Proyecto de SI

SI-31

Miguel Ángel Seara Losada
David Simón Nóvoa
Mauro Zelenka Pedrosa
Luis Fernando Pérez Moure

Índice de Contenidos

1. Introducción

El presente documento describe el desarrollo de un sistema de asistencia domiciliaria basado en agentes inteligentes implementados en la plataforma Jason, cuyo objetivo es simular la interacción entre un robot asistencial (enfermera), un usuario o propietario (owner) y un auxiliar a robot asistencial (auxiliar) dentro de un entorno doméstico.

El objetivo será la suministración de los medicamentos por parte del robot enfermera al owner, cumpliendo las pautas obtenidas. Asimismo, el robot auxiliar se encargará de controlar el stock de medicamentos, obteniendo más suministros de medicinas en caso necesario.

Para eso, los agentes deben poder moverse libremente, sorteando obstáculos y atravesando puertas para moverse entre habitaciones, evitando colisionar entre ellos, con paredes u obstáculos del entorno. Esto se consigue gracias a la implementación de un algoritmo de pathfinding, que en nuestro caso es el algoritmo A*.

Asimismo, como autómatas que operan en un entorno persistente, tanto la agente enfermera como el auxiliar gestionan un sistema de energía. Dicho sistema controla el consumo energético derivado de sus acciones y les obliga a realizar ciclos de recarga para evitar quedarse sin batería durante la ejecución de sus tareas.

A lo largo de este documento, se detalla el diseño, la arquitectura y los mecanismos que dan solución a estos desafíos, ofreciendo una visión integral de un sistema multiagente diseñado para la asistencia real y fiable.

2. Interfaz Gráfica y Visualización

2.1 Representación de los Elementos Gráficos del Entorno

El trabajo incluye una interfaz gráfica para la monitorización de la simulación. Dicha interfaz gráfica ofrece una visión del entorno domótico de una vivienda, que está organizada en habitaciones, todas ellas separadas por paredes y con puertas que interconectan las habitaciones como se puede observar en la Imagen 1. Entre estos, podemos encontrar:

- Una entrada, donde podemos localizar la puerta de entrada de la casa.
- Una cocina, que contiene un frigorífico para las cervezas y el almacén de medicamentos. La interfaz gráfica muestra junto a estos elementos la cantidad total de unidades disponibles de cada uno, permitiendo una supervisión constante del inventario de cada objeto.
- Una sala, donde podemos encontrar elementos interactivos para el agente owner. Dichos objetos son 4 sillones y un sofá donde el propio owner podrá sentarse a descansar. También encontraremos una mesa.
- Un pasillo, en el cual al final de todo encontramos un cargador para que los robots enfermera y auxiliar puedan cargarse.
- Tres habitaciones, todas ellas disponen de una cama, siendo pequeña en dos de ellas y grande en solo una habitación. Dichas camas pueden tener interacción con el owner.
- Y dos baños, donde no encontramos ningún objeto a excepción de las puertas de conexión dentro de estos.

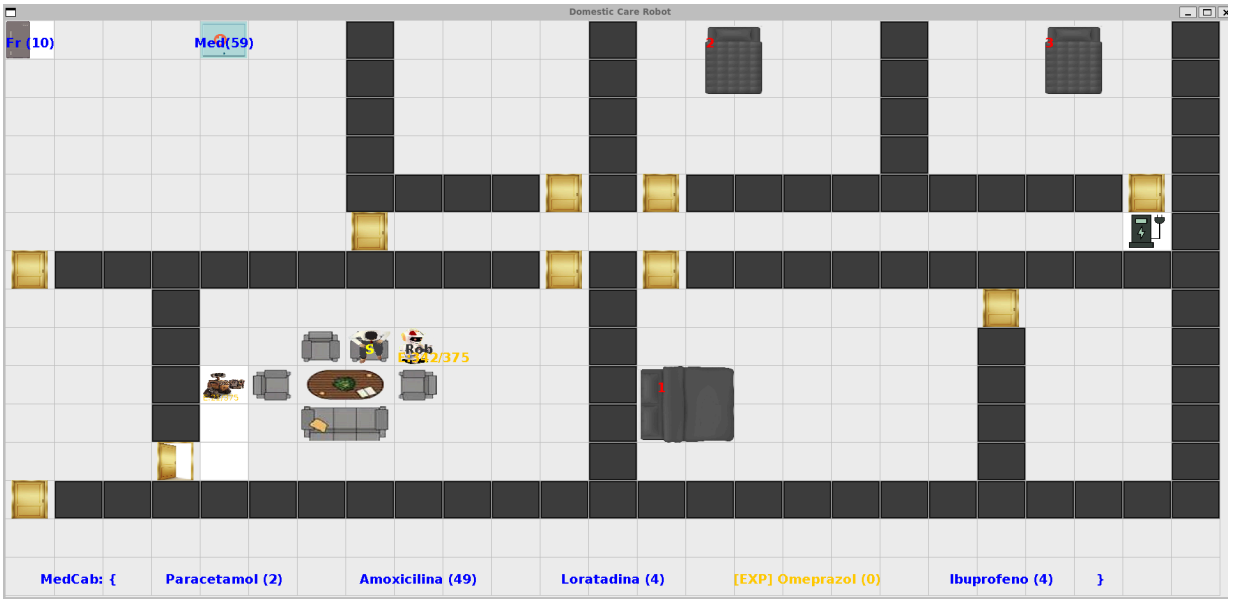

Imagen 1: Mapa del entorno doméstico

Además de lo anterior comentado, hay que resaltar la ubicación de objetos de especial relevancia que serán importantes en la ejecución del entorno:

- **Almacén de medicinas**: Donde se guardan las medicinas. Se encuentra en las coordenadas (4, 0).
- **Punto de carga**: Dispone de un único cargador no compartido, y será el lugar donde enfermera y auxiliar recarguen energía. Está ubicado en la celda (23, 5).

