabla se puedes ver algunas de las primeras características diferenciadoras de cada uno de los tipos de agentes CBRN.

Características	**Químico**	**Biológico**	**Radiológico**	**Nuclear**
Potencial destructor	Alto	Bajo a corto plazo Potencialmente muy alto	Bajo a corto plazo	Muy alto
Nivel de actuación	Táctico[4]	Estratégico[5]	Estratégico	Táctico/Estratégico
Tamaño de la zona afectada	Pequeña	Potencialmente muy alto	Pequeña	Grande
Capacidad de detección	Difícil sin despliegue de sensores	Difícil en fases iniciales	Imposible sin despliegue de sensores	Evidente

Tabla 2-1 Diferencias entre los tipos de agentes CBRN

2.1.1 Agentes químicos

Según la definición ofrecida por la **Organización para la Prohibición de las Armas Químicas** (OPCW de sus siglas en inglés) [15], la sustancias químicas (C) tóxicas son aquellas que actuando sobre personas o animales les provoca la "muerte, la incapacidad temporal o lesiones permanentes". Como podemos ver, y se dice de forma explícita en el texto, no se atenderán a los orígenes de esta substancia, pudiendo provenir de un arma química o de un proceso o vertido industrial.

[4] Sus resultados tienen efectos limitados en tiempo y área, normalmente enfocado a un combate u operación.
[5] Sus resultados buscan ser determinantes en un plazo mayor, normalmente enfocado a un conflicto completo.

La utilización de agentes químicos a gran escala de forma bélica llega en la Primera Guerra Mundial, en la que, llevados al campo de batalla, produjeron en torno a 90.000 muertes agónicas y casi un millón de personas heridas. Los agentes usados más comúnmente fueron el cloro, el fosgeno y el gas mostaza [16]. Las consecuencias de este trágico hecho son, por ésta y por otras razones, la existencia de la conocida como Zona Roja. La Zona Roja es un área de 800 hectáreas al Noreste de Francia, en Verdún, en la que debido a la cantidad de proyectiles sin explotar y la gran contaminación del suelo, provocada por el lanzamiento de gases tóxicos y la quema de munición química en los años posteriores a la guerra, está declarada como incompatible con la vida y por ello su acceso está prohibido. Se estima que se lanzaron 60 millones de proyectiles, de los que entre un cuarto y un octavo de ellos quedaron sin explotar. En cuanto al suelo, la cantidad registrada en algunos puntos ha llegado al 17.5% de arsénico (As), 35000 veces por encima de lo normal.

2.1.2 Agentes biológicos

Para definir lo que es un agente biológico recurriremos a la publicación NATO STANDARD ATP-45 [14], en la que se dice que son armas biológicas aquellos elementos que de alguna manera liberan agentes biológicos, incluidos vectores de propagación tales como los artrópodos. En la Figura 2-1 Vector de propagación de la malaria Anopheles).

Los ataques biológicos son uno de los más antiguos de este grupo de armas. Ya en el siglo XVI era común provocar plagas de ratas que transmitían múltiples enfermedades, destacando entre ellas la peste [18].

Figura 2-1 Vector de propagación de la malaria *Anopheles* [17]

2.1.3 Agentes radiológicos

Este tipo de agente, es cada vez más habitualmente tratado como un caso propiamente dicho, aunque es común verlo como una rama de los incidentes nucleares (como es el caso de la doctrina del Ejército de Tierra). Se trata de la dispersión de **isótopos radiactivos** en el ambiente, cuya inhalación o ingestión es dañina a largo plazo.

El poder destructivo de este ataque es relativamente bajo en comparación con el resto, sin embargo, tienen un gran impacto social, psicológico y político. Se ve como la principal amenaza radiológica la **bomba sucia** o **Dispositivo de Dispersión Radiológica (DDR).** Ésta consiste en adosar una bomba convencional a una masa de material radiactivo la cual, al ser detonada, libere el contaminante en la zona más extensa posible. En España, la principal debilidad respecto a este tipo de amenazas surge de la falta de experiencia en gestión de incidentes de este tipo. [19]

2.1.4 Incidentes nucleares

Las armas nucleares son las **más devastadoras** de todas las presentadas. La detonación de una bomba nuclear tiene efectos letales en radios de decenas de kilómetros, desde el punto de la explosión, pero además mantiene efectos de alta contaminación en zonas mucho más amplias y que permanecen

contaminadas durante mucho tiempo. Las unidades de medida de un arma nuclear son el ton, y equivale a una tonelada métrica (1000 kg) de TNT (trinitrotolueno). La mayor bomba nuclear detonada en la historia fue la conocida como **Bomba Zar** (Figura 2-2 Bomba Zar , de 50 Megatones, ésta fue lanzada en el Círculo Ártico por la URSS en plena guerra fría (1961). [21]

El escenario de amenaza de ataque nuclear se toma como probabilísticamente remoto, sin embargo, debido a las catastróficas consecuencias que conllevaría, lo convierte en un escenario de suma importancia. Las consecuencias de un ataque de este tipo son inmediatas y aterrorizantes, además la determinación de la autoría es algo que se encuentra todavía en estado de desarrollo, pues es especialmente difícil conseguir lo que se conoce como el ADN del mecanismo. [22]

Figura 2-2 Bomba Zar [20]

2.2 CBRN en las Fuerzas Armadas

2.2.1 Armada

Los **buques** de la Armada cuentan con **Planes de Defensa NBQR**, en los que se contempla el protocolo de medidas detalladas ante un evento CBRN. La condición de estanqueidad[6] durante la situación NBQR es estricta, impidiendo la propagación del agente contaminante a través del buque. La combinación de estas medidas, junto a un sistema de filtrado de aire y presurizado, dan lugar a lo que se conocen como **ciudadelas y santuarios**. Estas medidas constituyen el grueso de la Protección Colectiva (COLPRO).

Por otro lado, la **Fuerza de Infantería de Marina (FIM)** cuenta con una Sección de Defensa NBQ en el Grupo de Movilidad Anfibia (GRUMA) con pelotones de reconocimiento y descontaminación NBQ para apoyo a las operaciones de la Brigada, dotándola de capacidad de operar en ambientes CBRN.

2.2.2 Ejército de Tierra

Durante el reinado de Alfonso XIII se crea la primera unidad convencional especializada en CBRN, el Servicio de Guerra Química. Más tarde con la aparición del resto de dimensiones CBRN, la unidad irá adaptándose hasta el concepto actual. A día de hoy, el **Ejército de Tierra** estructura su defensa ante estos incidentes en **tres escalones**:

- A nivel individual, cada combatiente cuenta con un Equipo de Protección Individual (EPI-NBQ) y está instruido y adiestrado en su uso.
- A nivel Pequeña Unidad, existen módulos de defensa ante incidentes CBRN para unidades de entidad Batallón/Grupo.

[6] Disposición de las puertas aislantes que deben permanecer cerradas

- A nivel Gran Unidad, las Brigadas cuentan en su estructura orgánica con Compañías de Defensa NBQ como parte de los Batallones de Cuartel General. A nivel de Fuerza Terrestre cuentan con el Regimiento NBQ Nº1.

El **Regimiento NBQ "Valencia" Nº1** (Figura 2-3 Escudo del Regimiento NBQ "Valencia" Nº1) se remonta al 1 de marzo de 2005, pasando a asentarse desde entonces en Paterna (Valencia). Es la mayor unidad de las Fuerzas Armadas con cometidos específicos de Defensa CBRN y por ello se ha convertido en un referente a nivel nacional.

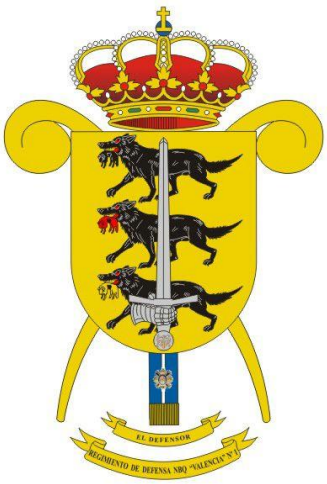

Figura 2-3 Escudo del Regimiento NBQ "Valencia" Nº1

2.2.3 Ejército del Aire

El **Ejército del Aire** está dotado con capacidad de defensa CBRN en **Bases Aéreas**. Esta capacidad es conseguida mediante el despliegue de sistemas de detección y control de la contaminación, además proporcionando medios para la protección individual y colectiva. Por otra parte, cuenta con capacidad desplegable de defensa NBQ, orientada a la seguridad de unidades desplegadas en el extranjero, disponiendo además de medios de identificación, señalización y de recuperación de afectados. De esta forma, el Ejército del Aire minimiza el impacto de eventos CBRN en el cumplimiento de sus misiones.

El **Escuadrón de Apoyo al Despliegue Aéreo (EADA)** asume los cometidos de permitir la defensa CBRN de agrupaciones aéreas expedicionarias mediante medios proyectables. Además, está dotada de una Sección de Defensa NBQ compuesta por Pelotones de Reconocimiento y Descontaminación NBQ, así como sistemas de protección colectiva desplegables.

2.2.4 Unidad Militar de Emergencias

La **Unidad Militar de Emergencias (UME)** se crea en 2005, constituyéndose como una fuerza **conjunta**[7] **y permanente** dentro de las Fuerzas Armadas. Su misión principal es la de intervención dentro de territorio nacional para contribuir a la seguridad y bienestar de los españoles, en coordinación con el resto de instituciones del Estado y administraciones públicas, en los supuestos de **grave riesgo, catástrofe, calamidad u otros.** Además, asume cometidos de defensa CBRN en las siguientes situaciones de emergencia:

- Las referidas a riesgos tecnológicos, entre ellos las situaciones CBRN.
- Aquellas derivadas de actos terroristas o ilícitos y violentos, incluyendo los que se perpetren contra infraestructuras críticas, instalaciones peligrosas o con agentes CBRN.

[7] Que cuenta con la participación de los dos Ejércitos y la Armada.

Es por ello que la UME cuenta con dos Compañías de Intervención en Emergencias Tecnológicas (CIET) en los batallones de Intervención en Emergencias I (BIEM I, Torrejón de Ardoz, Madrid) y IV (BIEM IV, Zaragoza).

2.3 Mensajes

2.3.1 Descripción

La actuación de cualquier sistema complejo, como es el caso de una intervención militar, requiere de la capacidad de transmitir información de forma robusta, fiable y rápida entre los componentes que formen parte del contingente de actuación, más aún si se tiene en cuenta que por norma general estos actores no se encuentran situados en el mismo lugar. Estos componentes, que se hallan desplegados en el terreno, deben ser capaces de comunicarse entre ellos, pero es incluso más crítico que sean capaces de comunicarse con el órgano de mando. La finalidad de esto es que el flujo de información sea bidireccional: los medios desplegados deben poder de transmitir datos que una vez procesados, para que al ser enviados a los puestos de mando sirvan para generar la toma de decisiones. De esta forma el órgano de mando debe disponer del enlace que le permita, no solo dar las órdenes oportunas a las unidades, sino también recibir la realimentación de la información relativa a la evolución del evento CBRN. La criticidad de este elemento de Mando y Control ha desembocado en la creación de estrictos protocolos de mensajería, en los que se trata de optimizar la cantidad de información transmitida frente a la legibilidad del mensaje por parte de un operador humano, y a su vez se intenta hacer prevalecer el texto frente al mensaje de voz debido a su mayor fiabilidad. Con estas medidas se pretende aprovechar al máximo el ancho de banda disponible en el enlace que estemos usando y generar protocolos robustos que eviten problemas de falta de trazabilidad, recepción de infamación ambigua y favorezcan su integración en sistemas de gestión automáticos.

2.3.2 Precedencia de un mensaje

Tal y como se ha tratado en el anterior apartado, el máximo aprovechamiento de los recursos es un objetivo permanente durante el empleo de las radiocomunicaciones. Por ello, para la contribución a la eficiencia en la explotación de los canales radio, se establece una prioridad en la transmisión de mensajes, llamada **precedencia**. La precedencia se define como la máxima demora tolerable para la recepción de un mensaje, lo cual irá finalmente asociado a una **prioridad** por parte del Sistema de Información y Comunicaciones (CIS). La correcta asignación de precedencias por parte de los remitentes de los mensajes asegura un correcto funcionamiento del sistema, del mismo modo que la primacía injustificada de mensajes anula el propósito del método.

La relación de precedencias asociadas a los mensajes CBRN, según está recogida en [23], es la siguiente:

- **FLASH (Z):** Tan rápido como sea posible, en un tiempo inferior a 10 minutos. Esta precedencia se reserva para informes de contacto inicial con el enemigo o de operaciones de combate de urgencia extrema.
- **INMEDIATO (O):** Tienen una demora de 30 minutos a 1 hora. Se usa para transmitir mensajes de alta urgencia que comprometan gravemente la seguridad de fuerzas o población, propias o aliadas.
- **PRIORIDAD (P):** Tiempo de recepción de 1 a 6 horas. Es la precedencia más alta para mensajes administrativos y se utiliza para conducción de operaciones en curso y otros temas de urgencia.
- **RUTINA (R):** Desde 3 horas a comienzos de trabajos del día siguiente. Se utilizará cuando no se requiera una precedencia mayor.

2.3.3 Seguridad de la información

2.3.3.1 Clasificación de seguridad

En el ámbito de las **Fuerzas Armadas**, la información que se trata tiene un carácter sensible, pues en ocasiones puede comprometer la seguridad de fuerzas propias o aliadas, así como población civil. Como consecuencia se establecen las clasificaciones de seguridad, que se definen como el nivel de **protección ofrecida a mensajes referidos a asuntos, actos, documentos, informaciones, datos y objetos** con el fin de evitar que personas no autorizadas accedan a esta información.

En la publicación [23] se reflejan los niveles de clasificación de seguridad:

- **SECRETO:** Precisa el nivel más alto de protección debido a su trascendental importancia o su capacidad de provocar riesgos o perjuicios de la seguridad y defensa del Estado.
- **RESERVADO**: Tienen una importancia menor que la de SECRETO, pero puede afectar a la seguridad y defensa del Estado.
- **CONFIDENCIAL:** Tienen una importancia menor que la de RESERVADO, pero puede dañar la seguridad del Ministerio de Defensa, afectar sus intereses u obstaculizar el cumplimiento de su misión.
- **DIFUSIÓN LIMITADA:** Tienen una importancia menor que la de CONFIDENCIAL, pero puede ir en contra de los intereses y misión del Ministerio de Defensa.
- **SINCLAS:** La información contenida en el mensaje no está comprendida en las clasificaciones de seguridad de los anteriores apartados y puede ser de conocimiento público dentro de las Fuerzas Armadas.

Por otro lado, al igual que sucede en el caso de la precedencia, la correcta gestión de la asignación de grados de clasificación asegura el buen funcionamiento del sistema. Una clasificación excesiva de la información provoca la pérdida del propósito de ésta. Otro factor a tener en cuenta en la asignación es el compromiso existente entre la seguridad y la accesibilidad a la información por parte de los destinatarios, es decir, mensajes con altas clasificaciones de seguridad serán difícilmente accesibles por aquellos a quien vaya dirigida la información, pues tendrán que pasar a través de todas las medidas de seguridad que la protegen.

2.3.4 Mensaje CBRN

El protocolo que regula la elaboración de los mensajes CBRN viene recogido en la publicación NATO STANDARD ATP-45. En ella se establecen los informes CBRN como una secuencia ordenada de mensajes que serán enviados al comandante del nivel que corresponda para permitir una correcta evaluación de la situación. Para la descripción del protocolo de mensajes distinguiremos entre **estructura y el formato del mensaje**.

2.3.4.1 Estructura

Como estructura se entiende la división de la información contenida en el mensaje en subelementos, de forma que fraccionemos los contenidos asociados a la información transmitida en bloques que sean fácilmente realizables.

Según recoge la ATP-45, la estructura del mensaje se dispone:

- Los mensajes están compuestos por módulos.
- Los módulos están compuestos por campos.

Las reglas básicas que cumplen estos bloques básicos de contenidos de información son:

- El primer módulo de cada mensaje está reservado para su identificación.
- El primer campo de cada módulo está reservado para su identificación.

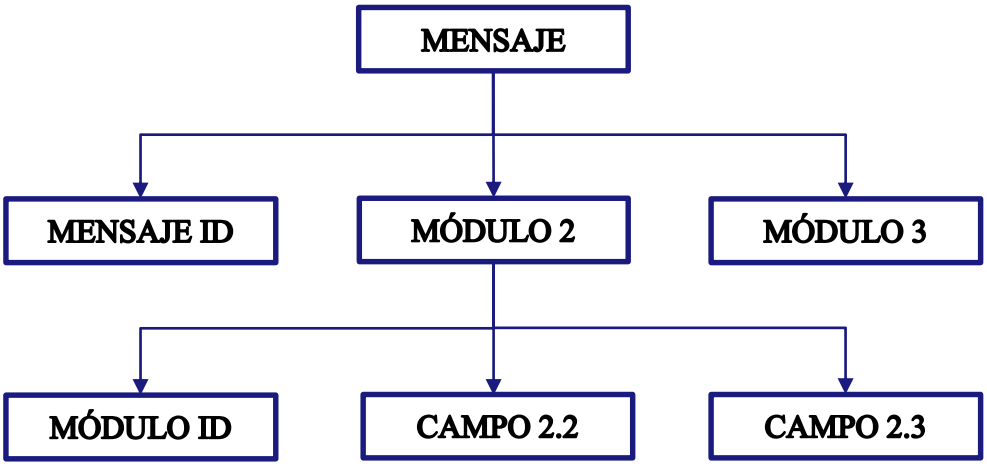

Figura 2-4 Estructura de mensajes CBRN

La estructura jerárquica de los contenidos de información incluidos en un mensaje CBRN puede visualizarse en la Figura 3-1, en donde el mensaje se compone de 3 módulos, y el módulo 2 se compone de 3 campos, siendo el primero el de identificación en ambos casos.

2.3.4.2 Formato

Se llama formato al conjunto de normas de redacción y sintaxis que permiten que la transmisión de información sea comprensible por el emisor y el receptor.

En lo referido a formato se contempla:

- El mensaje es una cadena de caracteres escritos siempre en mayúsculas.
- El símbolo separador de decimales será el punto (.).
- No se utilizará símbolo separador de miles.
- Los módulos y campos han de ir en el orden especificado en el protocolo.
- Al final de cada módulo[8] se escribirá una doble barra (//) y se cambiará de línea, ocupando cada módulo una línea diferente.
- Antes de cada campo se escribirá una barra (/), a excepción de aquellos que comiencen una nueva línea.
- Los campos incluidos en cada módulo están recogidos en la ATP-45.

  Ejemplo:

  MENSAJE ID
  MÓDULO ID1/CAMPO1.1/CAMPO1.2//
  MÓDULO ID2/CAMPO2.1/CAMPO2.2/CAMPO2.3//
  MÓDULO ID3/CAMPO3.1//

2.3.5 Condicionante de obligatoriedad

Los módulos y campos expresados en los mensajes tienen asignado un condicionante de obligatoriedad. Existen tres condicionantes que determinaran el nivel de obligatoriedad de incluirlo en el mensaje o no.

[8] Excepto el módulo de identificación.

- **Obligatorio (M):** El módulo es de obligatoria inclusión.
- **Operativo (O):** Se tendrán en cuenta consideraciones únicamente operativas para la inclusión de los módulos, siendo obligatorio si se dispone de dicha información.
- **Condicional (C):** Las condiciones de inclusión se establecen en cada uno de los módulos.

2.3.6 Secuencia de mensajes

La secuencia está constituida por una serie de seis mensajes CBRN, pudiendo cada uno ser retransmitido el número de veces que lo exija la situación. Esta secuencia de mensajes constituye la herramienta de comunicación **bidireccional** durante incidentes CBRN, por lo que habrá mensajes que sigan una ruta ascendente y otros descendente en la cadena de mando. La serie no será clasificada a menos que contenga información operativa específica, como puede ser el impacto del ataque sobre la situación de las tropas. Así mismo, cada uno de los mensajes de la serie tiene la finalidad de transmitir la siguiente información:

- **CBRN 1**. Identificación del observador o unidad que detecta el incidente e información general del entorno.
- **CBRN 2.** Análisis de la información aportada en CBRN 1.
- **CBRN 3.** Estimación inmediata de las áreas de contaminación y peligro.
- **CBRN 4.** Información de los datos de detección y los resultados de monitorización y seguimiento. Emitido por una unidad CBRN. Este mensaje se usa para dos fines:
  - Cuando no se ha observado ataque y el primer indicador de presencia de contaminantes es por detección.
  - Para dar medidas de contaminación como parte de una operación seguimiento o monitorización.
- **CBRN 5.** Información de las áreas de contaminación y peligro reales.
- **CBRN 6.** Información ampliada del estado del incidente incluyendo la capacidad para finalizar el incidente.

	CBRN 1	CBRN 2	CBRN 3	CBRN 4	CBRN 5	CBRN 6
Dirección	↑	↑	↓	↑	↑↓	↑
Prioridad	Flash	Inmediata	Inmediata	Inmediata	Inmediata	Inmediata
Origen	Cualquier unidad	Centro de Control	Centro de Control	Unidades especializadas	Centro de Control	Cada Escalón
Destino	Centro de Control	Ud. Superior	Unidades Subordinadas	Centro de Control	Unidades Superior y Subordinadas	Unidades Superior y Subordinadas

Tabla 2-2 Generalidades de los mensajes CBRN

Como se ve en la Tabla 2-2, la comunicación es bidireccional. Los mensajes que siguen el flujo ascendente en la cadena operativa son originados por unidades desplegadas en el campo, como pueden ser las redes de sensores previamente instaladas, y son datos que parten de la observación y recolección de muestras del entorno (CBRN 1 y CBRN 4), o del análisis e integración de la información aportada

por las subelementos del escalón que emite el mensaje (CBRN 2 Y CBRN 6). Por otro lado, los que siguen dirección descendente son aquellos redactados a partir del procesado, análisis e integración de múltiples mensajes ascendentes (CBRN 3 y CBRN 5), y contienen información importante para las unidades desplegadas.