Además de eso, en la Imagen 1 podemos observar 3 agentes en el entorno que son el robot, el owner y el auxiliar, donde se representan con figuras distintas cuyos iconos cambian dinámicamente para reflejar su estado y acciones:

- El icono del owner se actualiza para mostrar su dirección de movimiento o si está realizando una acción específica como sentarse.
- Los iconos del robot y del auxiliar cambian para mostrar visualmente el objeto que transportan. Por ejemplo, el robot tiene un icono diferente si lleva una cerveza o un medicamento, y el auxiliar también cambia su apariencia si está llevando una caja de medicinas.
- Adicionalmente, debajo de los iconos del robot y del auxiliar, se muestra un texto que indica su nivel de energía actual, permitiendo una supervisión rápida de su autonomía sin necesidad de consultar el panel de estado de energía.

Inicialmente, el agente owner comenzará en la posición (13, 9), el robot enfermera en la posición (19, 3) y el robot auxiliar en la posición (1, 8).

Por último, el owner podrá interactuar con objetos como las camas, los sillones, el sofá, el frigorífico y el cajón de medicamentos. El robot enfermera puede interactuar con el propio owner, el cajón de medicamentos y el cargador.

2.2 Paneles de Información

Además de la vista principal que modela el entorno, la interfaz se ha enriquecido con dos componentes flotantes que proporcionan información crucial en tiempo real:

1. **Panel de Reloj Simulado**: Una ventana independiente, que muestra el paso del tiempo dentro de la simulación (HH:MM). Este reloj es fundamental, ya que su avance rige la activación de eventos basados en horarios, como las pautas de medicación, y permite la monitorización de eventos que deben suceder en un tiempo determinado.

   Este reloj funciona de manera acelerada de modo que cada 30 segundos que pasan en el mundo real, transcurre una hora completa dentro de la simulación. En la Imagen 2 podemos observar la pestaña del reloj ya comentado.

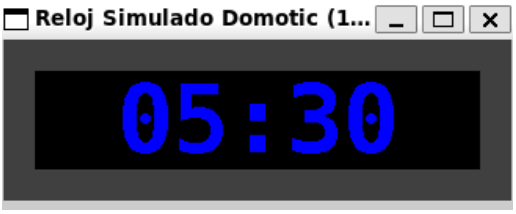

Imagen 2: Reloj Simulado

2. **Panel de Estado de Energía**: También disponemos de una pestaña gestionada por barras de progreso, donde se muestran los niveles de batería actuales del robot enfermera y del auxiliar.

   Gracias a este panel, es posible monitorizar el estado de la batería de los autómatas en tiempo real. La barra de progreso disminuirá con cada acción, y el decremento variará según la complejidad de la tarea, o se incrementará progresivamente si el agente está recargando energía en el cargador. La Imagen 3 ilustra esta funcionalidad.

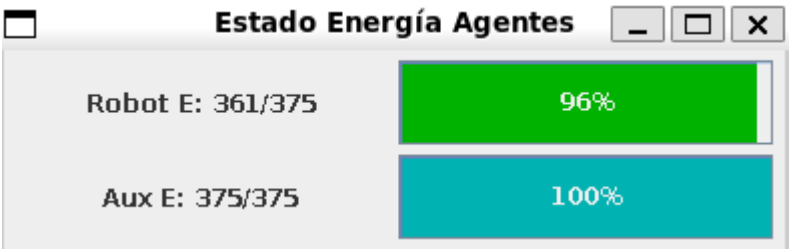

Imagen 3: Estado energía agentes robots

2.3 Visualización Dinámica del Inventario de Medicamentos

Una de las funcionalidades visuales más relevantes es la representación del inventario del botiquín. Para ofrecer una visibilidad clara y en tiempo real del stock, se amplió el espacio visual del mapa, añadiendo filas adicionales en la parte inferior de la ventana. Este espacio extra permite implementar un panel de texto dinámico como el ilustrado en la siguiente imagen.

Imagen 4: Estado del stock del cajón de medicamentos

Como se muestra en la imagen, este panel presenta la siguiente información:

- **Nombre del Medicamento**: Se lista cada uno de los medicamentos disponibles en el sistema.
- **Cantidad Disponible**: Junto a cada nombre, un número entre paréntesis indica la cantidad de unidades restantes de este medicamento. Por ejemplo, Paracetamol (2) indica que quedan 2 dosis.

- **Estado de Caducidad**: Si un medicamento ha superado su fecha de caducidad, el sistema añade el prefijo [EXP] y cambia el color del texto a naranja para alertar visualmente de que ese medicamento ya no es apto para el consumo.

Esta funcionalidad se gestiona en el Modelado de Vista. Cada vez que se redibuja el entorno, el sistema consulta cuantos medicamentos quedan y cuando expiran para generar y mostrar esta lista actualizada. Esto asegura que la información visual sea siempre un reflejo fiel del estado lógico del inventario.

Para finalizar, se muestra una visualización de una ejecución del proyecto, donde podemos observar que se muestran el mapa del entorno con sus respectivos objetos, y las pestañas con información adicional ya explicadas anteriormente.

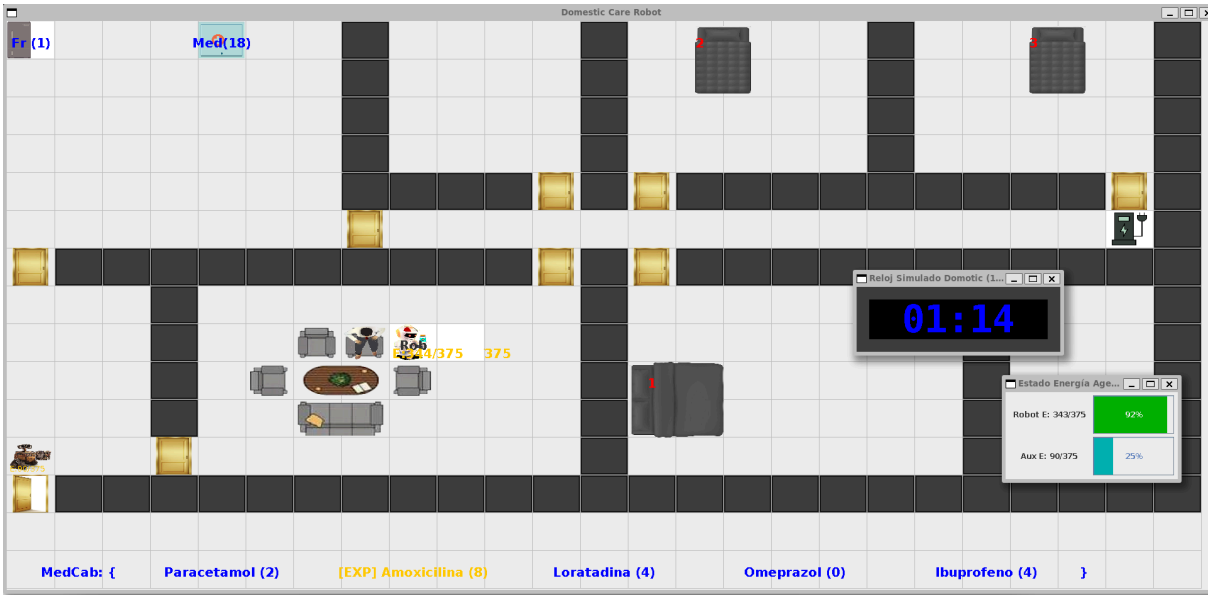

Imagen 5: Simulación de entorno

3. Mecanismos Clave de la Simulación

Esta sección detalla los sistemas fundamentales que gobiernan el comportamiento de los agentes y la lógica del entorno. Estos mecanismos son cruciales para garantizar una simulación robusta, realista y coherente.

3.1 Navegación, Colisiones y Gestión de Bloqueos

Para que los agentes se desplacen de forma autónoma y eficiente por la vivienda, se ha implementado un sistema de navegación avanzado que combina el algoritmo de búsqueda A* con una lógica de resolución de conflictos. El objetivo es garantizar un movimiento fluido y evitar situaciones de bloqueo permanente.

El sistema se basa en el algoritmo **A***, que calcula la ruta más corta entre dos puntos usando la **distancia Manhattan** como heurística, garantizando así trayectorias óptimas. El algoritmo explora las celdas adyacentes y consulta constantemente para descartar cualquier celda que contenga un obstáculo estático, como paredes o muebles.

Para la **gestión de bloqueos entre agentes**, se aplica una estrategia jerárquica:

1. **Intento de Empuje**: El agente activo empuja al otro hacia la celda de atrás, solo si la celda de empuje es válida.
2. **Maniobra Lateral Evasiva**: Si no puede empujar, el agente busca una celda adyacente desocupada a la que el agente pueda moverse temporalmente para ceder el paso, evitando así el estancamiento.
3. **Espera táctica**: Como último recurso, si un agente no puede avanzar ni resolver el bloqueo, activa un mecanismo de espera. El agente se detiene durante un número aleatorio de ciclos y, pasado ese tiempo, vuelve a calcular la ruta para reintentar el movimiento.