2.3.1 CBRN CHEM

2.3.1.1 CBRN 1 CHEM

CBRN 1 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	C	Tabla 3-2
BRAVO	Localización del observador y dirección del incidente	M	Tabla 3-4
DELTA	Comienzo y finalización de incidente	M	Tabla 3-7
FOXTROT	Localización del incidente	O	Tabla 3-8
GOLF	Medio y cantidad de lanzamiento	M	Tabla 3-10
INDIA	Información de la dispersión en incidentes químicos	M	Tabla 3-16
MIKER	Descripción y estado del incidente	O	Tabla 3-22
TANGO	Descripción del terreno y vegetación	O	Tabla 3-26
YANKEE	Dirección y velocidad del viento	O	Tabla 3-29
ZULU	Condiciones meteorológicas	O	Tabla 3-30
GENTEXT	Texto libre	O	Tabla 3-35

Tabla 2-3 CBRN 1 CHEM

Este mensaje, tal y como se muestra en la Tabla 2-2 Generalidades de los mensajes CBRN, es un mensaje que redacta y envía cualquier unidad desplegada en el terreno que observa o detecta el incidente. Se asume que las unidades que se encuentran en zona de operaciones, cuentan con un nivel básico de adiestramiento en incidentes CBRN.

2.4 Protocolo MQTT

En el apartado de las comunicaciones, se necesita un sistema que proporcione conexión bidireccional y que soporte múltiples terminales transmitiendo incluso simultáneamente. Además, es interesante tener capacidad de establecer una comunicación jerarquizada en la que se decida que usuarios tienen acceso a que mensajes. Por todo lo reseñado, MQTT es un protocolo que cumple con todos los requisitos con el aliciente de su sencillez y escalabilidad.

MQTT (Message Queing Telemetry Transport) es un protocolo de la capa de transporte, implementado sobre TCP/IP, el cual se caracteriza por estar enfocado a conexiones "*Machine to Machine*" (M2M). La principal ventaja que ofrece este sistema es la sencillez y velocidad, pues mantiene una conexión TCP abierta, comprobando la validez del enlace mediante llamadas y asentimientos. El modo de funcionamiento es muy sencillo, existen dos tipos de roles para los equipos, los *publisher*[9] y los *subscriber*[10]. El primero de ellos será el emisor de información y el segundo el receptor, sin embargo, los equipos no se conectan unos a otros directamente, sino que existe un paso intermedio, el *broker*. El *broker* es el administrador de los mensajes y hará llegar a los suscriptores los mensajes que les correspondan a cada uno. Definidos los emisores, receptores y administrador, solo falta por saber qué mensaje le tiene que llegar a cada receptor, esto se lleva a cabo mediante la creación de *topics*[11]. Los topics son parte del *broker*, y cada *publisher* o *subscriber* se asocia con uno o más *topics*. La finalidad

[9] Editor
[10] Suscriptor
[11] Tema

es que cada suscriptor solo reciba mensajes de los temas en los que esté suscrito. A continuación, en la Figura 2-5, vemos un ejemplo de estructura MQTT.

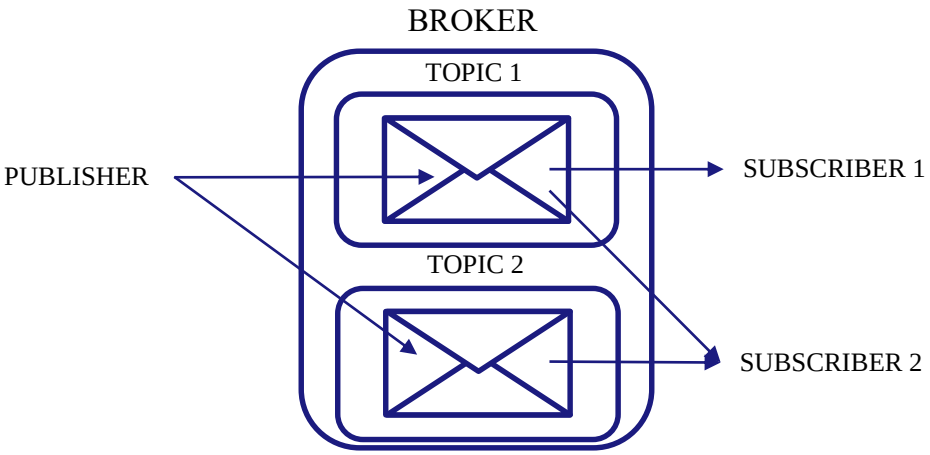

Figura 2-5 Ejemplo de organización protocolo MQTT

En la Figura 2-5, vemos una estructura en la cual el suscriptor 1 podrá tener acceso a la información que se publique en el *topic* 1, sin embargo, el suscriptor 2 accede a los *topics* 1 y 2.

Además, el protocolo MQTT está jerarquizado. Los temas pueden contener subtemas, dando lugar a la herencia de padres a hijos, es decir, un equipo que se suscriba a un tema, recibirá los mensajes publicados en éste y en el de los padres.

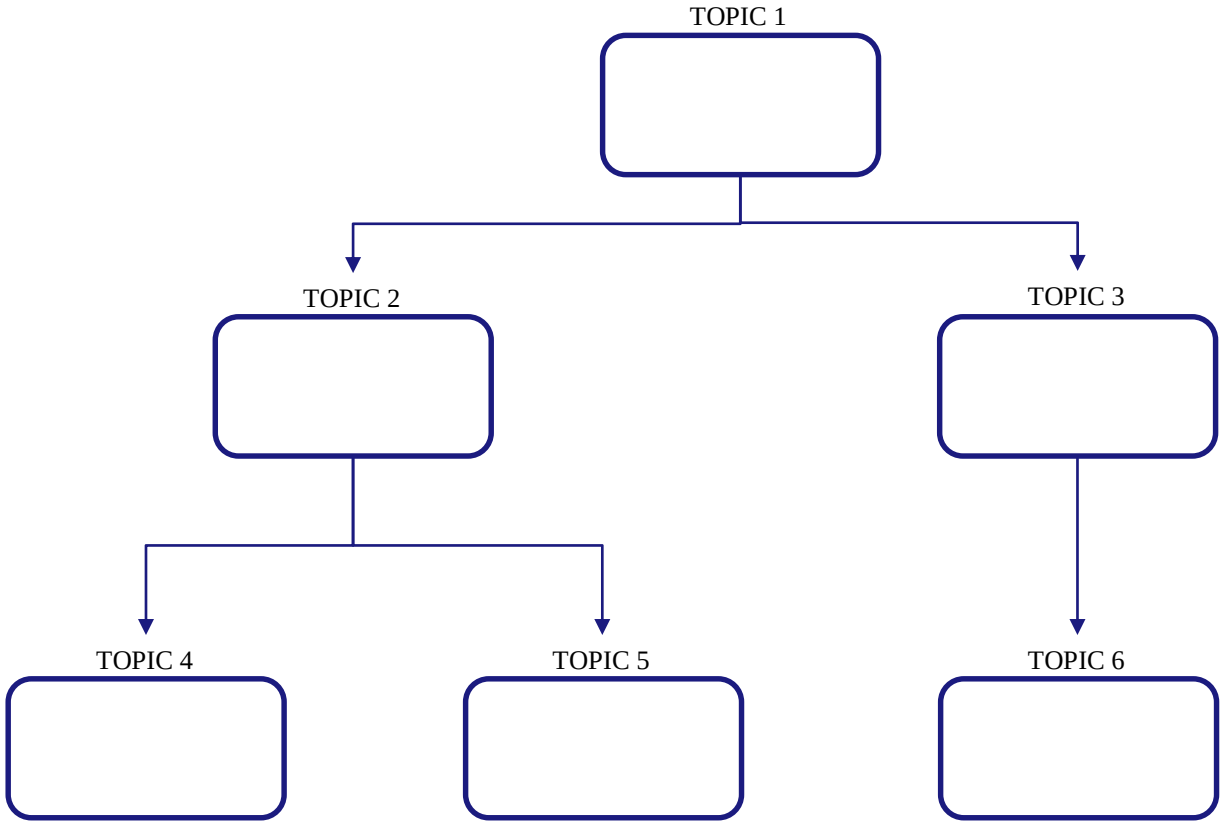

Figura 2-6 Ejemplo de estructura de *topics* en el protocolo MQTT

	TOPIC 1	TOPIC 2	TOPIC 3	TOPIC 4	TOPIC 5	TOPIC 6
TOPIC 1						
TOPIC 2						
TOPIC 3						
TOPIC 4						
TOPIC 5						
TOPIC 6						

Tabla 2-4 Tabla de explicación de las herencias del ejemplo en la Figura 2-6

En la Figura 2-6 se puede ver un ejemplo de estructura de temas en una conexión MQTT, y en la Tabla 2-4 Tabla de explicación de las herencias del ejemplo en la Figura 2-6 se puede ver las implicaciones de la herencia. En las filas se ven los temas a los que se puede suscribir y en las columnas los temas en los que se puede publicar, que como es evidente son los mismos. La intersección en la matriz coloreada en verde indica que: suscribiendo en el tema de la fila, recibiría el mensaje publicado en el tema de la columna. Por ejemplo, un suscriptor del TOPIC 4, recibirá mensajes publicados en: TOPIC 1, TOPIC 2 y TOPIC 4, permaneciendo el resto ocultos para él.

3 DESARROLLO DEL TFG

3.1 Introducción

Para la comprensión de los mensajes asociados al protocolo CBRN, se utilizará la máxima de la programación modular: *"Divide y vencerás"*, por lo que se atenderá con detalle a la estructura del mensaje CBRN, la cual se presentaba en el apartado 2.3.4.1. Se hará una revisión de los objetivos de comunicación de información que se persiguen en cada uno de los módulos, profundizando en cómo se subdivide cada uno de éstos, es decir, en el formato en que los campos codifican la información que se necesita transmitir. En [14] se puede ver que algunos campos pueden ser redactados de diferentes modos, cambiando los formatos del campo, las unidades o la información transmitida. En este trabajo se ha optado por la simplificación con el objetivo de homogenizar la descripción de sus contenidos, de forma que nos ceñiremos a aquellos mensajes de mayor interés y que nos permitan establecer las condiciones para el desarrollo de simulaciones básicas de escenarios CBRN. Por último, se dará una explicación de cómo se ha abordado la implementación del código que redacte y lea dichos campos.

3.2 Análisis de un mensaje CBRN 1

Para acometer la elaboración de mensajes CBRN en el entorno simulado, primero habrá que entender qué es lo que se pretende transmitir en cada uno de sus módulos. Los mensajes de esta serie se basan en la transmisión ordenada de información de manera que permita la recreación del escenario real y la predicción de la evolución del incidente. De esta forma, se dota al mando de las unidades con la capacidad de tomar decisiones, las cuales estarán encaminadas a llevar a cabo un control del evento con el mayor número de garantías posibles. Vale la pena recordar que, tal y como se veía en la Tabla 2-2 Generalidades de los mensajes CBRN, el tipo CBRN 1 es un mensaje emitido en sentido ascendente de cualquier unidad desplegada hacia el Centro de Control, y que éstas tienen un adiestramiento básico-medio en eventos de esta clase.

3.2.1 Encabezado identificador

En el identificador, se tiene que mostrar cual es el mensaje que vamos a transmitir, de esta forma el receptor del mensaje sabrá qué información espera recibir. Es por ello que en esta serie será siempre de la misma forma:

CBRN	**Nº de mensaje**	**Tipo de incidente**

CBRN	1..6[12]	CHEM/BIO/RAD/NUC

Tabla 3-1 Formato de identificación de mensajes CBRN

Como de muestra en la Tabla 3-1 Formato de identificación de mensajes CBRN, la primera cadena de caracteres serán las siglas 'CBRN', seguidas de un carácter numérico que irá del 1 al 6, el cual identifica al mensaje dentro de la serie, terminando con la identificación del tipo de incidente que se está informando.

3.2.2 Módulo ALFA

El módulo ALFA es el encargado de definir e identificar un mensaje, su procedencia y el tipo de incidente que reporta. Será el primer módulo corriente y, por lo tanto, en él se identificará el redactor del mensaje.

ALFA	/	CÓDIGO UNIDAD SUPERIOR (M)	/	CÓDIGO DE UNIDAD (M)	/	NÚMERO DE SECUENCIA (M)	/	TIPO DE INCIDENTE (M)	//
								Tabla 3-3	

Tabla 3-2 ALFA

En la Tabla 3-2 ALFA, vemos:

1) El primer campo, sin tener en cuenta el identificador, será el del Código de la Unidad Superior. Este se refiere a la máxima autoridad competente que está informando del evento. Por ello todas las Unidades rellenarán este campo con un guion, incluso el Centro de Control.

2) El segundo campo es el del Código de Unidad, y éste estará impuesto en la documentación referente a la operación que se está realizando. Puede ser un código alfanumérico y se trata de que, en la medida de lo posible, éste tenga algún tipo de relación con la unidad representada. Así, por ejemplo, si la unidad es la Segunda Compañía, su código será: 'CIA2'. En nuestro caso particular, para el CBRN 1, las unidades emisoras del mensaje serán los sensores desplegados en el escenario, por lo que cada uno de ellos se identificará en este campo.

3) El tercer campo es el número de secuencia, el cual identifica el incidente que estamos reportando. Recordemos la importancia de dicho campo, pues nada asegura que no vayan a suceder varios eventos CBRN simultáneamente o solapados en el tiempo. Por otro lado, hay que tener en cuenta el modelo de pirámide sobre el que se estructuran las unidades militares, pues un Centro de Control puede estar recibiendo mensajes de informes de diversas unidades que han presenciado cada una un incidente distinto, todas ellas identificando sus mensajes de la misma forma, comenzando por el número 1. Es por ello que no es extraño que una unidad tras emitir un mensaje con un número de secuencia, reciba una respuesta con otro número diferente. Esto puede deberse a que estamos recibiendo información sobre un incidente diferente al que se ha informado, o que nuestra unidad superior está asignando otro número de secuencia para este incidente. En cualquiera de los casos deberá especificarse en el campo de texto libre que veremos al final del mensaje.

4) El cuarto campo es el tipo de incidente, en él se identifica de nuevo la clase de evento que estamos reportando. Se seguirá la Tabla 3-3 Tipo de incidente.

[12] Notación UML. De 1 a 6.

Tipo de incidente	**Código**
Químico	C
Biológico	B
Radiológico	R
Nuclear	N
Desconocido	U

Tabla 3-3 Tipo de incidente

3.2.3 Módulo BRAVO

En este módulo empezaremos a definir el escenario, y por ello se dará información de la posición del observador y de la posición relativa de la fuente respecto de él.

BRAVO	/	LOCALIZACIÓN DEL OBSERVADOR (M)	/	DIRECCIÓN DESDE EL OBSERVADOR (M)	//
		Tabla 3-5		0-6400	

Tabla 3-4 BRAVO

Según vemos en la Tabla 3-4:

1) El primer campo será la localización del observador. Según se puede encontrar en [14], la posición geográfica puede darse en hasta siete formatos diferentes. La utilización de uno u otro dependerá en gran medida del estándar con el que se trabaje en la unidad. En el desarrollo de la herramienta se ha escogido el sistema de representación mediante latitud y longitud el cual se puede ver en la Figura 3-1 Explicación del formato de coordenadas. Este formato aporta sencillez y compatibilidad directa entre la interfaz y la aplicación. A la hora de trabajar con él, hay que tener en cuenta que en el mensaje no se transmitirán los signos de la coordenada, pues se referencia mediante el punto cardinal hacia el que se toma la coordenada. Puede verse en la Tabla 3-5.

LAT/LONG	**Coordenada (X)**	**Coordenada (Y)**	**Descripción**
Ejemplo	45.203500N	21.261500E	Longitud y latitud expresadas en grados con un máximo de 4 decimales. Definir dirección de la coordenada (N/S/E/W).

Tabla 3-5 Coordenadas en formato latitud y longitud

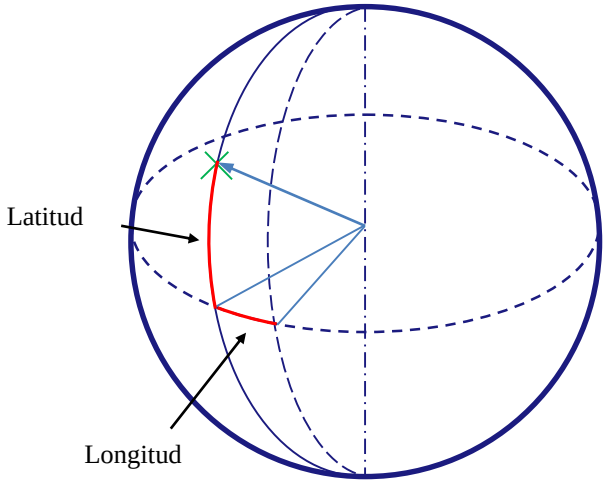

Figura 3-1 Explicación del formato de coordenadas

2) En el segundo campo del módulo se aportará la dirección relativa del blanco respecto del observador. Este dato se dará como un rumbo[13] medido en milésimas artilleras (ºº), una unidad de medida típica en las Fuerzas Armadas. Una milésima artillera se define como el ángulo bajo el que se ve 1 m a 1 km de distancia, por ello es frecuente su uso para estimaciones de distancias sin telémetro. La equivalencia que se toma es que una circunferencia completa son 6400ºº.

3.2.4 Módulo DELTA

Para continuar con la definición del escenario, tendremos que enmarcar el incidente en el tiempo. En el módulo DELTA se encuentran los momentos de inicio y finalización del incidente. El formato en el que se expresa un instante de tiempo es un sistema particular de las Fuerzas Armadas, el grupo fecha-hora. Este formato es una cadena de caracteres alfanuméricos expresado de tal forma:

DÍA	HORA	MINUTOS	HUSO HORARIO	MES	AÑO
2 caracteres	2 caracteres	2 caracteres	1 carácter	3 caracteres	4 caracteres

Tabla 3-6 Grupo fecha-hora

Ejemplo:

Día 16/02/2021 a las 17.05h en huso horario ‘Z’

161705ZFEB2021

Cabe destacar que el mes se redacta como las tres primeras letras del nombre del mes referido en mayúsculas. Por otro lado, las 00.00h puede dar lugar a confusión en lo que a día de refiere, es por ello que se prefiere la utilización de las 23.59h o de las 00.01h.

DELTA	/	GFH INICIO (M)	/	GFH FINAL (O)	//
		Tabla 3-6		Tabla 3-6	

Tabla 3-7 DELTA

Se puede ver como el primero de los campos, referido al inicio del incidente, será de obligatoria redacción, mientras que el segundo dependerá de la situación operacional, por lo que si llegase un mensaje con citado campo sin definir se entenderá que el incidente no se ha resuelto todavía. El inicio

[13] Ángulo bajo el que se ve un punto, medido desde el Norte Magnético con el sentido horario positivo

del incidente es algo relativo a la percepción del observador. Así, por ejemplo, un sensor que se encuentre en inmediaciones de la fuente tendrá una detección temprana, y un grupo fecha-hora anterior específico. Será responsabilidad del Centro de Control, determinar, tras analizar y procesar todos los mensajes recibidos, el dato que mejor se adapte para entender la situación real.

3.2.5 Módulo FOXTROT

En el módulo FOXTROT se transmitirá información sobre la posición de la fuente.

FOXTROT	/	LOCALIZACIÓN DE OBSERVADOR (M)		/	LOCALIZACIÓN (M)	//
		LAT/LON	Tabla 3-5		Tabla 3-9	

Tabla 3-8 FOXTROT

Por razones evidentes, a diferencia del módulo BRAVO, en el cual se enviaba la posición del observador, la localización de la fuente puede ser un dato que debe incluir ciertos niveles de incertidumbre (existe una amplia actividad investigadora destinada a la estimación de fuentes emisoras de contaminantes a partir de observadores y sensores sobre el terreno). Por eso mismo, la localización irá acompañada de un coeficiente de localización asociado a dicha incertidumbre, por lo que el valor de dicho coeficiente varía según la fiabilidad de la estimación. Factores como el conocimiento de la zona, el uso de telémetros o la cercanía a accidentes geográficos significativos, harán que la estimación sea más fiable.

Índice de localización	**Código**
Localización real	AA
Localización estimada	EE
Desconocido	NKN

Tabla 3-9 Índice de localización

3.2.6 Módulo GOLF

En este módulo se envía información referente al tipo de lanzamiento y a la cantidad de agente en dispersión.

GOLF	/	ÍNDICE OBRSERVADO-SOSPECHADO (M)	/	MEDIO DE LANZAMIENTO (M)	/	NÚMERO DE SISTEMAS DE LANZAMIENTO (M)	/
		Tabla 3-11		Tabla 3-12		0-999	

TIPO DE CONTENEDORES DE AGENTE (M)	/	TAMAÑO DE LA DISPERSIÓN (M)	/	ALTURA DE LANZAMIENTO (M)	//
Tabla 3-13		Tabla 3-14		Tabla 3-15	

Tabla 3-10 GOLF

Según vemos en la Tabla 3-10, este módulo se compone de seis campos:

1) El primer campo refleja cómo ha sido la alerta de este incidente. Esto se debe a que en entornos con amenaza probable de empleo de armas CBRN, los combatientes se adiestran en reconocer la presencia de contaminantes sin necesidad de observar la fuente. Como

ejemplo, un combatiente que se encuentre en una zona con estos riesgos, al detectar el olor de un contaminante como el gas mostaza que huele a ajo, dará la alerta de contaminante. Así se iniciará el protocolo sin necesidad de visualizar la fuente.