Esta jerarquía de decisiones se adapta de manera inteligente al contexto del encuentro:

- **Caso 1: Encuentro en una puerta.** Cuando dos agentes se encuentran en una puerta, el sistema analiza el espacio real para tomar la decisión más eficiente en lugar de seguir una regla fija. Primero, el agente evalúa si puede empuja**r** al otro a una celda válida detrás de él. Si esto no es viable, intenta realizar una maniobra lateral si la anchura de la puerta lo permite. Solo si ambas acciones físicas fallan, el agente recurre a una espera táctica temporal, cediendo el paso hasta que el camino se despeje.
- **Caso 2: Encuentro en un pasillo estrecho.** Este escenario es similar al de una puerta. Al no haber espacio a los lados, la Maniobra Lateral no es una opción. Si ambos agentes se encuentran de frente, es probable que el Intento de Empuje también falle. El sistema resuelve el punto muerto mediante la Espera Táctica. Si ambos agentes activan la espera, la aleatoriedad en la duración de la pausa asegura que uno de ellos se reactivará antes que el otro, tomando la iniciativa y despejando el camino.
- **Caso 3: Encuentro en un espacio abierto.** En zonas amplias como el salón, la estrategia más eficiente es la Maniobra Lateral Evasiva. Cuando un agente

encuentra su ruta óptima bloqueada, en lugar de esperar, se desvía a una celda adyacente libre y recalcula inmediatamente su ruta desde esa nueva posición. Esto permite un desvío fluido y casi instantáneo, manteniendo el movimiento constante y evitando paradas innecesarias.

Este sistema multifacético asegura que los agentes puedan navegar por toda la casa de manera fluida y resolver conflictos de forma autónoma, evitando situaciones de bloqueo permanente.

3.2 Gestión de Energía y Batería

Los agentes enfermera y auxiliar son autómatas con recursos limitados, por lo que su funcionamiento depende de un sistema de energía que deben gestionar activamente. Podemos destacar las siguientes funcionalidades:

- **Coste Energético de las Acciones**: Cada acción realizada por los agentes tiene un coste energético asociado, algunas acciones consumen más energía que otras.
- **Ciclo de Recarga**: Cuando el nivel de energía de un agente cae por debajo de un umbral predefinido, este adoptará el objetivo de recargarse. Para ello, deberá desplazarse hasta el único punto de carga y permanecer allí hasta que su batería se llene. Dado que el cargador es un recurso compartido, si un agente lo encuentra ocupado, deberá esperar su turno. Un detalle importante del sistema es que los tiempos de recarga son diferentes para cada agente: el robot enfermera necesita 60 minutos de tiempo simulado para una carga completa, mientras que el auxiliar se recarga en solo 30 minutos.
- **Penalización por Carga Parcial**: Si un agente interrumpe la carga más de tres veces, su capacidad máxima de energía se reduce en un 10%. Aunque la lógica está implementada, su efecto en la barra de energía máxima no se refleja visualmente.
- **Transferencia de Energía entre Agentes**: Para situaciones críticas, el auxiliar puede actuar como una batería de emergencia para el robot. Si el robot se está quedando sin energía, puede solicitar ayuda y el auxiliar, se desplazará hasta él y le transferirá una parte de su propia energía, asegurando que el robot pueda continuar con sus tareas prioritarias. Aunque esta lógica está implementada en el código cabe señalar que, en una revisión posterior, se detectó que su efecto no se ve reflejado.
- **Previsión de Gasto Energético (Funcionalidad No Implementada)**: Planteamos cómo realizar un mecanismo avanzado de previsión de energía que, por falta de tiempo, no se llegó a implementar. Antes de iniciar una tarea, el agente calcularía la energía total requerida. Este cálculo no solo incluiría el coste de la acción en sí, sino también la energía necesaria para el desplazamiento hasta el objetivo y, de forma crucial, el coste del viaje de vuelta hasta el cargador. El agente compararía este coste total con su energía disponible, y si la energía que gasta es mayor de la disponible, la tarea no se realiza, y el agente adoptaría como objetivo prioritario ir a recargar su batería.

3.3 Gestión de Medicación: Stock y Caducidad

El sistema implementa una gestión dual para el inventario de medicamentos, diferenciando entre la reposición por caducidad y la reposición por falta de stock, para garantizar tanto la seguridad como la disponibilidad continua.

- **Control de la Caducidad**: Cada medicamento tiene asignada una hora y minuto de caducidad específicos. El agente auxiliar es el principal responsable de esta tarea, monitorizando constantemente el reloj de la simulación. El sistema no actúa en la hora exacta en que caduca el medicamento; en su lugar, el auxiliar detecta que un medicamento ya ha caducado y, solo entonces, inicia proactivamente el objetivo de reposición. Este proceso implica recoger las nuevas unidades en el punto de entrega y gestionar su reabastecimiento en el almacén.
- **Proceso de Reposición Cantidad**: Para asegurar que los agentes nunca se encuentren con un medicamento completamente agotado, el sistema implementa que cada agente cuando detecta que las existencias son cero, su propia acción de solicitar el medicamento provoca la reposición de nuevas unidades, garantizando así que pueda completar su tarea con éxito.

3.4 Proceso de Obtención y Verificación de Medicamentos

El sistema gestiona la obtención de medicamentos a través de un mecanismo de pautas horarias que desencadena diferentes flujos de acción, dependiendo del estado y el comportamiento de los agentes.

Cuando llega la hora de una pauta, se activa de forma aleatoria uno de los siguientes tres escenarios para la obtención de la dosis:

- **Iniciativa Exclusiva del Owner**: El owner decide ir por su cuenta a la medicación. Se desplaza hasta el botiquín, recoge la dosis y notifica al robot de su acción. Este escenario no termina ahí, sino que desencadena un importante mecanismo de verificación, que explicaremos más adelante.
- **Entrega estándar del robot respaldada por el auxiliar:** Si el owner olvida su dosis, el robot enfermera delega la tarea de recogida en el auxiliar, quien irá al botiquín, cogerá la medicina y se la llevará físicamente al robot. Una vez que el robot tiene la dosis en su poder, finaliza la tarea entregándosela al owner.
- **Competición entre Robot y Owner:** Tanto el owner cómo el robot inician simultáneamente el objetivo de ir a por la medicación. El resultado depende de quién llegue primero al almacén de medicación:
    - ➢ **Si gana el owner**: Recoge la dosis y notifica al robot. Cuando el robot llega, ve que el owner ya tiene el medicamento y, en lugar de entregárselo, activa su plan de verificación.
    - ➢ **Si gana el robot**: Recoge la dosis y se la entrega directamente al owner, completando la tarea de forma estándar.

Verificación de Seguridad y Cumplimiento:

- **Control de Caducidad:** Antes de recoger un medicamento, tanto el robot como el owner tienen la obligación de verificar que dicho medicamento no esté caducado.

- **Verificación de la Toma:** El robot no confía ciegamente en las notificaciones del owner. Si este le informa de que ha tomado un medicamento, el robot se desplaza al estante de medicamentos para comprobar físicamente si el stock ha disminuido. Si no es así, le lleva la dosis personalmente. Es importante señalar que esta lógica implementada presenta un conflicto conocido con la reposición automática, que puede llevar a conclusiones erróneas del robot y entregas innecesarias.
- **Límite de Dosis Diaria (Contemplado)**: Adicionalmente, se ha contemplado en el código la posibilidad de implementar un límite en la cantidad de dosis diarias que el agente owner puede tomar de un mismo medicamento. Aunque esta funcionalidad se encuentra actualmente comentada y no está activa en la simulación final, su diseño está previsto para prevenir una posible sobredosificación.

4. Modelado y Diseño del Sistema

Tras haber analizado la interfaz y los mecanismos clave que rigen el comportamiento de los agentes, esta sección se centra en el diseño formal del mismo. Para ello, se van a utilizar a lo largo de la sección una serie de diagramas que ilustran la arquitectura, la organización de los agentes y el flujo de interacciones, con el objetivo de facilitar las explicaciones y que sea de entendimiento lo más ameno posible.

4.1 Modelado del entorno y Organización

Lo primero que haremos será definir el mundo en el que operan los agentes y los roles que cada uno desempeña para poder entender el entorno.

El entorno define el espacio donde interactúan los agentes. La Figura 1 ilustra este modelo de interacción, especificando las percepciones que los agentes reciben y las acciones que pueden ejecutar sobre él.

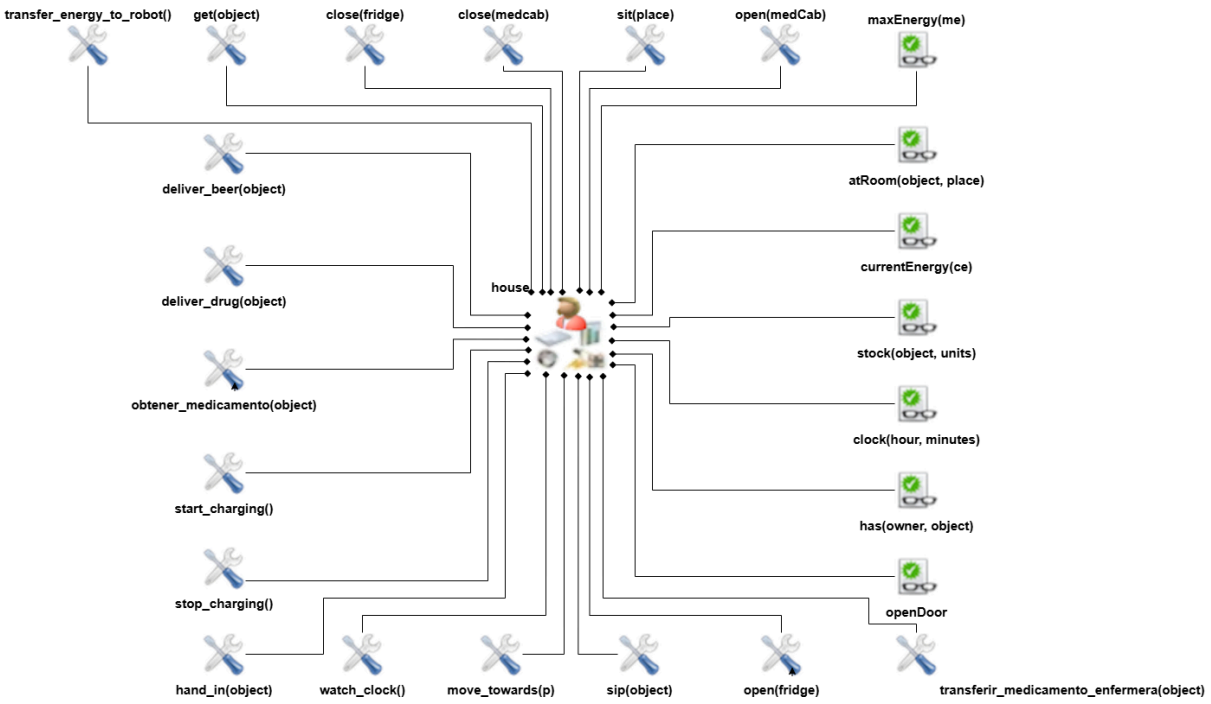

Figura 1: Diagrama de entorno

Dichos agentes, precisan del entorno, las siguientes percepciones:

- **has(owner, Object)**: Percibida por el owner, indicando que tiene o un medicamento a consumir, o bien una cerveza (no vacía).
- **stock(Object, Units)**: Percibida por robot, auxiliar y owner cuando abren la nevera, donde indican las unidades de medicamentos o cervezas que hay.
- **atRoom(Object, Place)**: Percibida por robot, auxiliar y owner, donde indican la habitación donde se encuentran los agentes y objetos de la casa.
- **openDoor**: Percibida por robot, auxiliar y owner, indicando si están abriendo una puerta.

- **clock(Hour, Minute):** Percibida por robot, auxiliar y owner, indicando la hora.
- **current_energy(CE)**: Percibida por robot y auxiliar, indicando la energía disponible que tienen.
- **max_energy(ME)**: También percibida por robot y auxiliar, para indicar el tope máximo de energía que pueden tener.

Además de estas percepciones, los agentes se comunican con el entorno a través de una serie de acciones externas:

- **sit(Place)**: El owner, se sentará en el lugar Place indicado (sillas o sofá).
- **open(medCab)**: Owner, robot y auxiliar dicen al entorno que abren el cajón de medicamentos, para que muestre el stock de medicamentos.
- **close(medCab)**: Owner, robot y auxiliar le indican al entorno que cierran el cajón de medicamentos.
- **open(fridge)**: Owner comunica al entorno que abre el frigorífico, para que muestre el stock de cervezas.
- **close(fridge)**: Owner indica al entorno que cierran el frigorífico.
- **move_towards(P)**: Los agentes indican al entorno que se están desplazando.
- **get(Object)**: Owner indica que ha cogido una cerveza o un medicamento al entorno.
- **hand_in(Object)**: Cuando owner, o bien coge una cerveza en el frigorífico, o bien robot le entrega una medicación, este la comunica al entorno antes de consumirla.
- **sip(Object)**: Owner comunica al entorno que está consumiendo/tomando el objeto que posee (cerveza o medicamento).
- **transferir_medicamento_enfermera(Object)**: Se comunica al entorno que auxiliar, una vez cogida la medicación en el cajón de medicamento, se la entrega a robot para el posterior reparto al owner.
- **watch_clock**: Owner comunica que quiere saber la hora en la que se encuentra, cuando se aburre.
- **obtener_medicamento(Drug)**: Robot comunica al entorno que quiero coger el medicamento a buscar (hay cantidades de dicho medicamento).
- **deliverdrug(Drug, Qtd)**: Auxiliar avisa al entorno que va a reponer una cantidad Qtd de un medicamento del cajón de medicamentos que trajo supermarket (ya sea caducado o agotado).
- **deliverbeer(Object, Qtd)**: Owner avisa al entorno que va a reponer una cantidad Qtd de cervezas del frigorífico que trajo supermarket (ya sea caducado o agotado).
- **cargar_medicamento**: Auxiliar comunica al entorno que está cargando un medicamento a reponer.
- **start_charging**: Robot o auxiliar comunican al entorno que va a cargar batería en el cargador (solo uno y sabiendo que no hay nadie cargando).
- **stop_charging**: Robot o auxiliar comunican al entorno que han cargado y dejan libre el cargador.

Para coordinar las tareas complejas que requieren la intervención de múltiples agentes, se ha diseñado un diagrama de organización. Este modelo visualiza los roles que asume cada agente y cómo colaboran para alcanzar un objetivo común.

La Figura 2 ilustra la organización para la tarea principal del sistema: la obtención y entrega de medicamentos al owner.