Índice observado-sospechado	**Código**
Sospechado	SUS
Observado	OBS

Tabla 3-11 Incidente observado-sospechado

2) En el segundo campo se da el medio de lanzamiento que ha provocado el incidente. De esta manera podremos comenzar a distinguir el origen del evento e incluso predecir otros datos del mismo. Por ejemplo, si el medio de lanzamiento ha sido un proyectil de mortero, el agente dispersado se limitará a aquellos que sean proyectables mediante este medio.

Medio de lanzamiento	**Código**
Aeronave	ACR
Bomba	BOM
Cañón	CAN
Artefacto	DEV
Instalación de fabricación de combustible	FFF
Almacén de material fisible	FMS
Instalación de reprocesado de combustible	FRF
Sistema lanzador múltiple de cohetes	MLR
Mortero	MOR
Misil	MSL
Carga útil de misil	MPL
Desconocido	NKN
Planta, fábrica	PLT
Vagón de ferrocarril	RLD
Reactor nuclear industrial	RNP
Reactor nuclear de investigación	RNR
Almacén de residuos nucleares	RWS
Barco, embarcación, buque	SHP
Instalación de material industrial radiológico tóxico	TIR
Transporte	TPT

Tabla 3-12 Tipo de medios lanzamiento

3) En este campo se reflejan el número de sistemas de lanzamiento empleados y se expresa como un número del 1 al 999.
4) El cuarto campo del módulo GOLF transmite información relativa al tipo de contenedor de la sustancia. Este campo puede llegar a ser de gran importancia a la hora de establecer las medidas de actuación y contención del agente. Por ejemplo, si es un gaseoducto el originador del problema, se buscarán las llaves en el circuito que eviten que siga dispersándose el contaminante.

Tipo de contenedor de agente	**Código**
Bomba de pequeño tamaño	BML

Bomba	BOM
Botella de gas presurizado	BTL
Bunker	BUK
Contenedor genérico de almacenamiento	CON
Tanque de 200 litros	DMR
Generador de aerosol	GEN
Contenedor intermedio para grandes cantidades	IBC
Grandes contenedores ISO	ISO
Mina CBRN	MNE
Carga útil de misil	MPL
Desconocido	NKN
Cabeza nuclear	NWH
Tubería, oleoducto, gaseoducto	PIP
Reactor	RCT
Cohete	RKT
Proyectil, armazón	SHL
Espray	SPR
Almacén, reserva	STK
Tanque de almacenamiento	TNK
Torpedo	TOR
Residuos	WST

Tabla 3-13 Tipo de contenedor de agente

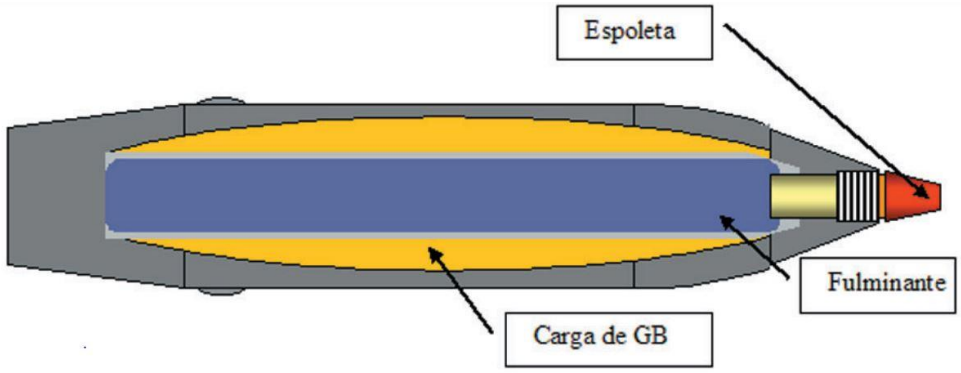

Figura 3-2 Carga sencilla. Proyectil de 155 mm con carga de agente GB [24]

5) En el quinto campo vendrá reflejada la cantidad de material que se disemina. Este dato suele ser difícil de obtener mediante una estimación del entorno. Sin embargo, puede ser que conozcamos, por ejemplo, las medidas del tanque que contiene el contaminante, por lo que, en casos similares a este, podremos dar un dato más cercano a la realidad. La cantidad de agente que se está liberando, por otro lado, es un dato útil a la hora de estimar la velocidad de propagación de la nube de contaminación y de sus efectos en las zonas alcanzadas. Para representarlo usaremos un código de representación por rangos.

Tamaño de la dispersión	**Código**	**Descripción**
Biológica industrial tóxica pequeña	SMLBIO	Menor que 1 kg
Biológica industrial tóxica mediana	MEDBIO	Entre 1 kg y 10 kg
Biológica industrial tóxica grande	LRGBIO	Entre 10 kg y 100 kg
Biológica industrial tóxica muy grande	XLGBIO	Mayor que 100 kg
Química industrial tóxica pequeña	SMLCHEM	Menor que 200 l/kg

Química industrial tóxica mediana	MEDCHEM	Entre 200 l/kg y 1500 l/kg
Química industrial tóxica grande	LRGCHEM	Entre 1500 l/kg y 50000 l/kg
Química industrial tóxica muy grande	XLGCHEM	Mayor que 50000 l/kg
Radiológica industrial tóxica pequeña	SMLRAD	Artefacto intacto o sin emitir
Radiológica industrial tóxica grande	LRGRAD	Fuente expuesta o en combustión
Radiológica industrial tóxica muy grande	XLGRAD	Artefacto en combustión o dañado y explosiones
Desconocido	NKN	

Tabla 3-14 Tamaño de la dispersión

6) Por último, el sexto campo corresponde a la altura de lanzamiento de la fuente. Sus implicaciones en el radio de dispersión son fáciles de ver si pensamos que cuanto mayor sea la altura a la que se está dispersando el agente, más distancia podrá recorrer antes de depositarse en el suelo.

Altura de lanzamiento de la fuente	**Código**
Aéreo	AIR
Desconocido	NKN
Subterráneo	SUBS
Superficie	SURF

Tabla 3-15 Altura de lanzamiento de la fuente

3.2.7 Módulo INDIA

En el módulo INDIA, se atiende al envío de información relativa al tipo de sustancia liberada, su persistencia y los procedimientos de detección usados.

INDIA	/	DEFINICIÓN DE LA SUSTANCIA (M)	/	TIPO DE PERSISTENCIA (O)	/	TIPO DE DETECCIÓN (O)	/	NIVEL DE CONFIANZA (O)	//
		Tabla 3-17		Tabla 3-19		Tabla 3-20		Tabla 3-21	
		Tabla 3-18							

Tabla 3-16 INDIA

Según se puede ver en la Tabla 3-16:

1) El primer módulo dará, si es conocido, el tipo de agente liberado. Esta información será utilizada por el Centro de Control para establecer las medidas de protección colectiva e individual que mejor se ajuste a la situación. Por ejemplo, si la liberación que está teniendo lugar es de un agente inocuo en la piel pero que causa daños al ser inhalado, los primeros esfuerzos se enfocarán en la protección respiratoria del personal. Se puede usar cualquiera de las dos codificaciones a continuación.

Tipo de sustancia	**Código**
Bacteriano	BAC
Biológico	BIO
Vesicante[14]	BL
Agente sanguíneo	BLOD

[14] Que provoca ampollas en la piel

Agente asfixiante	CHOK
Clamidia	CLA
Resultado de ataque nuclear	FL
Agente de la serie G	G
Mostaza	H
Incapacitante	INCP
Irritante	IRT
Agente nervioso	NERV
Desconocido	NKN
Cabeza nuclear	NWH
Otra sustancia	OTH
Agente penetrador	PENT
Bacteria Rickettsia	RIC
Material biológico tóxico industrial	TIB
Material químico tóxico industrial	TIC
Toxinas	TOX
Agente de la serie V	V
Vírico	VIR
Agente vomitivo	VMT

Tabla 3-17 Tipo de sustancia

Nombre de la sustancia	**Código**
Cianuro de hidrógeno	AC
Bencilato de quinuclidinilo	BZ
Fosgeno	CG
Cloruro de cianógeno	CK
Oxima de fosgeno	CX
Di-fosgeno	DP
Tabún	GA
Sarín	GB
Somán	GD
Ciclo-sarín	GF
Mostaza destilada	HD
Mostaza lewisita	HL
Mostaza nitrogenada	HN
Mostaza HT	HT
Lewisita	L
Cloropicrina	PS
Gas arsénico	SA
Gas lacrimógeno	TG
VX	VX
Bacillus Anthracis	ANTB
Brucella Spp	BRUB
Vibrio Chloreae	VICB
Escherichia	ESCB
Rickettsia Typhi	RITB
Yersinia Pestis	YPEB
Coxiella Burnetii	CBUB
Rickettsia	RICB

Salmonella Spp	SALB
Orientia Tsutsugamushi	OTSB
Shigella dysenteriae	SDYB
Salmonella Typhi	SATB

Tabla 3-18 Nombre de sustancia

2) En el segundo campo se encuentra el nivel de persistencia. Éste hace referencia a la volatilidad y solubilidad del agente en el ambiente. Los compuestos más químicamente estables tendrán niveles de persistencia mayores, y por lo tanto serán más difícilmente retirables del entorno.

Nivel de persistencia	**Código**
No persistente	NP
Persistente	P
Denso	T
Desconocido	NKN

Tabla 3-19 Nivel de persistencia

3) En el tercer lugar se redactará en tipo de detección, en el caso del escenario virtual se tendrá un sistema de detección remota. La importancia de esta información se debe a la cantidad de información que se puede esperar dependiendo de cómo se haya dado la detección.

Tipo de detección	**Código**
Laboratorio desplegado	DL
Sistema de detección puntual	PDS
Sistema de apoyo de detección activa	ASDS
Sistema de apoyo de detección pasiva	PSDS
Detección remota	RD
Detección satélite	SBD
Otro	OTH

Tabla 3-20 Tipo de detección

4) Por último, el nivel de confianza que se aplica a todo el módulo. El Centro de Control dará mayor peso a los datos reportados en la detección cuanto mayor sea este índice. Es importante por ello hacer un uso responsable del mismo, pues puede conducir a error durante el proceso de toma de decisión.

Nivel de confianza de la detección	**Código**
Definitiva	DEF
Evidente	EVI
Representativo	IND
Presumible	PRE

Tabla 3-21 Nivel de confianza de la detección

3.2.8 Módulo MIKER

En el módulo MIKER se presenta información relativa a la dispersión y la forma en que se ha liberado el contaminante.

MIKER	/	ESTADO DEL INCIDENTE (M)	/	DESCRIPCIÓN DEL INCIDENTE (O)	/	NIVEL DE LIBERACIÓN (O)	//
		Tabla 3-23		Tabla 3-24		Tabla 3-25	

Tabla 3-22 MIKER

En la Tabla 3-22 se puede ver:

1) El campo de estado de incidente aporta información sobre el estado de control de la dispersión y contaminación.

Estado del incidente	Código	Descripción
Activo	ACTIVE	Contaminación y peligro presente. Nivel de seguridad establecido según los procedimientos.
Continuado	CONTD	Contaminación y peligro reducida. Menor nivel de seguridad requerido que en los procedimientos.
Comprobado	CHECK	Permisivo para iniciar la retirada de EPI de acuerdo a los procedimientos
Limpio	CLEAR	Incidente finalizado

Tabla 3-23 Estado del incidente

2) En el segundo campo se ampliará información sobre el tipo de contaminación que se está produciendo de una manera cualitativa. Si no correspondiese ninguno de los casos reflejados a continuación se dejaría sin cubrir.

Descripción del incidente	Código
Dispersión radiológica activa	ARDD
Nube visible	CLOUD
Artefacto dispersor dañado	DPC
Evidencias de trastorno local	ESD
Explosiones e incendio	EXFIRE
Fuente expuesta	EXXS
Incendio activo	FIRE
Artefacto o dispositivo intacto	INT
Sustancia vertida al agua	INWAT
Flujo continuo en fuga de contenedor o tubería	LEAK
Líquido	LIQUID
Intercepción de misil	MSLINT
Dispersión radiológica no activa	NARDD
Gran cantidad de líquido estancado	POOL
Rotura catastrófica de tanque	RUP
Pequeña cantidad de derrame líquido	SPILL
Sólido	SOLID
Vapor detectado	VAP

Tabla 3-24 Descripción del incidente

3) Este módulo termina con el nivel de liberación del agente, en este campo se detalla cómo está siendo la continuidad en la diseminación del contaminante.

Nivel de liberación del agente	Código
Continuo	CONT
Liberación de una única nube	PUFF

Rociando	SPRAY

Tabla 3-25 Nivel de liberación del agente

3.2.9 Módulo TANGO

En el módulo TANGO, la información se enfoca en definir las características físicas del entorno, haciendo hincapié en los factores que pueden ser determinantes en la velocidad de propagación de la nube y la accesibilidad de nuestros medios.

TANGO	/	DESCRIPCIÓN TOPOGRÁFICA (M)	/	VEGETACIÓN (M)	//
		Tabla 3-27		Tabla 3-28	

Tabla 3-26 TANGO

Según se puede ver en la Tabla 3-26:

1) El primer campo se centra en los aspectos físicos del terreno.

Descripción topográfica	**Código**
Llano	FLAT
En pendiente	HILL
Desconocido	NKN
Portuario, marítimo	SEA
Urbano	URBAN
Valle	VALLEY

Tabla 3-27 Descripción topográfica

2) El segundo campo refleja el tipo de vegetación presente en la zona. Este campo debe ser muy tenido en cuenta pues, por ejemplo, un área de matorral bajo puede provocar que el contaminante quede depositado en él y afecte al personal que cruce por esa zona, o incluso pueden ofrecer protección los árboles altos de copas pobladas y densas.

Descripción de la vegetación	**Código**
Terreno sin vegetación	BARE
Desconocido	NKN
Otro	OTHER
Matorral bajo	SCRUB
Urbano	URBAN
Boscoso, arbolado	WOODS

Tabla 3-28 Descripción de la vegetación

3.2.10 Módulo YANKEE

En el módulo YANKEE, continuando con la definición del entorno, el siguiente factor sobre el que informar será el viento. Éste es otro factor clave, pues el viento será uno de los elementos más determinantes en la propagación del agente, tanto es así que existen formatos de mensaje específicos de informe de con condiciones de viento: CRBN CDR[15], CBRN EDR[16] y CBRN BWR[17]. Por ello, una

[15] Chemical Downwind Report
[16] Basic Wind Report
[17] Effective Downwind Report

información precisa sobre el viento conducirá a una mejor estimación de las zonas afectadas y predicción de la evolución del incidente.

YANKEE	/	DIRECCIÓN DE VIENTO PREDOMINANTE (M)	/	VELOCIDAD DE VIENTO PREDOMINANTE (M)	//
		(0-360)DTG		(0-999)KPH	

Tabla 3-29 YANKEE

1) En el primer campo se incluye la dirección del viento más representativo, sin atender a corrientes locales que se puedan crear debido a la orografía. Lo que se pretende es tener datos para poder realizar las estimaciones de las zonas afectadas con rapidez y contar con el tiempo que permita llevar a cabo las evacuaciones que se estimen y los reconocimientos necesarios en fases posteriores del incidente. El formato de representación será en grados sexagesimales, en el que la circunferencia completa son 360 grados. Después del número se escribirá 'DTG', para identificar que es una orientación medida desde el Norte Magnético, es decir, un rumbo.

2) En el segundo campo se da información de la velocidad asociada al viento más representativo. Este factor determinará la velocidad de propagación de la nube, y no deben incluirse los valores de las rachas de viento. Este factor podrá reflejarse en el campo de texto libre o campos específicos posteriores. El formato será con un número hasta 999 seguido de 'KPH', lo que determina que está expresado en kilómetros por hora.

3.2.11 Módulo ZULU

El módulo ZULU continúa definiendo la meteorología presente en la zona del incidente. Aquí se reflejarán datos que complementan a los de viento para un entendimiento completo de la situación atmosférica.

ZULU	/	INDICADOR DE ESTABILIDAD DE VIENTO (M)	/	TEMPERATURA EN SUPERFICIE (M)	/	HUMEDAD RELATIVA (M)	/
		Tabla 3-31		(-90/900)C		Tabla 3-32	

FENÓMENO METEOROLÓGICO PREDOMINANTE (M)	/	COBERTURA NUBOSA (M)	//
Tabla 3-33		Tabla 3-34	

Tabla 3-30 ZULU

1) El primer campo complementa directamente el módulo de viento, pues en aquél se incluía la dirección y viento predominante, sin importar su estabilidad. Podrá ampliarse la información en el campo de texto libre, como los sectores y rangos de variación de dirección y velocidad.

Indicador de estabilidad de viento	**Código**
Viento variable	VAB
Viento continuo	STL

Tabla 3-31 Indicador de estabilidad de viento

2) El segundo campo indica la temperatura del aire a nivel del suelo. El impacto de la temperatura en incidentes químicos está relacionado con la evaporación de contaminantes líquidos y su tiempo de presencia en la atmósfera. Un agente químico se evaporará antes a altas temperaturas, acelerando los efectos del ataque, sin embargo, también conllevará la eliminación prematura del agente. En el caso contrario, temperaturas más frías irán seguidas de una lenta diseminación del agente, que por otro lado puede hacer que el incidente se prolongue incluso semanas. Este campo redacta como un número de -90 a 900, seguido del carácter 'C', el cual indica que se trata de grados centígrados.

3) La humedad puede afectar a la efectividad del agente de diversos modos. En el caso de agentes provocadores de ampolla verán aumentada su eficacia, mientras que los agentes nerviosos no sufrirán un impacto directo. También puede afectar a la capacidad de agentes biológicos que no aguantes bien ambientes poco húmedos.

Rango de humedad relativa	**Código**
0%-9%	0
10%-19%	1
20%-29%	2
30%-39%	3
40%-49%	4
50%-59%	5
60%-69%	6
70%-79%	7
80%-89%	8
90%-100%	9

Tabla 3-32 Rango de humedad relativa

4) En cuanto al fenómeno meteorológico predominante cabe destacar que las precipitaciones pueden causar efectos muy parecidos a los de elevada humedad relativa. La diferencia puede ser que las precipitaciones pueden provocar la deposición del agente o incluso una menos tasa de evaporación en el caso de la nieve. Por otro lado, las capas de inversión provocarán condiciones climáticas muy estables, como consecuencia, la concentración del contaminante dentro de la capa de inversión será mayor que si no hubiese, mientras que fuera de la capa de inversión la densidad del agente será mucho menor.

Fenómeno meteorológico predominante	**Código**
Sin fenómeno climático significativo	0
Brisa marina	1
Brisa terrestre	2
Polvo, nieve, arena	3
Niebla	4
Llovizna	5
Lluvia	6
Nieve	7
Aguaceros	8
Tormenta	9
Parte superior de la capa de inversión térmica por debajo de 800 m	A
Parte superior de la capa de inversión térmica por debajo de 400 m	B
Parte superior de la capa de inversión térmica por debajo de 200 m	C

Tabla 3-33 Fenómeno meteorológico predominante

5) El sexto campo del módulo ZULU está reservado para la cobertura nubosa. La cantidad de nubes presentes en el cielo determinan en primer lugar la cantidad de luz que llega a la superficie y en segundo lugar la probabilidad de precipitaciones en el futuro. La cantidad de luz es un factor limitante en la utilización de agentes biológicos, pues muchos de ellos ven reducida su efectividad de forma drástica.

Cobertura nubosa	**Código**
Menos de medio cielo cubierto	0
Más de medio cielo cubierto	1
Completamente cubierto	2
Completamente despejado	3

Tabla 3-34 Cobertura nubosa

3.2.12 Módulo GENTEXT

La utilización de módulos es una herramienta útil para la transmisión estructurada de información ordenada y estandarizada, sin embargo, se encuentra limitada por la falta de cobertura sobre particularidades que se pueden dar al afrontar un problema tan complejo como un evento CBRN. El módulo GENTEXT se encarga de llegar a todas las grietas existentes en el resto de módulos, además el comandante usará este campo para la transmisión de evaluaciones e instrucciones especiales.

GENTEXT	/	INDICADOR DEL TEXTO (M)	/	TEXTO LIBRE (M)	//

Tabla 3-35 GENTEXT

1) En el primer campo se indicarán los temas que se expondrán a continuación, por ejemplo, ‘EVALUACIÓN DE LA SITUACIÓN’.
2) En el segundo campo se desarrollará los campos indicados en el anterior. La síntesis de la información ha de ser empleada para el máximo aprovechamiento de los sistemas de comunicaciones.

3.3 Implementación del código

La generación del mensaje ha sido implementada como una clase, llamada ‘CBRN’, que será la encargada de realizar el interfaz entre la aplicación de simulación de escenarios y el protocolo de envío de mensajes CBRN. Para ello la clase contiene la función ‘*send’* que es llamada desde el módulo ‘*Simulation’* cuando un sensor detecta presencia de un contaminante y necesita enviar un mensaje CBRN al Centro del Control. La clase, por lo tanto, cumple dos funciones diferentes:

1. Una función que obtiene los datos de los sensores empleados en aplicación de simulación y los transforma en el mensaje CBRN adecuado.
2. Otra función que realiza la operación inversa, es decir, recoge los datos del mensaje CBRN enviado al Centro de Control y los convierte en datos de simulación que permitan su representación visual y procesado para ser fácilmente entendible por el personal operativo.

3.3.1 Redacción del mensaje

Para explicar la estructura del código se empezará desde el nivel más general, el mensaje. El mensaje está compuesto de módulos, por lo que se ha hecho una función para cada uno. Cada **función de módulo** será llamada y su valor almacenado en una variable, tras lo cual se concatenan los valores ordenados en una cadena de texto que será el mensaje CBRN. A continuación, se explicará cómo se ha afrontado la implementación de cada una de las funciones de módulo.

3.3.2 Redacción del módulo

De forma análoga a la redacción del mensaje, en la función de módulo se hace mediante la llamada a una **función de campo**, que devuelve un valor el cual, tras ser almacenado en una variable, se encadenará al resto de valores de campo devueltos para formar el módulo. En los sucesivos apartados se explicará la implementación de cada uno de los campos.

3.3.3 Código del módulo ALFA

El módulo ALFA implementado en la función `getMessageAlfa`, se compone, por un lado, de los campos 1 y 2, que serán valores prestablecidos e indicados en las órdenes previas a la operación. Por ejemplo, el primer campo será siempre un guion debido a que el emisor no es la máxima autoridad y el segundo campo se establece como 'ESP' como identificador de la fuerza en una coalición multinacional. Por otro lado, se tienen campos que identifican a las unidades que emiten el mensaje y el tipo de incidente que se presenta, por ello deberá tener acceso a los datos del sensor que está emitiendo el mensaje.

3.3.3.1 Código del campo ALFA 3

El campo ALFA 3, implementado en la función `getMessageAlfaField3`, es el identificador de la unidad que emite, en nuestro caso el sensor. Este dato se encuentra almacenado en el diccionario `scenario_sensor_data`, por lo tanto, hay que pasarle esta lista a la función que incluye el valor asociado a `'sensor_id'`. El valor será un número entero que habrá que transformar en carácter usando `str()`.

3.3.3.2 Código del campo ALFA 4

El campo ALFA 4, implementado en la función `getMessageAlfaField4`, identifica el tipo de incidente que se está detectando. Como cada sensor sólo identifica un tipo de incidente, la implementación se hará accediendo a los datos del sensor, del mismo modo que se hacía en el campo ALFA 3. Esta vez, el valor que habrá que buscar será el asociado a la cadena `'type_id'`, lo que devolverá un número que identifica al tipo de incidente. Para ser compatible con el protocolo, este campo no ha de ser este número, sino el código del incidente (CHEM, BIO, RAD, NUC), Para hacer el cambio a este código se ha creado una variable de clase, una tupla que relaciona el código numérico usado en los sensores con los códigos de incidentes y su nombre en inglés:

```
self.sensor_types = {1:("Chemical","CHEM"), 2:("Biological","BIO"), 3:("Nuclear","NUC"), 4:("Radiological","RAD")}
```

Por ello, al ser el valor que se busca es el que se encuentra en segunda posición, se accederá a él de la siguiente forma:

```
self.sensor_types[scenario_sensor_data['type_id']][1]
```

Obteniendo el valor de cadena de caracteres que se pretende.

3.3.4 Código del módulo BRAVO

El módulo se implementa en la función `getMessageBravo` y está formado por datos que nacen del conocimiento de la posición del sensor, y de la posición relativa de la fuente respecto al sensor. La posición del sensor es conocida y se encuentra junto con el resto de datos del sensor, de forma análoga se podrá acceder a los datos de la fuente en el diccionario `scenario_source_data`.

3.3.4.1 Código del campo BRAVO 1

El primer campo, `getMessageBravoField1`, que será la posición del sensor se ha de tener en cuenta que una localización queda definida por dos coordenadas (x, y), sin embargo, en el estándar de formato latitud-longitud, la forma de dar las posiciones es (y, x), pues la latitud siempre precede a la longitud. Además, en el formato del mensaje no se permiten valores negativos, a diferencia de los que puede pasar el sensor. Por ello, será imperativo hacer una evaluación del signo de cada coordenada, para

que después de pasar realizar el valor absoluto se asigne el punto cardinal hacia el que se toma. Por ejemplo, una latitud que tenga un valor de -4.73916º, según el estándar del mensaje será 4.73916S. Tras haber tratado cada una de las coordenadas se concatenarán una seguida de la otro, latitud seguida de longitud, y se devolverá este valor.

3.3.4.2 Código del campo BRAVO 2

Este campo, `getMessageBravoField2`, representa el ángulo bajo el que el observador ve la fuente. Para ello, se le pasarán las posiciones de sensor y fuente. Esta última en un incidente real sería un dato oculto, el cual habría que estimar mediante algoritmos de estimación (como por ejemplo los filtros de Kalman). Sin embargo, se ha omitido esta parte por el carácter docente del simulador, no descartándose su futura implementación.

Para acometer la implementación de este campo hay que tener en cuenta que los ángulos serán medidos desde el Norte Magnético, que el sentido positivo será el horario y que la unidad de medida será la milésima. Se ha utilizado la función `atan2` la cual devuelve el ángulo en radianes cuando se le da una dupla de números. En este caso habrá que obtener la diferencia entre ambas coordenadas, utilizando como centro el sensor y pasar los datos traspuestos, es decir, primero la componente horizontal, para obtener el ángulo con el eje vertical. Por último, habrá que hacer una evaluación de signo, pues en el III y IV cuadrante se obtendría valores negativos, por lo que habrá que sumarle una vuelta en radianes, que son las unidades que devuelve esta función. Por último, se hace un factor de conversión a milésimas y se devuelve el valor en forma de caracteres.

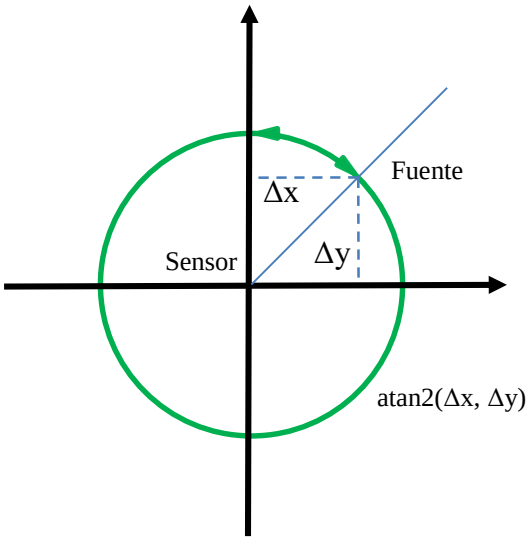

Figura 3-3 `atan2`

3.3.5 Código del módulo DELTA

Este módulo se desarrolla en la función `getMessageDelta`, en éste se dará información temporal del inicio y final del incidente.

3.3.5.1 Código del campo DELTA 1

Para hacer la función `getMessageDeltaField1`, se ha utilizado la librería `datetime`, en concreto la función `datetime.now.strftime`. Esta librería toma datos de fecha y hora del equipo, por lo que los datos expresados en este campo se asumen que van a simular los datos de escenarios reales (al fin y al cabo, este es el objetivo final del simulador). Para implementar este campo, se obtienen de la función los valores que se necesiten y se concatenan en el orden explicado en la Tabla 3-6 Grupo fecha-hora. El grupo referido al mes, se obtiene mediante el uso de un diccionario en el que a cada mes le corresponde su código en este formato.

3.3.5.2 Código del campo DELTA 2

Para la función que devuelve el campo DELTA 2, nombrada como `getMessageDeltaField2`, se ha programado para incrementar el grupo fecha-hora de DELTA 1 mediante la entrada de valores, divididos en días, horas y minutos, que es lo que se ha estimado necesario en un principio. Para una correcta expresión del formato ha sido necesario hacer evaluaciones de cuando de cambia de hora, día, mes o año. Por ejemplo, si se tiene el grupo fecha-hora 212000ZFEB2021 y transcurren cinco horas el resultado ha de ser 220100ZFEB2021, en lugar de 212500ZFEB2021. Se ha tenido en cuenta los días que tiene cada mes, al igual que los años bisiestos.

3.3.6 Código del módulo FOXTROT

En el módulo FOXTROT, implementado en la función `getMessageFoxtrot`, transmite los datos de localización de la fuente y el índice de localización, para ello será necesario aportarle los datos de la fuente.

1) El primer campo será la localización de la fuente y se realiza de forma completamente análoga a la del sensor.
2) El segundo campo se hará mediante una variable local con una numeración que será traducida por un diccionario.

3.3.7 Código de módulos diccionario

Los módulos GOLF, INDIA, MIKER, TANGO, YANKEE y ZULU son módulos cuyos datos se obtienen mediante campos de diccionarios. Estos datos se extraen creando una variable local para cada uno y extrayendo el valor asociado a esta. En un futuro podría implementarse un menú de configuración del entorno en el que el propio instructor decida los valores que toman éstos.

3.4 Implementación del código del cliente de mensajes

Después de la redacción de los mensajes CBRN, éstos son recibidos por el cliente como una cadena de texto. La finalidad del cliente es extraer datos de esta cadena, para poder pasarlos a un módulo de representación. Es importante resaltar esto último, pues el cliente en ningún momento debe tener acceso a los datos de simulación, el cliente tomará como reales los datos trasmitidos en el mensaje y tras la consecución varios mensajes, decidirá qué datos se ajustan más a la realidad. Por ejemplo, el Centro de Control actuará como cliente, y el sensor que da la alerta como servidor. El cliente recibe una alerta, y tras procesarlo realiza una representación de la situación, si más tarde recibe datos que no se corresponden con los anteriores, decidirá en base a los parámetros de confianza reflejados y las predicciones elaboradas en el Centro de Control para dar con la versión más realista de la situación.

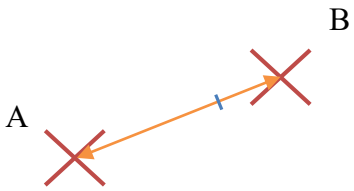

Figura 3-4 Interpretación de la posición de una fuente por el Centro de Control

Como se puede ver en la Figura 3-4, entre dos sensores que reportar posiciones de la misma fuente de manera contradictoria, el Centro de Control asignará unos parámetros basados en la precisión que el sensor puede aportar en esa medición y ponderará los datos para dar con la posición más ajustada. En el caso de la Ilustración, la posición B será más fiable que la de A y por ello se sitúa más cerca de aquélla que de ésta.

Para proceder al análisis de mensaje recibido habrá que remontarse al apartado 2.3.4.2, en el que se hacía ver el formato de redacción y que ahora se usará para la atomización y análisis del mensaje.

3.4.1 Atomización del mensaje recibido

Los mensajes se reciben como una cadena de texto, sin embargo, es conocido el orden de los elementos y los caracteres usados para la separación de los mismos. Por ello el objetivo es convertir la cadena original en una lista de elementos, cuyo orden es conocido, para acabar resolviendo problemas lo más pequeños posible.

Para la primera división, la del mensaje en módulos, se usará la función `split`, la cual recorrerá la cadena de principio a fin, creando un nuevo elemento de la lista cada vez que detecte los caracteres que tiene como parámetros:

```
module_complete_lines = mqtt_message.split(self.cbrn_module_separator)
```

En este punto el mensaje está dividido en módulos y puede recorrerse de la siguiente forma:

```
for module_complete_line in module_complete_lines:
```

Esto hace que el código salte de módulo a módulo, por lo tanto, en cada uno de los módulos se puede repetir la operación ejecutada para dividir el mensaje y en este caso dividir los módulos:

```
field_data = module_complete_line.split(self.cbrn_field_separator)
```

Ahora ya se puede evaluar el primer elemento del nuevo vector creado y compararlo con los nombres de los módulos para identificar cada uno. A continuación, se explicará la redacción de las funciones asociadas al análisis de los campos y módulos que se han realizado en este trabajo, teniendo en cuenta que se han realizado aquellos que aportan los datos mínimos necesarios para una representación en la aplicación de cliente y aquéllos que son más útiles para el Centro de Control.

3.4.2 Análisis módulo ALFA

En el análisis del módulo serán de interés el tercer y cuarto campo, que contienen el número de secuencia del mensaje y el tipo de incidente que está teniendo lugar. Para ello simplemente se accede a la posición 3 y 4 y se almacenan como una lista en una variable `extract_message_alfa.`

3.4.3 Análisis módulo BRAVO

En el módulo BRAVO se encuentran las coordenadas del observador, en el caso del simulador el sensor que emite el mensaje. El procesado de coordenadas será implementado en una función propia, la cual será llamada a lo largo de los mensajes CBRN, se ha tomado esta decisión en base a la cantidad de coordenadas transmitidas, no solo en estos primeros mensajes, sino a lo largo de toda la serie CBRN. La función será llamada `extractLatLongFromText`, recibe un único parámetro de entrada, el cual será la coordenada en el formato del mensaje y devuelve una lista de dos caracteres flotantes, que serán las coordenadas del punto, las cuales se almacenarán en la variable `extract_message_bravo`. Posteriormente se detallará la implementación de esta función.

3.4.4 Análisis módulo DELTA

El módulo delta contiene un grupo fecha-hora en el primero de los campos que será el inicio del incidente. Para la extracción de la información que este módulo contiene se ha construido la función `extractMessageDelta`, que recibirá un parámetro de cadena de caracteres que será la fecha en formato militar y devolverá cuatro parámetros que serán hora, día, mes y año, grabándolos en `extract_message_delta`. Para procesar la información condensada en el campo se precisa la segmentación del campo. La forma escogida ha sido la reflejada en el siguiente ejemplo:

```
day = original_text [:2]
```

En este caso se está almacenando en la variable `day` los dos primeros caracteres de `original_text`, que es el grupo fecha-hora recibido, además estos dos caracteres son eliminados de `original_text` con la siguiente operación:

```
dtg_without_day = original_text [2:]
```

De esta forma se va fraccionando el campo y extrayendo información, sin embargo, la información asociada al mes habrá que descifrarla, pues se utilizó un diccionario para adaptar el carácter numérico al código de tres letras del formato. Para esta operación se ha creado una función que trasponga los diccionarios, consiguiéndose así un máximo provecho de las variables ya declaradas.

3.4.5 Análisis módulo FOXTROT

La extracción del módulo FOXTROT, contenedor de la posición de la fuente, se construye de forma completamente análoga a la del módulo BRAVO, utilizando la función `extractLatLongFromText` a la que se le pasará como parámetro la posición de la fuente en formato [14], y se devuelven las dos coordenadas del punto.

3.4.6 Análisis módulo YANKEE

En la función `extractMessageYankee` se extraerán los datos relativos al viento. Para ello hay que atender al formato en el que se encuentran estos dos datos, velocidad y dirección del viento. La forma es de 1 a 4 dígitos para la magnitud de la medida, seguido de los caracteres de unidades. Para ello se ha hecho una lectura carácter a carácter de cada uno de los campos. La condición para separar la unidad de medida de su magnitud será la condición de que este símbolo sea un número, en caso afirmativo se almacena en una variable, la cual luego se devuelve como carácter entero.

3.4.7 Extracción de coordenadas

A lo largo de los mensajes CBRN es recurrente la utilización de coordenadas, sin embargo, el formato utilizado no es gestionable por la aplicación. Por ello se ha creado una función `extractLatLongFromText` que directamente realice la conversión del formato y pueda ser llamada durante el resto del código.

El primer paso será recorrer carácter a carácter la cadena de texto recibida, con la finalidad de llevar a cabo una evaluación del signo de cada una de las coordenadas. Las condiciones de la apreciación del será encontrar los símbolos ‘N’, ‘S’, ‘E’ y ‘W’.

El segundo paso será extraer el valor numérico absoluto de la coordenada. El método empleado ha sido la división de la cadena por los símbolos alfabéticos mostrados en el párrafo anterior.

Por último, se convertirán los valores conseguidos a carácter flotante y se cambiará el signo o no en función de la evaluación realizada en el primero de los pasos.

4 VALIDACIÓN

Para la validación de la clase CBRN se ha preparado una serie de escenarios que represente cada tipo de incidente. Además, se ha tomado un ejercicio de una publicación del Mando de Adiestramiento y Doctrina [25], la cual, pese a estar algo desactualizada, sirve como referencia a la hora de comparar los resultados iniciales ofrecidos por la herramienta.

4.1 Aplicación a un ejercicio de redacción de mensajes

El enunciado se presenta al alumno, que puede tener acceso a las publicaciones de referencia y apuntes propios.

"Se ha producido un ataque químico en 31TCG 02906702 con unos 20-30 proyectiles de mortero que al explosionar han dispersado un agente identificado, por olor, como CLORO (Cl); el ataque ha tenido lugar a las 0830 Z 10 FEB 2002. El observatorio que envía los datos al NCNBQ. se encuentra situado en el punto de coordenadas 31TCG 03556660. La orientación al GZ., desde el observatorio, es de 5900 milésimas (MLG). El terreno donde se produce la dispersión del agresivo, un valle, está próximo a un río, por lo que la humedad relativa es del 52%, existen bastante árboles y vegetación, pero los alrededores son montañosos y carecen de ella, la temperatura ambiental ronda los 16 grados Celsius. Los datos remitidos por la estación meteorológica, encuadrada en el CG. son:

- *Temperatura a 1,80 metros del suelo: 16º.*
- *Temperatura a 0,30 metros del suelo: 15º.*
- *Velocidad del viento: 12 KPH.*
- *Dirección del viento (hacia dónde sopla): 2700 milésimas (MLG).*
- *Nubosidad: cielo semicubierto.*

La duración del ataque ha sido de un minuto y la persistencia probable del agresivo, en el ambiente, se considera de 10 minutos." [25]

4.1.1 Análisis del enunciado

En el enunciado la mayoría de datos necesarios para la redacción han sido indicados de forma directa, sin embargo, otros han sido indicados de tal forma que el alumno tiene que hacer una evaluación o búsqueda en la publicación para hallar el dato.

Por otro lado, como se ha dicho durante la redacción del trabajo se han hecho algunas simplificaciones en cuanto a los formatos usados. Por ello, se efectuarán las siguientes adaptaciones:

1) Las coordenadas del observador y de la fuente han sido cambiadas por (36.17592, -5.877643) y (36.14945, -5.863523).

2) La hora de inicio del evento no es la del enunciado sino la real en el momento de la simulación.

De la lectura del enunciado se obtiene:

1) Duración del incidente: 1 minuto
2) Índice observado-sospechado: Observado
3) Medio de lanzamiento: Mortero
4) Contenedor del agente: Proyectil
5) Tamaño de la dispersión: 20-30 contenedores
6) Altura de lanzamiento: Superficie
7) Tipo de sustancia: Agente asfixiante
8) Nivel de persistencia: No persistente
9) Descripción topográfica: Valle
10) Descripción de la vegetación: Arbolado
11) Dirección del viento: 2700º
12) Velocidad del viento: 12km/h

4.1.2 Simulación del ejercicio

En la Figura 4-1, se muestran las condiciones iniciales, en este caso un sensor químico en la playa del Campo de Adiestramiento Sierra del Retín y al NW la posición de la fuente química.

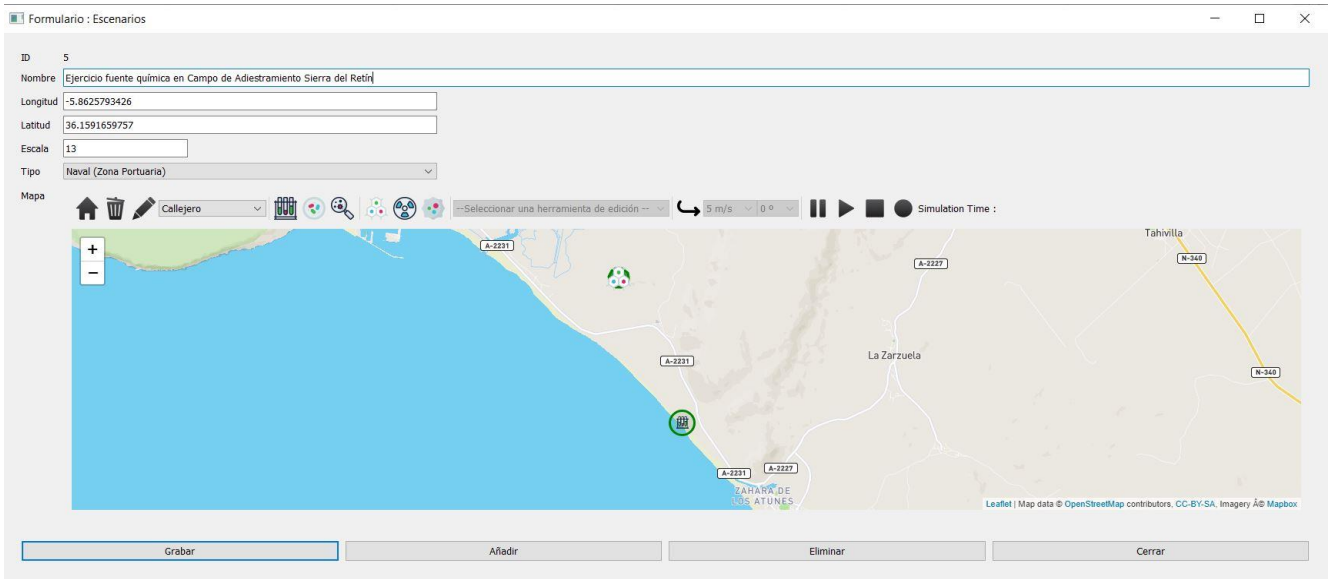

Figura 4-1 Condiciones iniciales de un ejercicio de redacción de CBRN 1 CHEM

En la siguiente figura se puede observar la nube dispersándose con geometría circular.