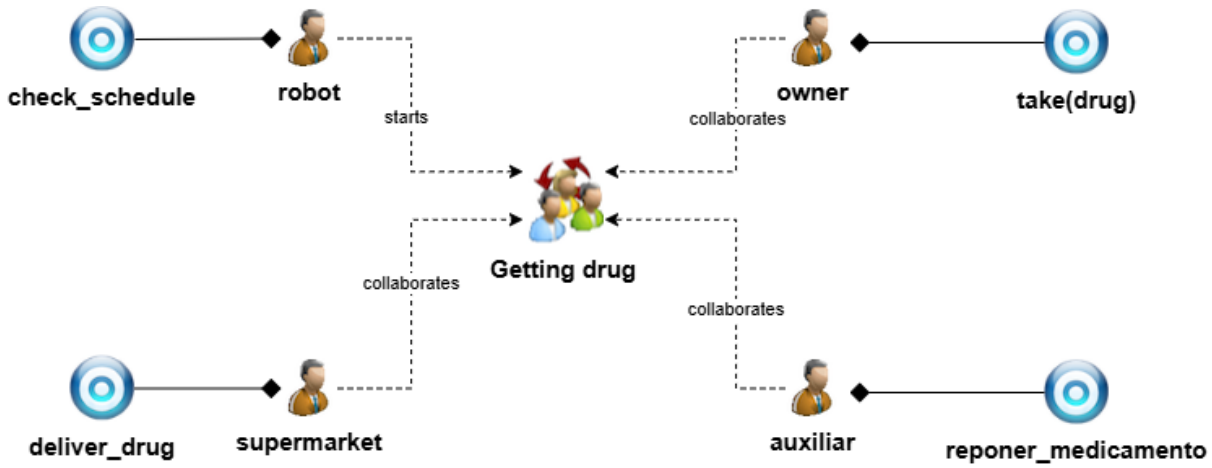

Figura 2: Diagrama de organización para robot doméstico

Como se puede observar, la interacción central, getting drug es un objetivo colaborativo. Se puede destacar lo siguiente:

- El robot enfermera inicia la tarea al detectar la hora de la medicación.
- El auxiliar colabora reponiendo el medicamento caducado o asistiendo directamente al robot enfermera.
- El agente supermarket colabora entregando nuevos medicamentos cuando se le solicita.
- El owner participa como receptor final del proceso tomando la medicación adecuada.

4.2 Modelado de objetivos e interacciones

Hemos definido el entorno donde van a trabajar nuestros agentes, y un modelado que ilustra el objetivo de que el agente owner tome la medicación. A continuación, en la Figura 3 se presenta el diagrama de objetivos que modela la secuencia ideal de tareas para completar la entrega de medicación al owner. Este diagrama descompone la meta principal en una serie de subobjetivos interconectados.

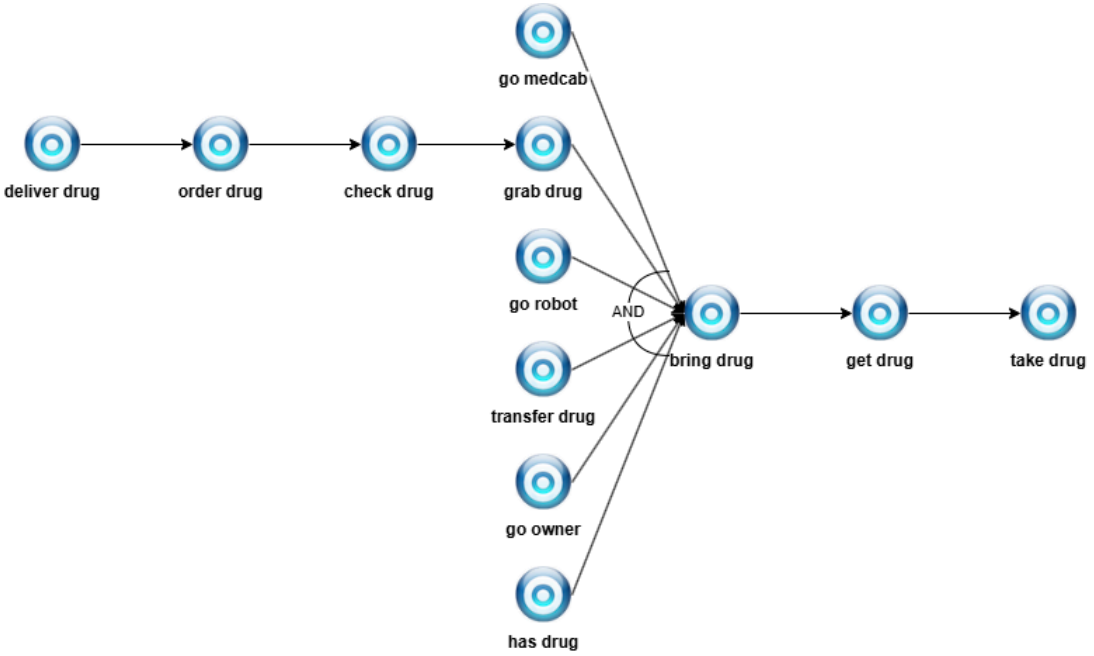

Figura 3: Diagrama de objetivos ideal para robot doméstico

El flujo de objetivos para una entrega de medicación exitosa se descompone de la siguiente manera:

1. **Objetivo Final (take drug)**: La meta principal es que el owner consuma su medicación. Para que esto ocurra, primero debe recibirla.
2. **Objetivo de Entrega (get drug)**: Para que el owner reciba la medicación, el robot debe primero llevarla hasta él.
3. **Objetivo de Transporte (bring drug)**: Este es el objetivo central del robot y requiere la consecución de una serie de tareas secuenciales:
    - Ir al almacén de medicamentos (go medcab).
    - Coger el medicamento correcto (grab drug).
    - Si es el auxiliar quien realiza la tarea, debe dárselo al robot (transfer drug).
    - Desplazarse hasta la ubicación del owner (go owner).
    - Entregarle la medicina (hand_in).
4. **Objetivo Condicional de Reposición (order drug)**: Este flujo se activa solo si, al intentar coger un medicamento (check drug), se detecta que no hay existencias. En ese caso, se activa el objetivo order drug para solicitar nuevas unidades al agente supermarket, quien las proporcionará a través del objetivo deliver drug.