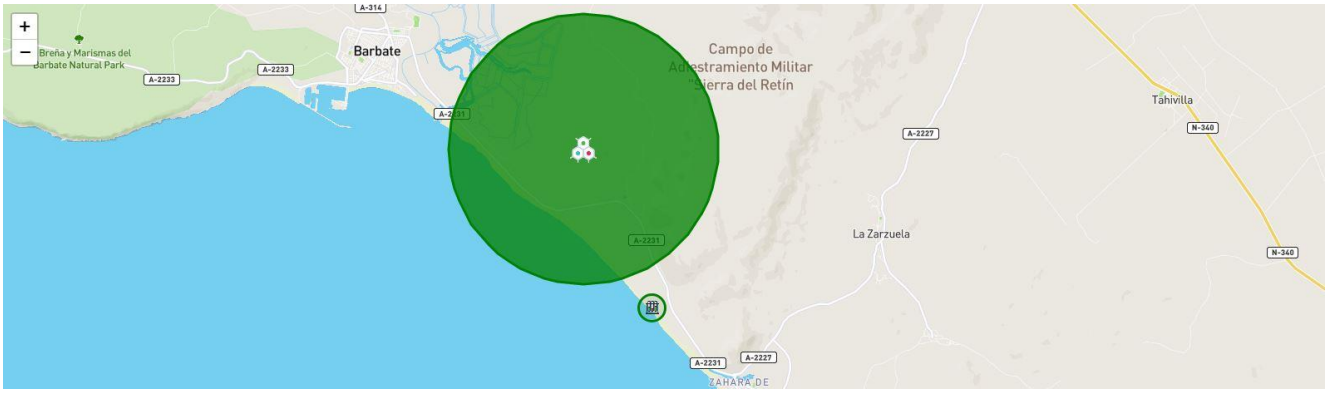

Figura 4-2 Evolución de la nube de un ejercicio de redacción de CBRN 1 CHEM

A partir de este escenario, en el momento en que el sensor detecta la contaminación se genera el siguiente mensaje:

CBRN 1 CHEM//

ALFA/ESP/-/1/CHEM//

BRAVO/36.14945N5.863523W/5903//

DELTA/220942ZFEB2021/220943ZFEB2021//

FOXTROT/36.17592N5.877643W/EE//

GOLF/OBS/MOR/NKN/SHL/20-30/SURF//

INDIA/CHOK/NP/OTH/-//

MIKER/-/-/-//

TANGO/VALLEY/WOODS//

YANKEE/2700MLG/12KPH//

ZULU/S/16C/5/-/1//

GENTEXT/-/EL AGENTE DISPERSADO HA SIDO IDENTIFICADO POR EL OLOR COMO CLORO (Cl). SU TIEMPO PROBABLE DE PELIGRO SE CONSIDERA DE 10 MINUTOS. //

La aplicación empleada en el Centro de Control, muestra un mapa de la Zona de Operaciones general (que en la configuración actual se ha decidido que sea la Península Ibérica) hasta que recibe un mensaje de un incidente.

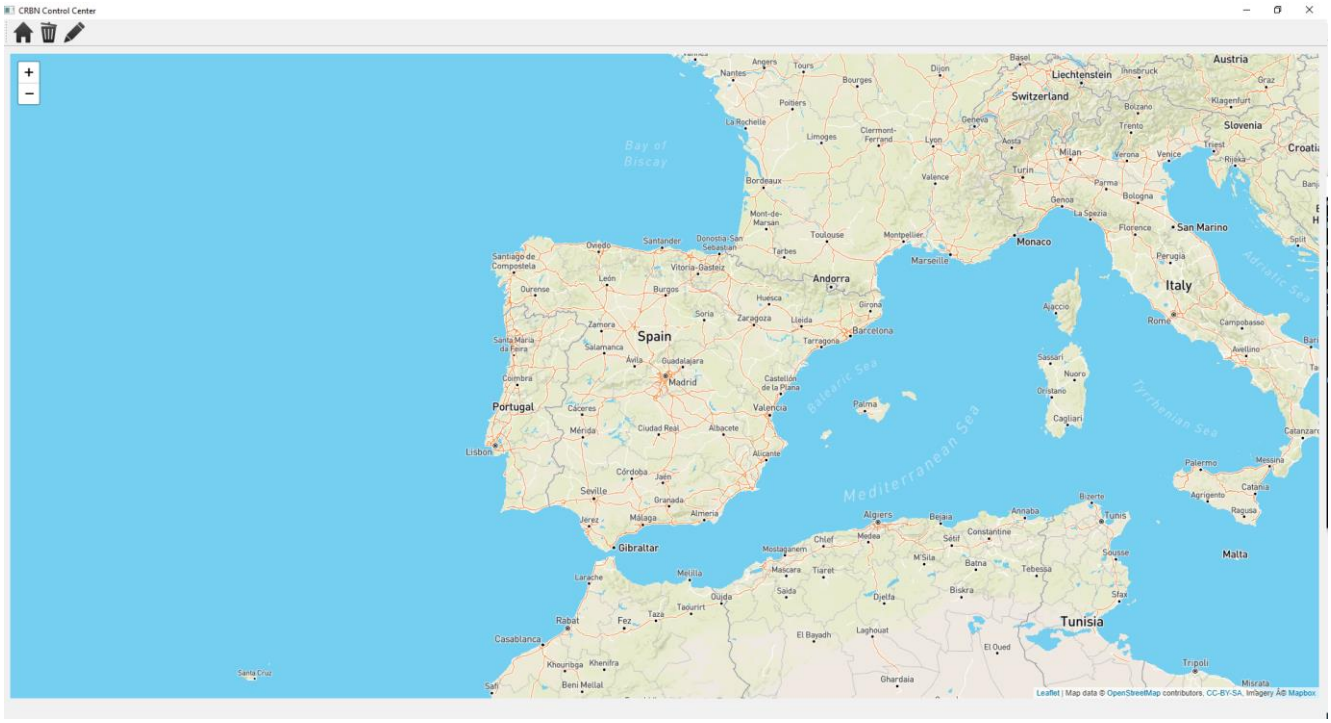

Figura 4-3 Estado inicial de la interfaz del receptor

En ese momento se procesa el mensaje recibido, obteniendo nuevos datos de representación, los cuales se emplean para localizar la ventana del mapa en la zona del incidente y mostrar los sensores que ha activado la detección. Por otro lado, los sensores mostrarán el mensaje de alerta que estén transmitiendo, de forma que el usuario puede consultar el último mensaje CBRN asociado a cada sensor

activo. Podemos ver esta situación en la Figura 4.4, en donde tras recibir la detección el software del centro de control actualiza su localización para alertar sobre una nueva alarma recibida.

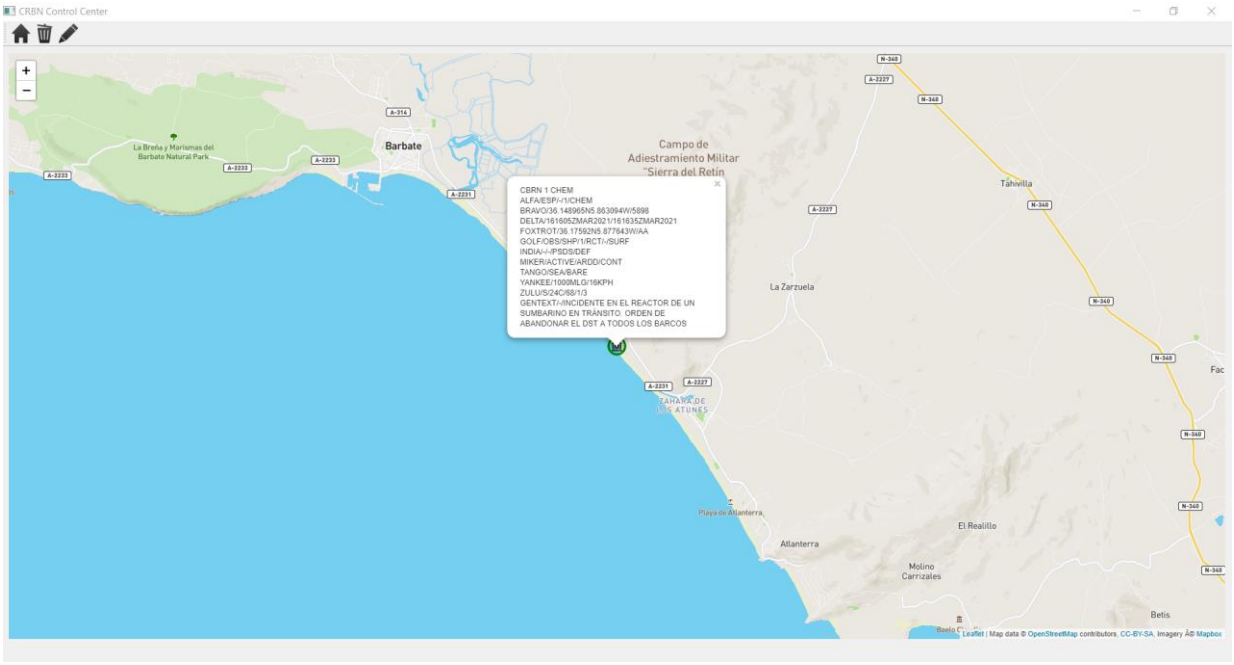

Figura 4-4 Recepción de informe de un ejercicio de redacción de CBRN 1 CHEM

4.2 Aplicación a un ataque químico en la Escuela Naval Militar

A continuación, se plantea el escenario de la Escuela Naval Militar situada en Marín. Como se puede ver en la Figura 4-5, se ha planificado un despliegue de sensores de diferentes tipos, usando la Isla de Tambo para colocar sensores de alerta temprana, se han colocado sensores biológicos en los tanques de agua y un despliegue perimetral de sensores químicos. La fuente se ha programado como química, por ello solo los sensores químicos reaccionarán con la nube y emitirán mensajes.

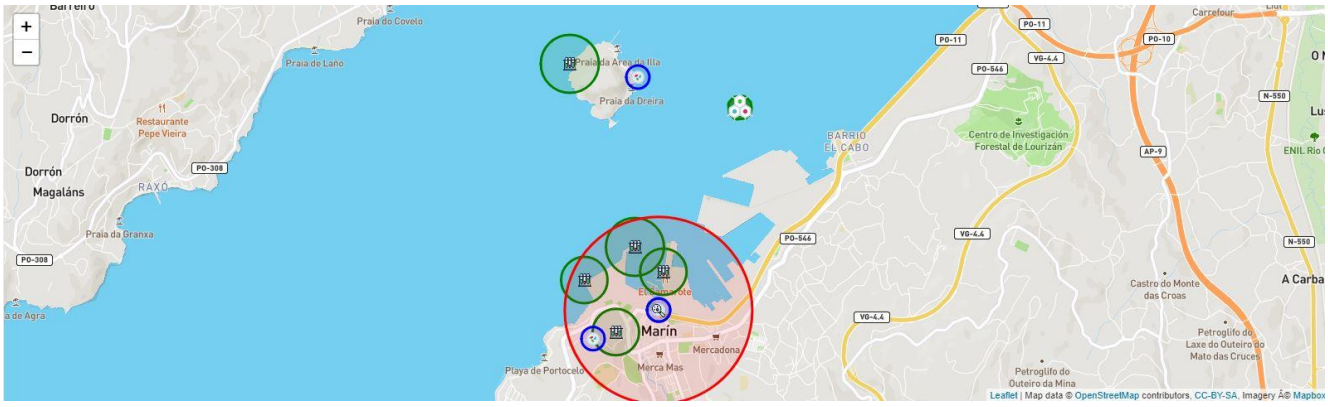

Figura 4-5 Situación inicial de un ataque químico en la Escuela Naval Militar

A continuación, se ve la dispersión de la nube y como va alcanzando a los sensores. En este caso se dispone de sensores de diferentes tipos, de esta forma, solamente aquellos que sean del mismo tipo emitirán el mensaje CBRN, es decir, para el caso que estamos tratando, solo los sensores químicos detectarán la nube química.

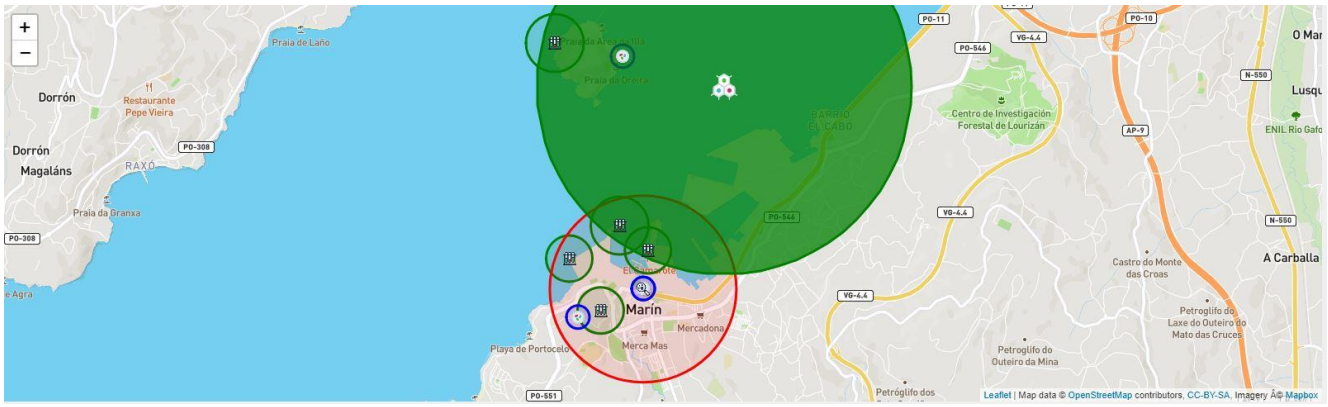

Figura 4-6 Evolución de un ataque químico en la Escuela Naval Militar

Tras producirse la colisión entre los sensores y la nube, cada uno emitirá un mensaje como el siguiente, el cual fue emitido por uno de ellos.

CBRN 1 CHEM//

ALFA/ESP/-/1/CHEM//

BRAVO/42.398678N8.70542W/810//

DELTA/261825ZFEB2021/261835ZFEB2021//

FOXTROT/42.409326N8.694563W/EE//

GOLF/OBS/DEV/1/BTL/5/SURF//

INDIA/H/T/PSDS/EVI//

MIKER/ACTIVE/CLOUD/CONT//

TANGO/SEA/BARE//

YANKEE/3000MLG/10KPH//

ZULU/U/10C/70/6/2//

GENTEXT/-/EL AGENTE DISPERSADO HA SIDO IDENTIFICADO COMO GAS MOSTAZA LANZADO DESDE UN BUQUE EN TRÁNSITO. TODO EL PERSONAL DE LA ENM ESTÁ SIENDO ALISTADO CON EPI//

En este caso el receptor recibirá los mensajes únicamente de los sensores químicos, permaneciendo el resto de ellos ocultos para él, tal y como podemos ver en la Figura 4.7.

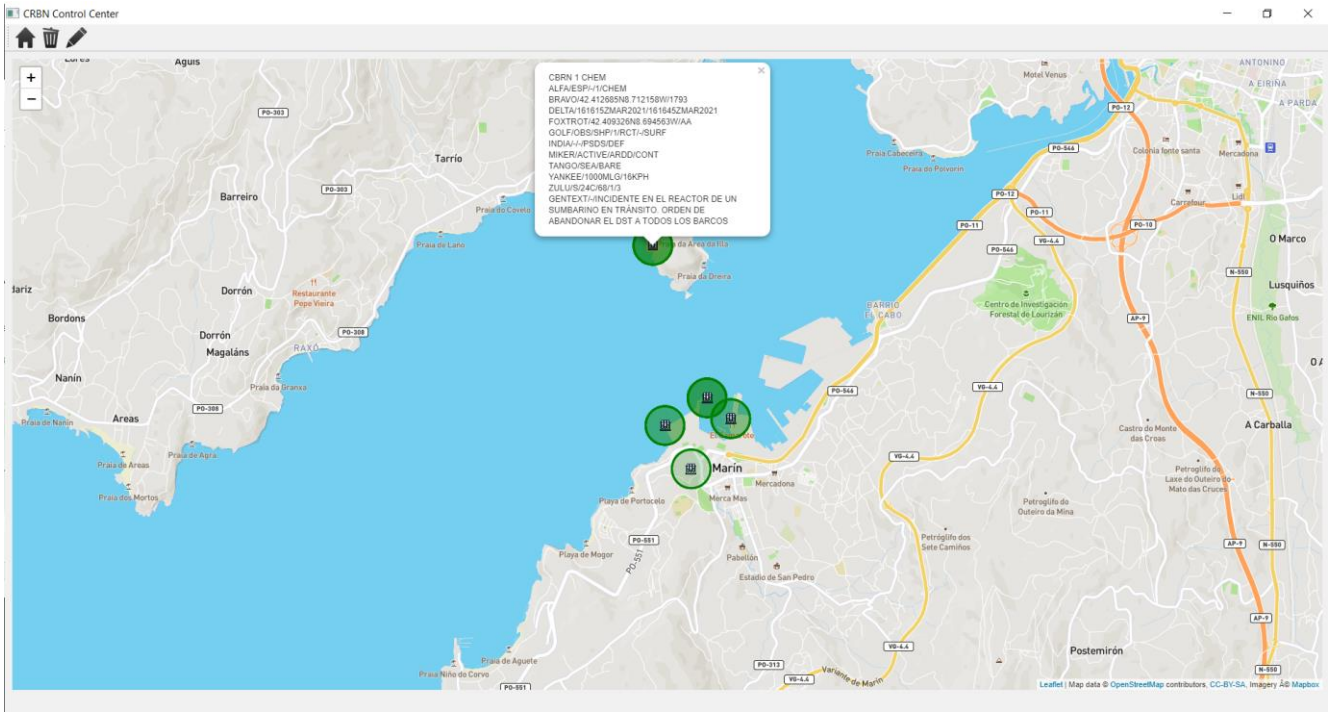

Figura 4-7 Recepción de informe de un ataque químico en la Escuela Naval Militar

4.3 Aplicación a un incidente nuclear en un submarino en el Estrecho de Gibraltar

El Estrecho de Gibraltar es el paso que comunica el Océano Atlántico con el Mar Mediterráneo, una zona estratégica y de tránsito para más de 100.000 barcos al año, entre todos ellos, puede darse la situación de cruce de submarinos nucleares. En este caso, Figura 4-8, se ha simulado un incidente nuclear en uno de ellos, habiendo colocado sensores en la costa ibérica y ceutí.

Figura 4-8 Situación inicial de un incidente nuclear en el Estrecho de Gibraltar

En este caso la propagación de la fuente se produce de forma rápida y violenta, logrando alcances considerables y reportándose el incidente desde múltiples posiciones a lo largo del litoral.

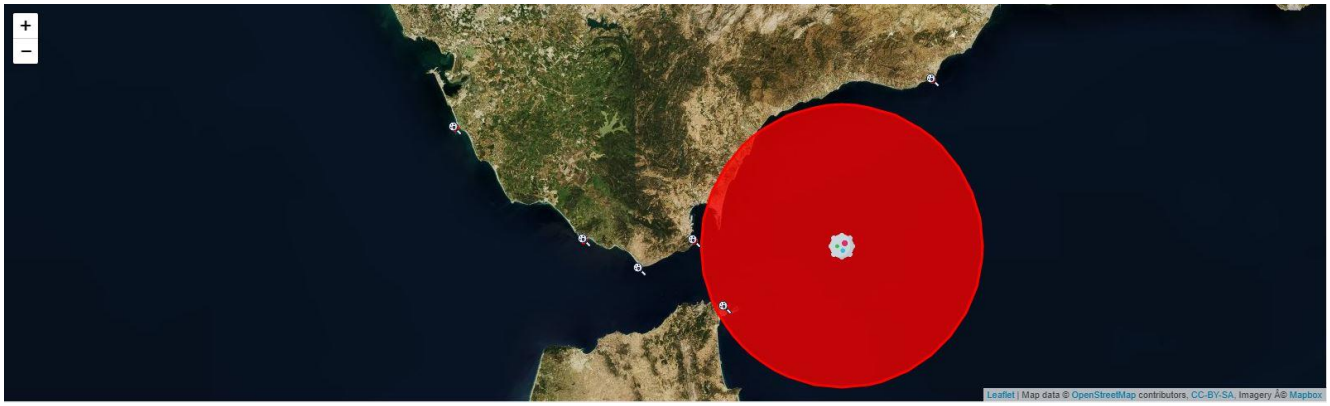

Figura 4-9 Evolución de un incidente nuclear en el Estrecho de Gibraltar

Uno de los mensajes emitido para reportar este evento nuclear será el siguiente:

CBRN 1 NUC//

ALFA/ESP/-/9/NUC//

BRAVO/36.508118N4.637604W/3809//

DELTA/261910ZFEB2021/262010ZFEB2021//

FOXTROT/36.064642N4.938354W/AA//

GOLF/OBS/SHP/1/RCT/-/SURF//

INDIA/-/-/PSDS/DEF//

MIKER/ACTIVE/ARDD/CONT//

TANGO/SEA/BARE//

YANKEE/1000MLG/16KPH//

ZULU/S/24C/68/1/3//

GENTEXT/-/INCIDENTE EN EL REACTOR DE UN SUMBARINO EN TRÁNSITO. ORDEN DE ABANDONAR EL DST A TODOS LOS BARCOS//

4.4 Aplicación a un incidente en la Central Nuclear de Cofrentes

En Cofrentes (Valencia) se sitúa la Central Nuclear con mayor potencia eléctrica instalada de España, con 1.092 MWe[18]. Se sitúa a orillas del Río Júcar y cuenta con una red de sensores desplegados a lo largo de su cauce para la monitorización de niveles de radiación.

[18] MW eléctrico

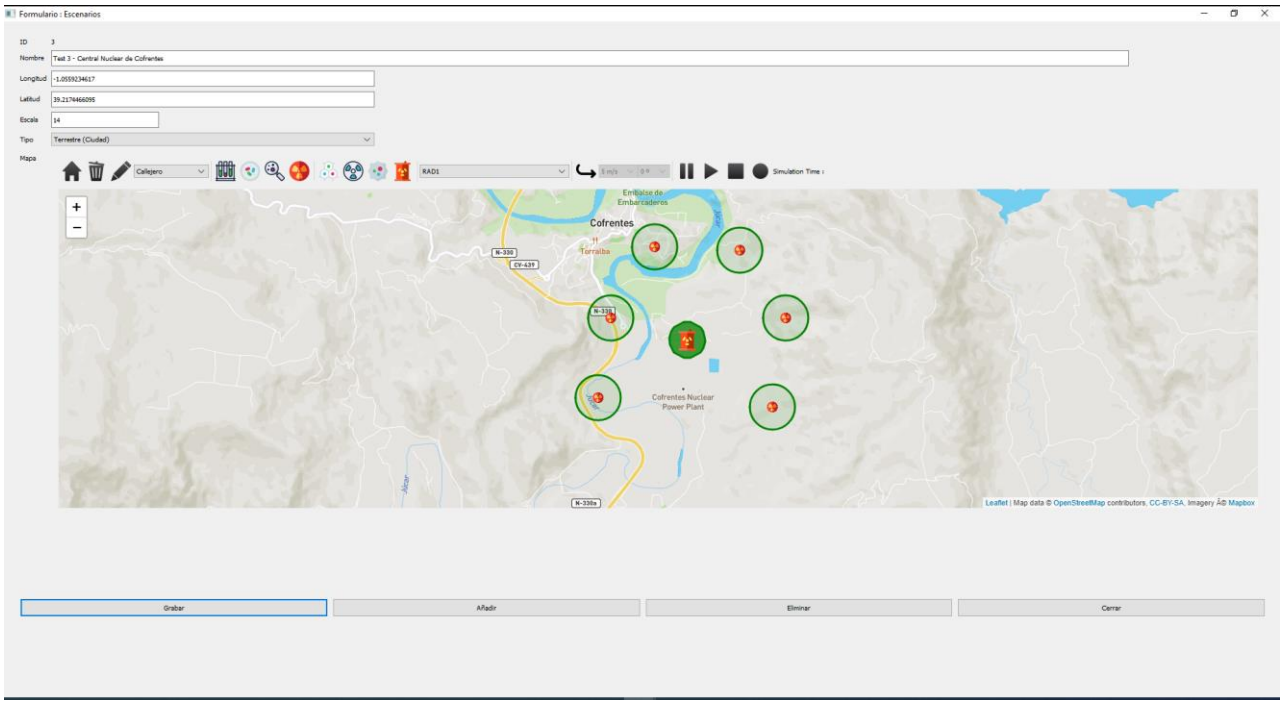

Figura 4-10 Situación inicial de un incidente radiológico en la Central Nuclear de Cofrentes

A continuación, la fuente comienza a dispersarse y a alcanzar el despliegue de sensores.

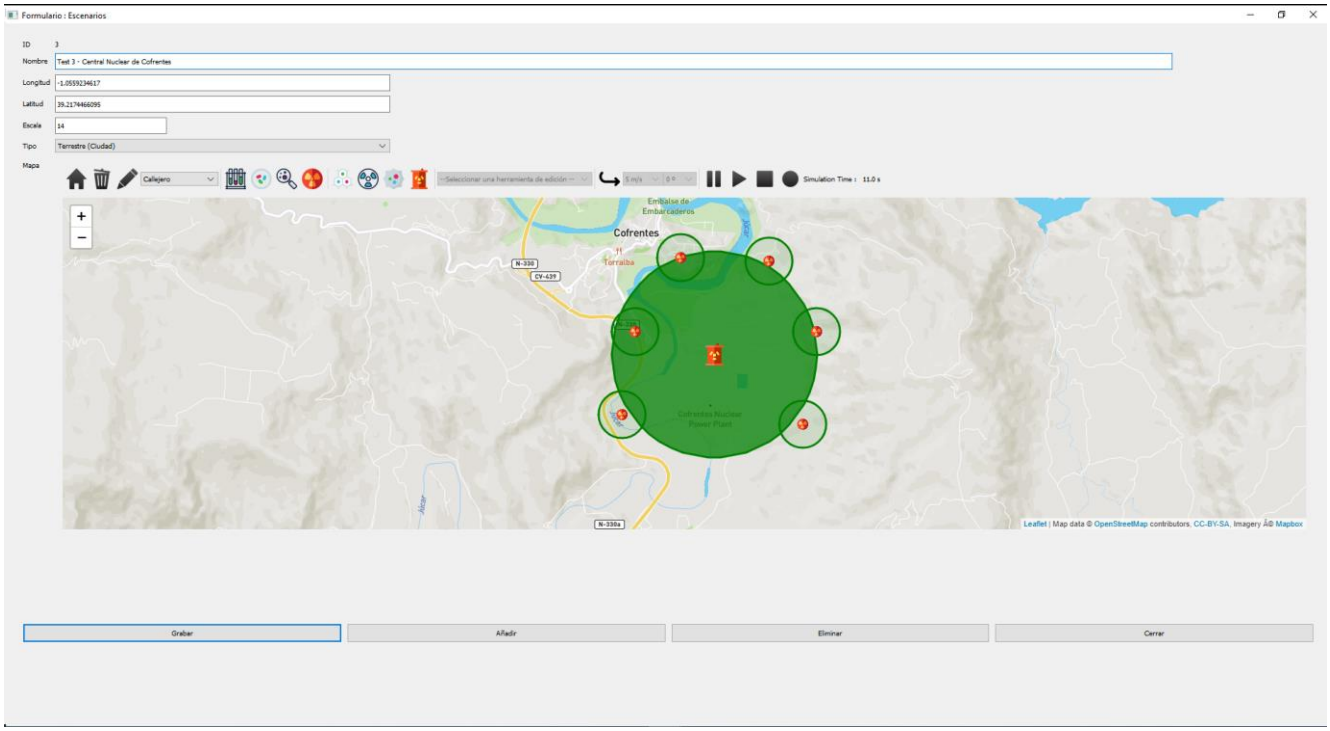

Figura 4-11 Evolución de un incidente radiológico en la Central Nuclear de Cofrentes

Entonces en el Centro de Control se genera una alerta, mostrándose el siguiente escenario.

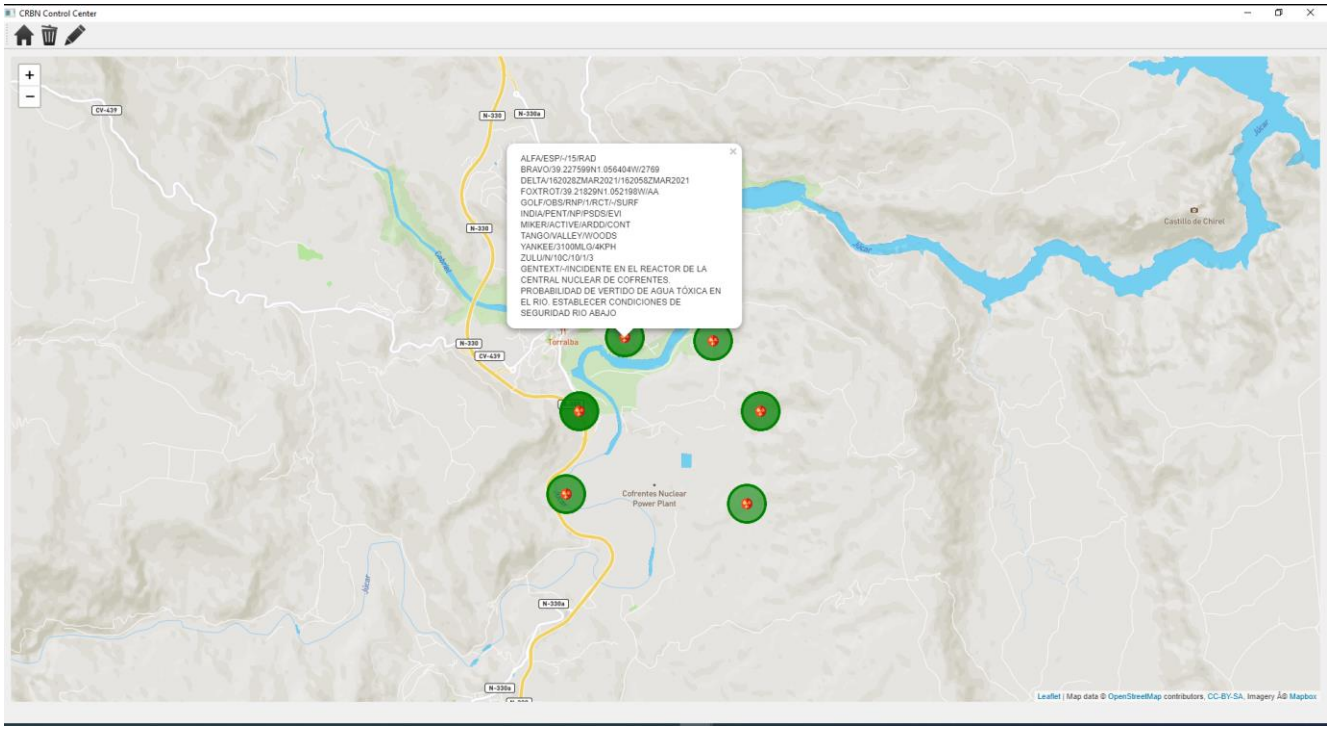

Figura 4-12 Recepción de informe de un incidente en el Laboratorio de Alta Seguridad Biológica del CISA

4.5 Aplicación a un incidente en el Laboratorio de Alta Seguridad Biológica del CISA

El CISA (Centro de Investigación en Sanidad Animal), cuenta con un Laboratorio de Alta Seguridad situado en Valdeolmos desde 1993. Este laboratorio es considerado con de Nivel de Contención Biológica (NCB) 3 según la Organización Mundial de la Salud, y como NCB-4 según la Organización Mundial de Sanidad Animal. Cuenta con investigaciones de virus catalogados de alto riesgo para el ser humano, así como enfermedades altamente trasmisibles de carácter animal y zoonóticas (que podrían transmitirse también a humanos). En la Figura 4-13 se puede observar que se ha colocado un despliegue en dos anillos de detección, el primero con sensores puntuales en zonas sensibles del Laboratorio que proporcionarán una alerta temprana ante cualquier incidente, para el segundo anillo se han utilizado sensores con mayor rango de detección, para

Figura 4-13 Condiciones iniciales de un incidente en el Laboratorio de Alta Seguridad Biológica del CISA

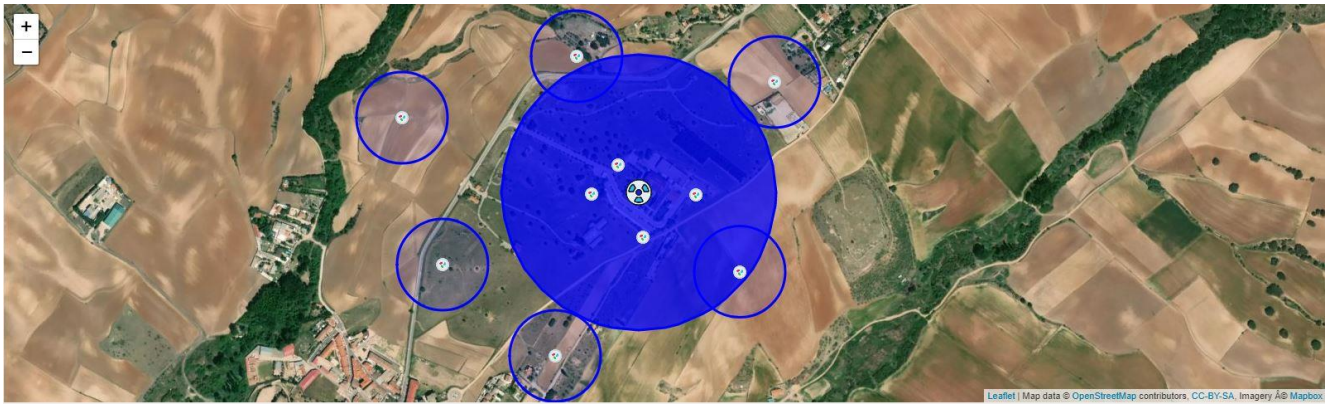

Figura 4-14 Evolución de un incidente en el Laboratorio de Alta Seguridad Biológica del CISA

El mensaje emitido por los sensores del anillo cercano de detección será del tipo:

CBRN 1 BIO//

ALFA/ESP/-/6/BIO//

BRAVO/40.642383N3.446129W/6285//

DELTA/271304ZFEB2021/271334ZFEB2021//

FOXTROT/40.643266N3.446231W/AA//

GOLF/SUS/-/1/NKN/-/SURF//

INDIA/BIO/-/PSDS/EVI//

MIKER/CHECK/-/PUFF//

TANGO/FLAT/SCRUB//

YANKEE/6200MLG/6KPH//

ZULU/S/30C/5/0/3//

GENTEXT/-/INCIDENTE EN LABORATORIO DE ALTA SEGURIDAD BIOLOGICA. EVACUACIÓN DE PERSONAL EN LOCALIDADES A MENOS DE 30KM.//

El escenario mostrado en el cliente asociado al evento CBRN anterior es el mostrado en la Figura 4-12, en donde en el instante mostrado se han activado tres de los sensores del anillo interior.

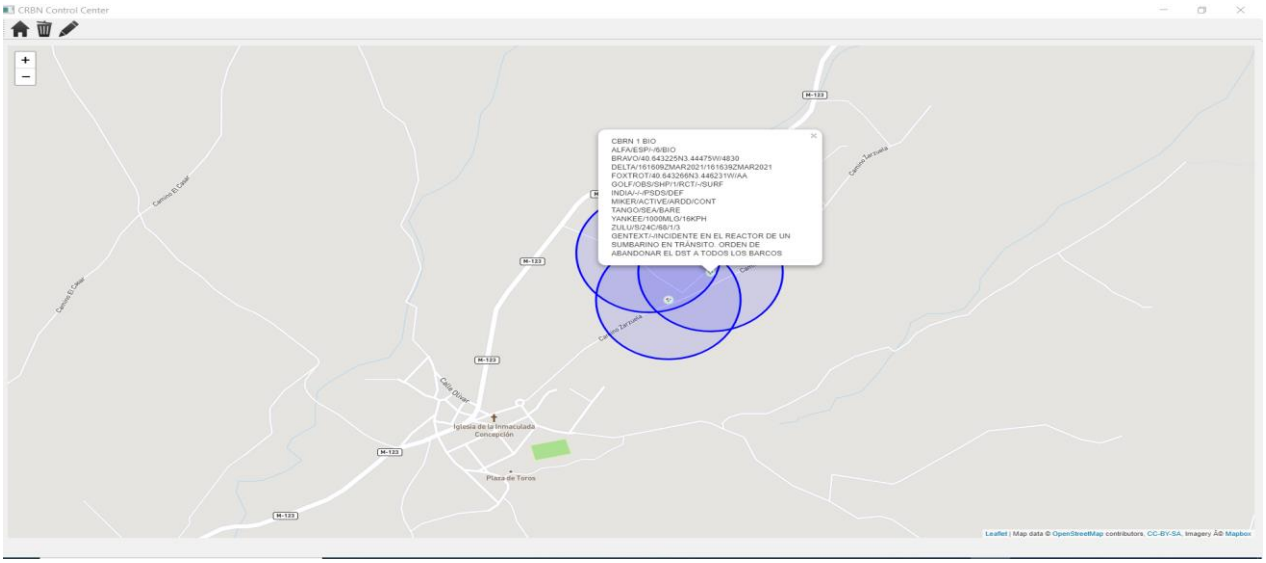

Figura 4-15 Recepción de informe de un incidente en el Laboratorio de Alta Seguridad Biológica del CISA

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

- A lo largo de la elaboración del trabajo se ha podido observar la cantidad de posibilidad que esta línea de desarrollo posee. Además, la decisión de elaborar un software enfocado a adiestramiento encaja con la línea natural de vida de una aplicación de este tipo, pudiendo llegar a convertirse en un programa operativo en Unidades.
- En cuanto a la decisión de construir el simulador sobre Qt ha impactado negativamente en la velocidad de desarrollo de éste, generando en ocasiones dificultades en la resolución de problemas sencillos debido a la inestabilidad de algunas de las librerías.
- El protocolo MQTT ha cumplido con los requerimientos del sistema mostrándose un medio de comunicación robusto.

5.2 Líneas futuras

Tras haber enfrentado un problema de estas características es inevitable pensar en la cantidad de mejoras posibles a realizar pero que por tiempo y alcance del proyecto se ven fuera de rango. Sin embargo, se detallarán a continuación las más relevantes como propuesta para futuras líneas de investigación y desarrollo:

- Ampliación de la herramienta al resto de mensajes CBRN y para completar el protocolo completo de gestión de escenarios de este tipo de situaciones.
- Integración del en plataforma web o soporte móvil.
- Integración de datos atmosféricos reales obtenidos de forma automática de sistemas de información en la internet.
- Integración de información física con módulos SIG[19].
- Integración de modelos de dispersión del agente en diferentes medios (atmosfera, río, mar, etc.).
- Estudio y elaboración de un plan de toma de datos para la elaboración de un *dataset* enfocado a apoyo a técnicas de apoyo a la decisión.
- Integración de bases de datos con características de dispersión de cada agente.
- Aplicación de técnicas de identificación no lineal para estimación de la posición de la fuente, tales como el desarrollo de modelos Fuzzy para la estimación de la incertidumbre en modelos

[19] Sistema de Información Geográfica

predictivos de evolución de nubes contaminantes o el modelado mediante técnicas de análisis basadas en datos de entrada y salida, como por ejemplo modelos NARMAX.

- Diseño de sistemas de optimización de posición para crear redes de sensores robustas y de respuesta rápida.
- Integración de redes de sensores wearables en el personal de seguimiento.

6 BIBLIOGRAFÍA

[1] F. J. Blasco Robledo, «Las armas de destrucción masiva y su trascendencia en el mundo,» *Instituto Español de Estudios Estratégicos,* 2013.

[2] B. Cánovas Sánchez, «Capacidades militares en temas NBQ-R,» de *Las armas NBQ-R como armas de terror*, Centro Superior de Estudios de la Defensa Nacional, 2011.

[3] G. González Martínez, «El terrorismo NBQ-R en la Unión Europea y en España,» de *Las armas NBQ-R como armas de terror*, Centro Superior de Estudios de la Defensa Nacional, 2011.

[4] O. ben Laden, Interviewee, *Osama Claims he Has Nukes: if U.S. uses N-arms it will get the same response.* [Entrevista]. 2001.

[5] O. ben Laden, Interviewee, *Conversation with terror.* [Entrevista]. 1998.

[6] «El Orden Mundial,» 17 11 2020. [En línea]. Available: https://elordenmundial.com/que-es-un-estado-fallido/. [Último acceso: 21 01 2021].

[7] Estados Partes, *Convención sobre la prohibicion del desarrollo, la producción y el almacenamiento de armas bacteriológicas (biológicas) y toxínicas y sobre su destrucción,* 1972.

[8] Estados Partes, *Convención sobre la prohibición del desarrollo, la producción, el almacenamiento y el empleo de armas quimicas y sobre su destruccion,* 1992.

[9] Estados Partes, *Tratado sobre la prohibición de las armas nucleares,* 2017.

[10] Estados Partes, *Protocolo sobre la prohibición del empleo en la guerra de gases asfixiantes, tóxicos o similares y de medios bacteriológicos,* Ginebra, 1925.

[11] Estados Partes, *Tratado de no proliferación nuclear,* 1968.

[12] A. Gouyez ben Allal, *La política nuclear de la OTAN: la amenaza de las armas nucleares tácticas para la seguridad internacional y el régimen de no proliferación nuclear,* Madrid, 2014.

[13] Disposición 12869 del BOE núm. 178 de 2011, *Protocolo de Intervención de la Unidad Militar de Emergencias,* 2011.

[14] NATO, *NATO STANDARD ATP-45, Warning and Reporting and Hazard Prediction of Chemical, Biological, Radiological and Nuclear Incidents (Operators Manual) Edition F Version 1,* 2019.

[15] «Organisation for the Prohibition of Chemical Weapons,» [En línea]. Available: https://www.opcw.org/. [Último acceso: 21 01 2021].

[16] J. M. Cañizares Martínez, *La defensa civil contra incidentes nucleares-biológicos-químicos (N.B.Q.), aplicación a un ataque con una "bomba sucia" en la ciudad de Valencia,* Valencia, 2009.

[17] «Web de NetDoctor,» [En línea]. Available: https://www.netdoctor.es/articulo/malaria. [Último acceso: 09 02 2021].

[18] G. A. Ospina Tascón, *Bioterrorismo: Ántrax,* 2001.

[19] J. Cobo Mora, «Respuesta sanitaria frente a incidentes no convencionales. Generalidades.,» Madrid, 2012.

[20] «Web de Reuters,» [En línea]. Available: https://www.reuters.com/article/idUSKBN25O1U9. [Último acceso: 10 02 2021].

[21] BBC News Mundo, «Web de BBC News,» 28 08 2020. [En línea]. Available: https://www.bbc.com/mundo/noticias-internacional-53951839. [Último acceso: 10 02 2020].

[22] B. Lara Fernández, *Cooperación internacional en la lucha contra el terrorismo nuclear y riesgo de los «Estados fallidos» y de las redes de tráfico ilegal en relación con este terrorismo,* Instituto Español de Estudios Estratégicos, 2011.

[23] Estado Mayor de la Armada, *Instrucciones Generales de Comunicaciones ACP-121 SP NAVY SUPP-1 (A),* 2008.

[24] J. A. Martínez Pons, «Armas químicas: qué son y cómo actúan,» *Anales de la Real Sociedad Española de Química,* 2006.

[25] Mando de Adiestramiento y Doctrina, *NBQ AGBS-TS-003,* 2016.

[26] Basic School Marine Corps Training Command Camp Barret, *Chemical, Biological, Radiological and Nuclear Defense- B2I0413XQ-DM- Student Handout,* 2019.

[27] U.S. Army CBRN School, *CBRN Cookbook,* 2005.

ANEXO I: CBRN CHEM

CBRN 2 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	M	Tabla 3-2
DELTA	Comienzo y finalización de incidente	M	Tabla 3-7
FOXTROT	Localización del incidente	M	Tabla 3-8
GOLF	Medio y cantidad de lanzamiento	M	Tabla 3-10
INDIA	Información de la dispersión en incidentes químicos	M	Tabla 3-16
MIKER	Descripción y estado del incidente	O	Tabla 3-22
TANGO	Descripción del terreno y vegetación	O	Tabla 3-26
YANKEE	Dirección y velocidad del viento	O	Tabla 3-29
ZULU	Condiciones meteorológicas	O	Tabla 3-30
GENTEXT	Texto libre	O	Tabla 3-35

Tabla A1-1 CBRN 2 CHEM

CBRN 3 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	M	Tabla 3-2
DELTA	Comienzo y finalización de incidente	M	Tabla 3-7
FOXTROT	Localización del incidente	M	Tabla 3-8
GOLF	Medio y cantidad de lanzamiento	O	Tabla 3-10
INDIA	Información de la dispersión en incidentes químicos	M	Tabla 3-16
MIKER	Descripción y estado del incidente	O	Tabla 3-22
OSCAR	Periodo de validez de líneas de contorno	C	
PAPAA	Estimación de las zonas de dispersión y peligro	M	
PAPAX	Estimación de la zona de peligro para un periodo meteorológico	M	
TANGO	Descripción del terreno y vegetación	O	Tabla 3-26
XRAYB	Información del contorno estimado	O	
YANKEE	Dirección y velocidad del viento	O	Tabla 3-29
ZULU	Condiciones meteorológicas	O	Tabla 3-30
GENTEXT	Texto libre	O	Tabla 3-35

Tabla A1-2 CBRN 3 CHEM

CBRN 4 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	O	Tabla 3-2
INDIA	Información de la dispersión en incidentes químicos	M	Tabla 3-16
INDIAC	Información del muestreo y dispersión en incidentes químicos	O	
QUEBEC	Localización de la lectura/muestreo/detección y tipo de muestreo/detección	M	
ROMEO	Nivel de contaminación	O	
SIERRA	Grupo Fecha Hora de la medidas de contaminación	M	
TANGO	Descripción del terreno y vegetación	O	Tabla 3-26
WHISKEY	Información del sensor	O	
YANKEE	Dirección y velocidad del viento	O	Tabla 3-29
ZULU	Condiciones meteorológicas	O	Tabla 3-30
XRAYZ	Información del contorno de la nube estimada	O	
OSCARZ	Grupo Fecha Hora del contorno de la nube estimada	O	
GENTEXT	Texto libre	O	Tabla 3-35

Tabla A1-3 CBRN 4 CHEM

CBRN 5 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	M	Tabla 3-2
DELTA	Comienzo y finalización de incidente	O	Tabla 3-7
INDIA	Información de la dispersión en incidentes químicos	M	Tabla 3-16
OSCAR	Grupo Fecha Hora del contorno de la nube	M	
XRAYA	Información del contorno de la nube real	M	
GENTEXT	Texto libre	O	Tabla 3-35

Tabla A1-4 CBRN 5 CHEM

CBRN 6 CHEM			
Módulo	**Descripción**	**Cond.**	**Referencia**
ALFA	Identificación del incidente	O	Tabla 3-2
DELTA	Comienzo y finalización de incidente	O	Tabla 3-7
FOXTROT	Localización del incidente	O	Tabla 3-8
INDIA	Información de la dispersión en incidentes químicos	O	Tabla 3-16
MIKER	Descripción y estado del incidente	O	Tabla 3-22
GENTEXT	Reachback	O	Tabla 3-35
GENTEXT	Texto libre	M	Tabla 3-35

Tabla A1-5 CBRN 6 CHEM

ANEXO II: CLASE CBRN

```
# -*- coding: utf-8 -*-
import traceback, math
from datetime import datetime
from MQTT import MQTT
from re import split
class CBRN():
    def __init__(self):
        super().__init__()
        self.MQTTObj =  MQTT()
        self.topics = None
        self.mqtt_broker = None
        self.mqtt_port = None
        self.cbrn_field_separator = '/'
        self.cbrn_module_separator = '//'
        self.sensor_types = {1: ("Chemical", "CHEM"), 2: ("Biological", "BIO"), 3:  (
"Nuclear", "NUC"), 4:("Radiological", "RAD")} # Read from database table SENSOR_TYPES
 ... to do it!!!
        self.reversed_sensor_types = {"CHEM" : 1, "BIO" : 2, "NUC" : 3, "RAD" : 4 }
        self.location_precision = 1.0e6
        self.month_list = {"01": "ENE","02": "FEB","03": "MAR", "04": "ABR","05": "MA
Y","06": "JUN","07": "JUL","08": "AGO","09": "SEP","10": "OCT","11": "NOV","12": "DIC
",}

        self.example = 0 #0-> ENM; 1-> Estrecho; 2->CISA; 3->Central nuclear; 4-
> Ejercicio

        return

    def __del__(self):
        if not self.MQTTObj is None:
            self.MQTTObj.disconnect()
            del(self.MQTTObj)
    #This function returns th reverse o inversion of a dictionary, it is useful for t
he client that has to decipher
    def reverseDictionary(self, dictionary):
        try:
            reversed_dictionary = {value: key for key, value in dictionary.items()}
            return reversed_dictionary
        except:
            print('CBRN (getMessageHeader) error:', str(traceback.format_exc()))
            return None

    # This function provides the header of the CBRN 1 message, by giving the kind of
incident taking place
    def getMessageHeader(self, scenario_sensor_data):
        try:
```

```
            if scenario_sensor_data['type_id'] in range(4):
                return 'CBRN 1 ' + self.sensor_types[scenario_sensor_data['type_id']]
[1] + self.cbrn_module_separator
            else:
                return None
        except:
            print('CBRN (getMessageHeader) error:', str(traceback.format_exc()))
            return None

    ################################### ALFA  ###################################

    # This function provides the alfa module of the CBRN message
    def getMessageAlfa(self, scenario_sensor_data):
        try:
            return  'ALFA'+ self.cbrn_field_separator + 'ESP' + self.cbrn_field_separ
ator  + '-
' + self.cbrn_field_separator + self.getMessageAlfaField3(scenario_sensor_data) + sel
f.cbrn_field_separator + self.getMessageAlfaField4(scenario_sensor_data) + self.cbrn_
module_separator
        except:
            print('CBRN (getMessageAlfa) error:', str(traceback.format_exc()))
            return '-'
    """
    # This field is only written by the main authority which never is the Control Cen
ter
    def getMessageAlfaField2(self, scenario_sensor_data):
        try:
            return '-'
        except:
            print('CBRN (getMessageAlfaField3) error:', str(traceback.format_exc()))
            return '-'
    """
    #This function provides the unit that makes the message, in our case the id of th
e sensor contained in scenario_sensor_data
    def getMessageAlfaField3(self, scenario_sensor_data):
        try:
            sensor_id = scenario_sensor_data['sensor_id']
            return str(sensor_id)
        except:
            print('CBRN (getMessageAlfaField3) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the unit that makes the message, in our case the type of
sensor reporting
    def getMessageAlfaField4(self, scenario_sensor_data):
        try:
            if scenario_sensor_data['type_id'] in self.sensor_types:
                return self.sensor_types[scenario_sensor_data['type_id']][1]
            else:
                return '-'
        except:
```

```
            print('CBRN (getMessageAlfaField3) error:', str(traceback.format_exc()))
            return '-'
    ################################### BRAVO ###################################
    # This function provides the bravo module of the CBRN message
    def getMessageBravo(self, scenario_source_data, scenario_sensor_data):
        try:
            return  'BRAVO'+ self.cbrn_field_separator  + self.getMessageBravoField1(
scenario_sensor_data) + self.cbrn_field_separator + self.getMessageBravoField2(scenar
io_source_data, scenario_sensor_data) + self.cbrn_module_separator
        except:
            print('CBRN (getMessageBravo) error:', str(traceback.format_exc()))
            return None

    #This function returns the location of the sensor that sends the message

    def getMessageBravoField1(self, scenario_sensor_data):
        try:
            latitude = round(scenario_sensor_data['latitude']*self.location_precision
)
            longitude = round(scenario_sensor_data['longitude']*self.location_precisi
on)
            if latitude > 0:
                sensor_latitude = str(latitude/self.location_precision)+'N'
            elif latitude < 0:
                latitude = abs(latitude)/self.location_precision
                sensor_latitude = str(latitude)+'S'
            if longitude > 0:
                sensor_longitude = str(longitude/self.location_precision)+'E'
            elif longitude < 0:
                longitude = abs(longitude)/self.location_precision
                sensor_longitude = str(longitude)+'W'
            coordinates = sensor_latitude+sensor_longitude
            return coordinates
        except:
            print('CBRN (getMessageBravoField1) error:', str(traceback.format_exc()))
            return '-'

    #This function provides de angle of the source measured from the sensor, expresse
d in mils
    #360degrees = 6400mils
    def getMessageBravoField2(self, scenario_source_data, scenario_sensor_data):
        try:
            sensor_coordinate_x = round(scenario_sensor_data['longitude']*self.locati
on_precision)
            source_coordinate_x = round(scenario_source_data['longitude']*self.locati
on_precision)
            sensor_coordinate_y = round(scenario_sensor_data['latitude']*self.locatio
n_precision)
```

```
            source_coordinate_y = round(scenario_source_data['latitude']*self.locatio
n_precision)
            dif_coordinate_x =  source_coordinate_x - sensor_coordinate_x
            dif_coordinate_y =  source_coordinate_y - sensor_coordinate_y

            if  dif_coordinate_x >= 0 and dif_coordinate_y != 0: #1st and 2nd quadran
ts
                radians = math.atan2(dif_coordinate_x, dif_coordinate_y)
            elif dif_coordinate_x < 0 : #3rd and 4th quadrants
                radians = 2.0*math.pi+ math.atan2(dif_coordinate_x, dif_coordinate_y)
            elif dif_coordinate_x == 0 and dif_coordinate_y == 0: #source and sensor
in the exact same position
                return 'error'
            mils = round (radians*1019.0) #conversion from rad to mils mils = radians
*6400/2pi
            return str(mils)
        except:
            print('CBRN (getMessageBravoField2) error:', str(traceback.format_exc()))
            return '-'
    ################################### DELTA ###################################
    # This function provides the delta module of the CBRN message
    def getMessageDelta(self):
        try:
            return  'DELTA'+ self.cbrn_field_separator  + self.getMessageDeltaField1(
) + self.cbrn_field_separator + self.getMessageDeltaField2() + self.cbrn_module_separ
ator
        except:
            print('CBRN (getMessageDelta) error:', str(traceback.format_exc()))
            return None
    #This function provides the date-time group when sending the message
    #format: DDHHmmhMMMYYYY
    #DD = two ciphers for the day
    #HH = two ciphers for the hour
    #mm = two ciphers for the minutes
    #z = time zone, Z
    #MMM = three letters for the month using the code
    #YYYY = four ciphers for the year
    def getMessageDeltaField1(self):
        try:

            now = datetime.now()
            module1 = now.strftime('%d%H%MZ')
            month = now.strftime('%m')
            year = now.strftime ('%Y')

            month_format = self.month_list[month]
            dtg = module1 + month_format + year
            return dtg
        except:
            print('CBRN (getMessageDeltaField1) error:', str(traceback.format_exc()))
```

```
            return '-'
    #This function provides the date-time group when the incident is over
    def getMessageDeltaField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                duration_days = 0
                duration_hours = 0
                duration_minutes = 30
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                duration_days = 0
                duration_hours = 0
                duration_minutes = 30
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                duration_days = 0
                duration_hours = 0
                duration_minutes = 30
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                duration_days = 0
                duration_hours = 0
                duration_minutes = 30
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                duration_days = 0
                duration_hours = 1
                duration_minutes = 0


            now = datetime.now()
            day = now.strftime('%d')
            day = int(day) + duration_days
            hour = now.strftime('%H')
            hour = int(hour) + duration_hours
            minute = now.strftime('%M')
            minute = int(minute) + duration_minutes
            month = now.strftime('%m')
            month= int(month)
            year = now.strftime ('%Y')
            year = int(year)
            while minute >= 60: #checking if it is a new hour or not/ the same for th
e month and year
```

```
                minute -= 60
                hour += 1
            while hour >= 24:
                hour -= 24
                day += 1
            if year % 4 == 0: #checking if it is a leap-year
                february_days = 29
            else:
                february_days = 28

            if  day > 31 and (month == 1 or 3 or 5 or 7 or 8 or 10 or 12): #months wh
ich have 31 days
                print(day)
                day -= 31
                month += 1
            elif  day > 30 and (month == 4 or 6 or 9 or 11): #months which have 30 da
ys
                day -= 30
                month += 1
            elif  day > february_days and month == 2:
                day -= 28
                month += 1
            if month > 12:
                month=1
                year += 1
            if minute < 10:
                minute = '0'+str(minute)
            else:
                minute = str(minute)
            if hour < 10:
                hour = '0'+str(hour)
            else:
                hour = str(hour)
            if day < 10: #Turning int into str will not give the 0 before numbers bet
ween 0 and 9 directly
                day = '0'+str(day)
            else:
                day = str(day)
            if month < 10: #The dictionary where the month is coded needs the  before
 numbers between 0 and 9
                month = '0'+str(month)
            else:
                month = str(month)
            year = str(year)
            module1 = day+hour+minute+'Z'
            month_format = self.month_list[month]
            dtg = module1 + month_format + year
            return dtg
        except:
            print('CBRN (getMessageDeltaField2) error:', str(traceback.format_exc()))
```

```
            return '-'
    ################################### FOXTROT ###################################
    # This function provides the foxtrot module of the CBRN message
    def getMessageFoxtrot(self, scenario_source_data):
        try:
            return 'FOXTROT'+ self.cbrn_field_separator  + self.getMessageFoxtrotFiel
d1(scenario_source_data) + self.cbrn_field_separator + self.getMessageFoxtrotField2()
 + self.cbrn_module_separator
        except:
            print('CBRN (getMessageFoxtrot) error:', str(traceback.format_exc()))
            return None
    # This function provides the location of the source
    def getMessageFoxtrotField1(self, scenario_source_data):
        try:
            latitude = round(scenario_source_data['latitude']*self.location_precision
)
            longitude = round(scenario_source_data['longitude']*self.location_precisi
on)
            if latitude > 0:
                source_latitude = str(latitude/self.location_precision)+'N'
            elif latitude < 0:
                latitude = abs(latitude)/self.location_precision
                source_latitude = str(latitude)+'S'
            if longitude > 0:
                source_longitude = str(longitude/self.location_precision)+'E'
            elif longitude < 0:
                longitude = abs(longitude)/self.location_precision
                source_longitude = str(longitude)+'W'
            coordinates = source_latitude + source_longitude
            return coordinates
        except:
            print('CBRN (getMessageFoxtrotField1) error:', str(traceback.format_exc()
))
            return '-'
    # This function provides information about how reliable is the location given upp
er
    # Input: location_index
    # Output: FoxtrotField2
    def getMessageFoxtrotField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                location_index = '1'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                location_index = '0'
```

```
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                location_index = '0'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                location_index = '0'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                location_index = '1'

            location_list = {
        "0": "AA",
        "1": "EE",
        "2": "NKN",
        }
            valor =  location_list [location_index]
            return valor
        except:
            print('CBRN (getMessageFoxtrotField2) error:', str(traceback.format_exc()
))
            return '-'
    ################################### GOLF ###################################
    # This function provides the golf module of the CBRN message
    def getMessageGolf(self):
        try:
            return  'GOLF'+ self.cbrn_field_separator  + self.getMessageGolfField1()
+ self.cbrn_field_separator + self.getMessageGolfField2()  + self.cbrn_field_separato
r + self.getMessageGolfField3()  + self.cbrn_field_separator + self.getMessageGolfFie
ld4()  + self.cbrn_field_separator + self.getMessageGolfField5()  + self.cbrn_field_s
eparator + self.getMessageGolfField6() + self.cbrn_module_separator
        except:
            print('CBRN (getMessageGolf) error:', str(traceback.format_exc()))
            return None
    #This function provides information about if the incident was observed or if ther
e are just evidences
    def getMessageGolfField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                detection = 'OBSERVADO'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                detection = 'OBSERVADO'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
```

```
                ###### CISA ######
                detection = 'SOSPECHOSO'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                detection = 'OBSERVADO'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                detection = 'OBSERVADO'

            observed_suspected = {
                "OBSERVADO": "OBS",
                "SOSPECHOSO": "SUS",
            }
            valor = observed_suspected[detection]
            return valor
        except:
            print('CBRN (getMessageGolfField1) error:', str(traceback.format_exc()))
            return '-'
    #This function provides information about which was the mean of delivery
    def getMessageGolfField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                mean = 'DEVICE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                mean = 'SHIP'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                mean = '-'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                mean = 'REACTOR NUCLEAR PLANT'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                mean = 'MORTAR'

            delivery_mean = {
                'AIRCRAFT': 'ACR',
                'BOMB':'BOM',
                'CANNON':'CAN',
                'DEVICE':'DEV',
```

```
                'FUEL FABRICATION FACILITY':'FFF',
                'FISSILE MATERIAL STORAGE':'FMS',
                'FUEL REPROCESSING FACILITY':'FRF',
                'MULTIPLE LAUNCH ROCKET SYSTEM':'MLR',
                'MORTAR':'MOR',
                'MISSILE':'MSL',
                'MISSILE PAY LOAD':'MPL',
                'NOT KNOWN':'NKN',
                'PLANT':'PLT',
                'RAILROAD CAR':'RLD',
                'REACTOR NUCLEAR PLANT':'RNP',
                'RESEARCH NUCLEAR REACTOR':'RNR',
                'RADIOACTIVE WASTE STORAGE':'RWS',
                'SHIP':'SHP',
                'TOXIC INDUSTRIAL RADIOLOGICAL FACILITY':'TIR',
                'TRANSPORT':'TPT',
            }
            valor = delivery_mean[mean]
            return valor
        except:
            print('CBRN (getMessageGolfField2) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the number of sources releasing the agent
    def getMessageGolfField3(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                number_source = '1'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                number_source = '1'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                number_source = '1'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                number_source = '1'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                number_source = 'NKN'

            return str(number_source)
        except:
            print('CBRN (getMessageGolfField3) error:', str(traceback.format_exc()))
            return '-'
```

```
    #Thisfunction provides the type of container releasing the agent
    def getMessageGolfField4(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                container = 'PRESSURIZED GAS BOTTLE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                container = 'REACTOR'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                container = 'NOT KNOWN'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                container = 'REACTOR'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                container = 'SHELL'

            delivery_mean = {
                'PRESSURIZED GAS BOTTLE': 'BTL',
                'BUNKER': 'BUK',
                'GENERIC STORAGE CONTAINER': 'CON',
                'NOMINAL 200 LITRES STORAGE DRUM': 'DRM',
                'GENERATOR (AEROSOL)': 'GEN',
                'INTERMEDIATE BULK CONTAINER': 'IBC',
                'LARGE ISO CONTAINERS': 'ISO',
                'MINE (CBRN FILLED ONLY)': 'MNE',
                'MISSILE PAYLOAD': 'MPL',
                'NOT KNOWN': 'NKN',
                'NUCLEAR WARHEAD': 'NWH',
                'PIPE OR PIPELINE': 'PIP',
                'REACTOR': 'RCT',
                'ROCKET': 'RKT',
                'SHELL': 'SHL',
                'SPRAY (TANK)': 'SPR',
                'STOCKPILE': 'STK',
                'STORAGE TANK': 'TNK',
                'TORPEDO': 'TOR',
                'WASTE': 'WST',
                }
            valor = delivery_mean[container]
            return valor
        except:
```

```
            print('CBRN (getMessageGolfField4) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the total number of containers releasing the agent
    def getMessageGolfField5(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                number_container = '5'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                number_container = '-'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                number_container = '-'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                number_container = '-'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                number_container = '20-30'

            return str(number_container)
        except:
            print('CBRN (getMessageGolfField5) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the height of the dissemination followed by the code of t
he unit of measurement
    def getMessageGolfField6(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                release_height= 'SURFACE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                release_height= 'SURFACE'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                release_height= 'SURFACE'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                release_height= 'SURFACE'
```

```
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                release_height= 'SURFACE'

            list_height_release={
            'AIR' : 'AIR',
            'NOT KNOWN' : 'NKN',
            'SUB SURFACE' : 'SUBS',
            'SURFACE' : 'SURF',
            }
            valor = list_height_release[release_height]
            return valor
        except:
            print('CBRN (getMessageGolfField6) error:', str(traceback.format_exc()))
            return '-'
    ################################### INDIA ###################################
    # This function provides the india module of the CBRN message
    def getMessageIndia(self):
        try:
            return  'INDIA'+ self.cbrn_field_separator  + self.getMessageIndiaField1(
) + self.cbrn_field_separator + self.getMessageIndiaField2()  + self.cbrn_field_separ
ator + self.getMessageIndiaField3()  + self.cbrn_field_separator + self.getMessageInd
iaField4() + self.cbrn_module_separator
        except:
            print('CBRN (getMessageIndia) error:', str(traceback.format_exc()))
            return None
    #This function provides type of substance released
    def getMessageIndiaField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                type_substance='MUSTARD'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                type_substance='-'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                type_substance='BIOLOGICAL'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                type_substance=''
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
```

```
                type_substance='CHOKING AGENT'

            list_substance = {  #Type of agent detected/suspected/observed
                'BACTERIAL' : 'BAC',
                'BIOLOGICAL' : 'BIO',
                'BLISTER AGENT' : 'BL',
                'BLOOD' : 'BLOD',
                'CHOKING AGENT' : 'CHOK',
                'CHLAMYDIA' : 'CLA',
                'NUCLEAR WEAPON FALLOUT' : 'FL',
                'G AGENT' : 'G',
                'MUSTARD' : 'H',
                'INCAPACITATING' : 'INCP',
                'IRRITANT' : 'IRT',
                'NERV AGENT' : 'NERV',
                'NOT KNOWN' : 'NKN',
                'NUCLEAR WARHEAD' : 'NWH',
                'PENETRATING AGENT' : 'PENT',
                'RICKETTSIAE' : 'RIC',
                'TOXIC INDUSTRIAL BIOLOGICAL' : 'TIB',
                'TOXIC INDUSTRIAL CHEMICAL' : 'TIC',
                'TOXIN' : 'TOX',
                'V-AGENT' : 'V',
                'VIRAL' : 'VIR',
                'VOMITING AGENT' : 'VMT',
            }
            valor = list_substance[type_substance]
            return valor
        except:
            print('CBRN (getMessageIndiaField1) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the volatility of the agent
    def getMessageIndiaField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                persistency = 7
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                persistency = '-'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                persistency = 2
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                persistency = '-'
```

```
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                persistency = 3

            if persistency < 3.33 and persistency >= 0 :
                return 'NP'
            elif persistency > 3.33 and persistency < 6.66:
                return 'P'
            elif persistency > 6.66 and persistency <= 10:
                return 'T'
            else:
                return 'NKN'
        except:
            print('CBRN (getMessageIndiaField2) error:', str(traceback.format_exc()))
            return '-'
    #This function provides how the agent was detected
    def getMessageIndiaField3(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                type_detection='PASSIVE STAND-OFF DETECTION SYSTEM'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                type_detection='PASSIVE STAND-OFF DETECTION SYSTEM'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                type_detection='PASSIVE STAND-OFF DETECTION SYSTEM'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                type_detection='PASSIVE STAND-OFF DETECTION SYSTEM'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                type_detection='OTHER'

            list_detection = {  #Type of agent detected/suspected/observed
                'DEPLOYED LABORATORY' : 'DL',
                'POINT DETECTION SYSTEM' : 'PDS',
                'ACTIVE STAND-OFF DETECTION SYSTEM' : 'ASDS',
                'PASSIVE STAND-OFF DETECTION SYSTEM' : 'PSDS',
                'REMOTE DETECTION' : 'RD',
                'SATELLITE-BASED DETECTION' : 'SBD',
                'OTHER' : 'OTH',
            }
```

```
            valor = list_detection[type_detection]
            return valor
        except:
            print('CBRN (getMessageIndiaField3) error:', str(traceback.format_exc()))
            return '-'
    #This function provides info about how reliable is the detection of the agent
    def getMessageIndiaField4(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                confidence_level='EVIDENTIAL'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                confidence_level='DEFINITIVE'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                confidence_level='EVIDENTIAL'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                confidence_level='EVIDENTIAL'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                confidence_level='-'

            list_confidence = {
                'DEFINITIVE' : 'DEF',
                'EVIDENTIAL' : 'EVI',
                'INDICATIVE' : 'IND',
                'PRESUMPTIVE' : 'PRE',
            }
            valor = list_confidence[confidence_level]
            return valor
        except:
            print('CBRN (getMessageIndiaField4) error:', str(traceback.format_exc()))
            return '-'
    ################################### MIKER ###################################
    # This function provides the miker module of the CBRN message
    def getMessageMiker(self):
        try:
            return  'MIKER'+ self.cbrn_field_separator  + self.getMessageMikerField1(
) + self.cbrn_field_separator + self.getMessageMikerField2()  + self.cbrn_field_separ
ator + self.getMessageMikerField3() + self.cbrn_module_separator
        except:
            print('CBRN (getMessageMiker) error:', str(traceback.format_exc()))
            return None
```

```
    #This function provides information about the origin of the incident
    def getMessageMikerField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                activity='ACTIVE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                activity='ACTIVE'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                activity='CHECK'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                activity='ACTIVE'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                activity='-'

            event_activity = {
                'ACTIVE':'ACTIVE',
                'CONTINUED':'CONTD',
                'CHECK':'CHECK',
                'CLEAR':'CLEAR',
            }
            valor = event_activity[activity]
            return valor
        except:
            print('CBRN (getMessageMikerField1) error:', str(traceback.format_exc()))
            return '-'

    #This function provides information about the origin of the incident
    def getMessageMikerField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                description_incident='VISIBLE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                description_incident='ACTIVATED RADIOLOGICAL DISPERSION DEVICE'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
```

```
                ###### CISA ######
                description_incident='-'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                description_incident='ACTIVATED RADIOLOGICAL DISPERSION DEVICE'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                description_incident='-'

            list_description = {
                'ACTIVATED RADIOLOGICAL DISPERSION DEVICE':'ARDD',
                'VISIBLE':'CLOUD',
                'DAMAGED PACKAGE AND CONTAMINATION':'DPC',
                'EVIDENCE OF SITE DISRUPTION':'ESD',
                'EXPLOSIONS AND FIRE':'EXFIRE',
                'EXPOSED SOURCE':'EXS',
                'BURNING FIRE':'FIRE',
                'INTACT PACKAGE OR DEVICE':'INT',
                'SUBSTANCE SPILLED INTO WATER':'INWAT',
                'CONTINUOUS FLOW FROM DAMAGED PIPE OR CONTAINER':'LEAK',
                'LIQUID':'LIQUID',
                'MISSILE INTERCEPT':'MSLINT',
                'NON ACTIVATED RADIOLOGICAL DISPERSION DEVICE':'NARDD',
                'LARGE QUANTITY OF STILL LIQUID':'POOL',
                'CATASTROPHIC RUPTURE OF A TANK':'RUP',
                'SMALL QUANTITY OF STILL LIQUID':'SPILL',
                'SOLID':'SOLID',
                'VAPOUR DETECTED':'VAP',
            }
            valor = list_description[description_incident]
            return valor
        except:
            print('CBRN (getMessageMikerField2) error:', str(traceback.format_exc()))
            return '-'
    #This function provides information about how is being the releasing of the agent
    def getMessageMikerField3(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                status_release='CONTINUOUS'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                status_release='CONTINUOUS'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
```

```
                status_release='SINGLE RELEASE OF A CLOUD'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                status_release='CONTINUOUS'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                status_release='-'

            list_status = {
                'CONTINUOUS' : 'CONT',
                'SINGLE RELEASE OF A CLOUD' : 'PUFF',
                'SPRAYING' : 'SPRAY',
            }
            valor = list_status[status_release]
            return valor
        except:
            print('CBRN (getMessageMikerField3) error:', str(traceback.format_exc()))
            return '-'
    ################################### TANGO ###################################
    # This function provides the tango module of the CBRN message
    def getMessageTango(self):
        try:
            return  'TANGO'+ self.cbrn_field_separator  + self.getMessageTangoField1(
) + self.cbrn_field_separator + self.getMessageTangoField2() + self.cbrn_module_separ
ator
        except:
            print('CBRN (getMessageTango) error:', str(traceback.format_exc()))
            return None
    #This function provides the nature of the scenario where the incident is taking p
lace
    def getMessageTangoField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                terrain_topography = 'SEA'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                terrain_topography = 'SEA'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                terrain_topography = 'FLAT'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
```

```
                terrain_topography = 'VALLEY'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                terrain_topography = 'VALLEY'

            list_terrain = {
                'FLAT':'FLAT',
                'HILL':'HILL',
                'NOT KNOWN':'NKN',
                'SEA':'SEA',
                'URBAN':'URBAN',
                'VALLEY':'VALLEY',
            }
            valor = list_terrain[terrain_topography]
            return valor
        except:
            print('CBRN (getMessageTangoField1) error:', str(traceback.format_exc()))
            return '-'
    #This function provides information about the vegatation
    def getMessageTangoField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                vegetation_description = 'BARE OF VEGETATION'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                vegetation_description = 'BARE OF VEGETATION'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                vegetation_description = 'SCRUBBY VEGETATION'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                vegetation_description = 'WOODED TERRAIN'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                vegetation_description = 'WOODED TERRAIN'

            list_vegetation = {
                'BARE OF VEGETATION':'BARE',
                'NOT KNOWN':'NKN',
                'OTHER':'OTHER',
                'SCRUBBY VEGETATION':'SCRUB',
                'URBAN':'URBAN',
                'WOODED TERRAIN':'WOODS',
```

```
            }
            valor = list_vegetation[vegetation_description]
            return valor
        except:
            print('CBRN (getMessageTangoField2) error:', str(traceback.format_exc()))
            return '-'
    ################################### YANKEE ###################################
    # This function provides the yankee module of the CBRN message
    def getMessageYankee(self):
        try:
            return  'YANKEE'+ self.cbrn_field_separator  + self.getMessageYankeeField
1() + self.cbrn_field_separator + self.getMessageYankeeField2() + self.cbrn_module_se
parator
        except:
            print('CBRN (getMessageYankee) error:', str(traceback.format_exc()))
            return None
    #This function provides the direction of downwind (where it blows to)
    def getMessageYankeeField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                downwind_direction = 3000
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                downwind_direction = 1000
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                downwind_direction = 6200
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                downwind_direction = 3100
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                downwind_direction = 2700

            return str(downwind_direction)+'MLG'
        except:
            print('CBRN (getMessageYankeeField1) error:', str(traceback.format_exc())
)
            return '-'
    #This function provides the speed of the wind referenced upper
    def getMessageYankeeField2(self):
        try:
```

```
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                downwind_speed = 10
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                downwind_speed = 16
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                downwind_speed = 6
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                downwind_speed = 4
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                downwind_speed = 12

            return str(downwind_speed)+'KPH'
        except:
            print('CBRN (getMessageYankeeField2) error:', str(traceback.format_exc())
)
            return '-'
    ################################### ZULU ###################################
    # This function provides the golf module of the CBRN message
    def getMessageZulu(self):
        try:
            return  'ZULU'+ self.cbrn_field_separator + self.getMessageZuluField1() +
 self.cbrn_field_separator + self.getMessageZuluField2()  + self.cbrn_field_separator
 + self.getMessageZuluField3()  + self.cbrn_field_separator + self.getMessageZuluFiel
d4()  + self.cbrn_field_separator + self.getMessageZuluField5() + self.cbrn_module_se
parator
        except:
            print('CBRN (getMessageZulu) error:', str(traceback.format_exc()))
            return None
    #This function provides the stability of the wind referenced upper
    def getMessageZuluField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                air_stability = 'UNSTABLE'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                air_stability = 'STABLE'
```

```
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                air_stability = 'STABLE'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                air_stability = 'NEUTRAL'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                air_stability = 'STABLE'

            list_stability={ #simplified list which is usually enough for any message
                'NEUTRAL' : 'N',
                'STABLE' : 'S',
                'UNSTABLE' : 'U',
            }
            valor = list_stability[air_stability]
            return valor
        except:
            print('CBRN (getMessageZuluField1) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the temperature of surface air
    def getMessageZuluField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                air_temperature = 10  #surface air temperature measured in celsius
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                air_temperature = 24  #surface air temperature measured in celsius
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                air_temperature = 30  #surface air temperature measured in celsius
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                air_temperature = 10  #surface air temperature measured in celsius
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                air_temperature = 16  #surface air temperature measured in celsius

            return str(air_temperature)+ 'C'
        except:
```

```
            print('CBRN (getMessageZuluField2) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the relative humidity in percentage without symbol
    def getMessageZuluField3(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                relative_humidity = 70
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                relative_humidity = 68
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                relative_humidity = 5
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                relative_humidity = 10
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                relative_humidity = 5

            if relative_humidity >0 and relative_humidity<100:
                return str(relative_humidity)
        except:
            print('CBRN (getMessageZuluField3) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the most significant weather in the scenario
    def getMessageZuluField4(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                weather_phenomena = 'RAIN'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                weather_phenomena = 'SEA BREEZE'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                weather_phenomena = 'NO SIGNIFICANT WEATHER PHENOMENA'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                weather_phenomena = 'SEA BREEZE'
```

```
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                weather_phenomena = '-'

            list_phenomena={
                'NO SIGNIFICANT WEATHER PHENOMENA':'0',
                'SEA BREEZE':'1',
                'LAND BREEZE':'2',
                'DUST/SNOW/SAND':'3',
                'FOG/HAZE':'4',
                'DRIZZLE':'5',
                'RAIN':'6',
                'SNOW':'7',
                'SHOWERS':'8',
                'THUNDERSTORMS':'9',
                'TOP OF INVERSION LAYER LOWER THAN 800 METRES':'A',
                'TOP OF INVERSION LAYER LOWER THAN 400 METRES':'B',
                'TOP OF INVERSION LAYER LOWER THAN 200 METRES':'C',
            }
            valor = list_phenomena[weather_phenomena]
            return valor
        except:
            print('CBRN (getMessageZuluField4) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the cloud coverage of the sky
    def getMessageZuluField5(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                cloud_coverage = 'COMPLETELY COVERED (OVERCAST)'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                cloud_coverage = 'NO CLOUD (CLEAR CONDITIONS)'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                cloud_coverage = 'NO CLOUD (CLEAR CONDITIONS)'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                cloud_coverage = 'NO CLOUD (CLEAR CONDITIONS)'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                cloud_coverage = 'MORE THAN HALF COVERED (BROKEN)'
```

```
            list_coverage={
                'LESS THAN HALF COVERED (SCATTERED)': '0',
                'MORE THAN HALF COVERED (BROKEN)' :'1',
                'COMPLETELY COVERED (OVERCAST)' :'2',
                'NO CLOUD (CLEAR CONDITIONS)' :'3',
            }
            valor = list_coverage[cloud_coverage]
            return valor
        except:
            print('CBRN (getMessageZuluField5) error:', str(traceback.format_exc()))
            return '-'
    ################################### GENTEXT ###################################
    # This function provides the gentext module of the CBRN message
    def getMessageGentext(self):
        try:
            return  'GENTEXT'+ self.cbrn_field_separator  + self.getMessageGentextFie
ld1() + self.cbrn_field_separator + self.getMessageGentextField2() + self.cbrn_module
_separator
        except:
            print('CBRN (getMessageGentext) error:', str(traceback.format_exc()))
            return None
    #This function provides the assesments from the commander
    def getMessageGentextField1(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                assesments = '-'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                assesments = '-'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                assesments = '-'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                assesments = '-'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                assesments = '-'

            return assesments
        except:
            print('CBRN (getMessageZuluField5) error:', str(traceback.format_exc()))
            return '-'
    #This function provides the free text from the commander
```

```
    def getMessageGentextField2(self):
        try:
            if self.example == 0: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ENM ######
                free_text = 'EL AGENTE DISPERSADO HA SIDO IDENTIFICADO COMO GAS MOSTA
ZA LANZADO DESDE UN BUQUE EN TRÁNSITO. TODO EL PERSONAL DE LA ENM ESTÁ SIENDO ALISTAD
O CON EPI'
            elif self.example == 1: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### ESTRECHO ######
                free_text = 'INCIDENTE EN EL REACTOR DE UN SUMBARINO EN TRÁNSITO. ORD
EN DE ABANDONAR EL DST A TODOS LOS BARCOS'
            elif self.example == 2: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CISA ######
                free_text = 'INCIDENTE EN LABORATORIO DE ALTA SEGURIDAD BIOLOGICA. EV
ACUACIÓN DE PERSONAL EN LOCALIDADES A MENOS DE 30KM.'
            elif self.example == 3: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### CENTRAL NUCLEAR ######
                free_text = 'INCIDENTE EN EL REACTOR DE LA CENTRAL NUCLEAR DE COFRENT
ES. PROBABILIDAD DE VERTIDO DE AGUA TÓXICA EN EL RIO. ESTABLECER CONDICIONES DE SEGUR
IDAD RIO ABAJO'
            elif self.example == 4: #0-> ENM; 1-> Estrecho; 2->CISA; 3-
>Central nuclear; 4-> Ejercicio
                ###### EJERCICIO ######
                free_text = 'EL AGENTE DISPERSADO HA SIDO IDENTIFICADO COMO GAS MOSTA
ZA LANZADO DESDE UN BUQUE EN TRÁNSITO. TODO EL PERSONAL DE LA ENM ESTÁ SIENDO ALISTAD
O CON EPI'

            return free_text
        except:
            print('CBRN (getMessageGentextField2) error:', str(traceback.format_exc()
))
            return '-'
    ################################### CLIENT ###################################
    #The set client extracts info a data from the mqtt message about the scenario
    #This function splits the message into modules and fiels so they can be analized
    def extractCBRNMessageInformation(self, mqtt_message):
        try:
            module_complete_lines = mqtt_message.split(self.cbrn_module_separator) #S
plitted in modules
            for module_complete_line in module_complete_lines: #Going over every sing
le module
                field_data = module_complete_line.split(self.cbrn_field_separator) #S
plitting modules into fields
                if field_data [0] == 'ALFA':                                    #I
dentifying which module it is by the heading field!!!!!!!
```

```
                    extract_message_alfa = [int(field_data[3]),field_data[4]]
                elif field_data [0] == 'BRAVO':
                    extract_message_bravo = self.extractLatLongFromText(field_data[1]
)
                elif field_data [0] == 'DELTA':
                    extract_message_delta = self.extractMessageDelta(field_data[1])
                elif field_data[0] == 'FOXTROT' :
                    extract_message_foxtrot = self.extractMessageFoxtrot (field_data)
                elif field_data[0] == 'YANKEE':
                    extract_message_yankee = self.extractMessageYankee(field_data)
            information_extracted=[extract_message_alfa, extract_message_bravo, extra
ct_message_delta, extract_message_foxtrot, extract_message_yankee[0], extract_message
_yankee[1]]
            return information_extracted

        except:
            print('CBRN (extractCBRNMessageInformation) error:', str(traceback.format
_exc()))
            return None
    #This function extracts the information for the Delta module sent
    def extractMessageDelta(self, original_text):
        try:
            day = original_text[:2]
            dtg_without_day = original_text[2:]
            time = dtg_without_day[:4]
            dtg_without_day_time = original_text[7:]
            month= dtg_without_day_time[:3]
            number_month=self.reverseDictionary(self.month_list)[month]
            year = original_text[10:]
            return time, day, number_month, year
        except:
            print('CBRN (extractMessageDelta) error:', str(traceback.format_exc()))
            return None
    #This function extracts the information for the Foxtrot module sent
    def extractMessageFoxtrot (self, original_text):
        try:
            source_location = self.extractLatLongFromText (original_text[1])
            return source_location
        except:
            print('CBRN (extractMessageFoxtrot) error:', str(traceback.format_exc()))
            return None
    #This function extracts the information for the Yankee module sent
    def extractMessageYankee (self, original_text):
        try:
            downwind_direction_extracted=''
            downwind_speed_extracted=''
            character_list = list (original_text[1])
            for character in character_list:
                if character.isdigit():
                    downwind_direction_extracted+=character
```

```
                else:
                    break
            downwind_direction_extracted = int (downwind_direction_extracted)

            character_list = list (original_text[2])
            for character in character_list:
                if character.isdigit():
                    downwind_speed_extracted+=character
                else:
                    break
            downwind_speed_extracted = int (downwind_speed_extracted)

            return downwind_direction_extracted, downwind_speed_extracted
        except:
            print('CBRN (extractMessageFoxtrot) error:', str(traceback.format_exc()))
            return None

    #This function turns coordinates in text format into a pair of integer coordinate
s with sign
    def extractLatLongFromText(self, original_text):
        try:
            character_list = list(original_text)
            for character in character_list : #Checking the sing of the coordinate
                if character == 'E':
                    longitude_sign = True
                elif character == 'W':
                    longitude_sign = False
                if character == 'N':
                    latitude_sign = True
                elif character == 'S':
                    latitude_sign = False
            longitude_abs = split('E|W|N|S', original_text)[1]
            latitude_abs = split('N|S', original_text)[0]
            if longitude_sign :
                longitude = float(longitude_abs)
            if not longitude_sign:
                longitude = -float(longitude_abs)
            if latitude_sign:
                latitude = float(latitude_abs)
            if not latitude_sign:
                latitude = -float(latitude_abs)
            return latitude, longitude #returns 2 floats
        except:
            print('CBRN (extractLatLongFromText) error:', str(traceback.format_exc())
)
            return None
```

```
    ################################### COMUNICATIONS ##############################
####

    def send(self, scenario_source_data, scenario_sensor_data):
        try:

            message_header = self.getMessageHeader(scenario_sensor_data)
            message_module_alfa = self.getMessageAlfa(scenario_sensor_data)
            message_module_bravo = self.getMessageBravo(scenario_source_data, scenari
o_sensor_data)
            message_module_delta = self.getMessageDelta()
            message_module_foxtrot = self.getMessageFoxtrot(scenario_source_data)
            message_module_golf = self.getMessageGolf()
            message_module_india = self.getMessageIndia()
            message_module_miker = self.getMessageMiker()
            message_module_tango = self.getMessageTango()
            message_module_yankee = self.getMessageYankee()
            message_module_zulu = self.getMessageZulu()
            message_module_gentext = self.getMessageGentext()

            mqtt_message = message_header + message_module_alfa + message_module_brav
o + message_module_delta + message_module_foxtrot
            mqtt_message = mqtt_message + message_module_golf + message_module_india
+ message_module_miker + message_module_tango
            mqtt_message = mqtt_message + message_module_yankee + message_module_zulu
 + message_module_gentext
            print('mqtt_message=', mqtt_message)
            print(self.extractCBRNMessageInformation(mqtt_message))
        except:
            print('CBRN (send) error:', str(traceback.format_exc()))
            return

    def connect(self, mqtt_broker, mqtt_port, mqtt_topics):
        try:
            return True
        except:
            print('CBRN (connect) error:', str(traceback.format_exc()))
            return
```