Mientras que los diagramas anteriores definen los objetivos, el diagrama de interacción de la Figura 4 modela cómo los agentes se comunican y colaboran mediante el intercambio de mensajes y la activación de roles para lograr dichos objetivos.

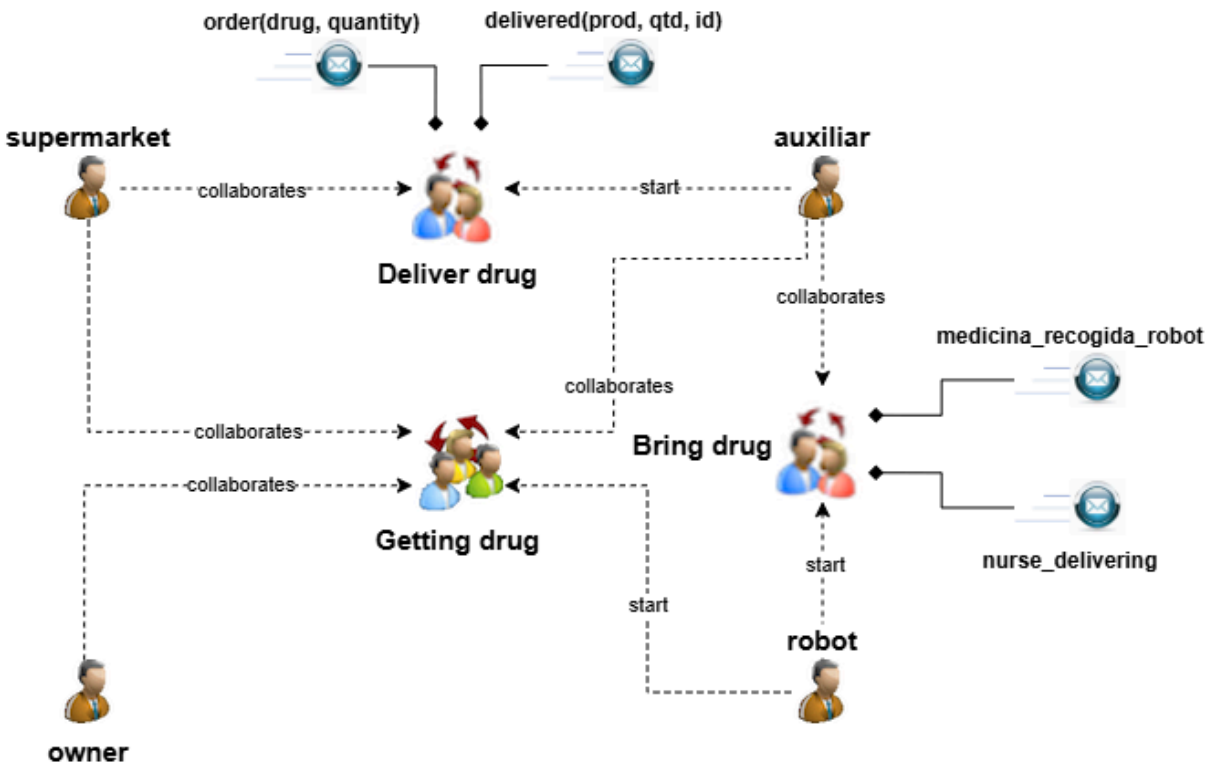

Figura 4: Diagrama de interacción para el robot doméstico

El diagrama ilustra, como se explicó en la Figura 2, la interacción principal Getting Drug, qué es iniciada por el robot enfermera, y participan los agentes owner, supermarket y auxiliar. Además, se puede visualizar dos conversaciones colaborativas entre algunos agentes para la realización de la interacción anterior:

1. **Conversación de Entrega (bring drug)**:
    - Esta interacción es iniciada por el robot enfermera cuando necesita entregar una medicación.
    - En un posible escenario, la enfermera puede delegar la tarea de recogida al auxiliar. El robot inicia la conversación y el auxiliar responde colaborando para llevarle la medicina. Una vez el auxiliar completa su parte, el robot notifica al owner que la entrega está en curso.

2. **Conversación de Reposición (deliver drug)**:
    - Esta interacción la inicia el agente auxiliar cuando detecta que ha caducado un medicamento.
    - El auxiliar se comunica con el agente supermarket enviándole un mensaje de pedido.
    - El agente supermarket colabora respondiendo con un mensaje de confirmación una vez que el producto ha sido entregado en el punto de recogida.

4.3 Diseño Detallado de los Agentes

A continuación, se presenta un análisis detallado del comportamiento de cada uno de los agentes, junto a los planes, objetivos y creencias de cada uno.

4.3.1 Owner

El agente owner representa al propietario de la vivienda, simulando tanto un comportamiento humano con rutinas diarias como la necesidad de seguir una pauta médica. Puede actuar de forma proactiva y tomar su medicación, pero también puede olvidarla, requiriendo la asistencia del robot. Cuando la toma por su cuenta, debe notificarlo para que el robot verifique la acción.

Las Figuras 5, 6a y 6b representan el diagrama de agente de owner y sus diagramas de tareas, que se han separado para mayor claridad.

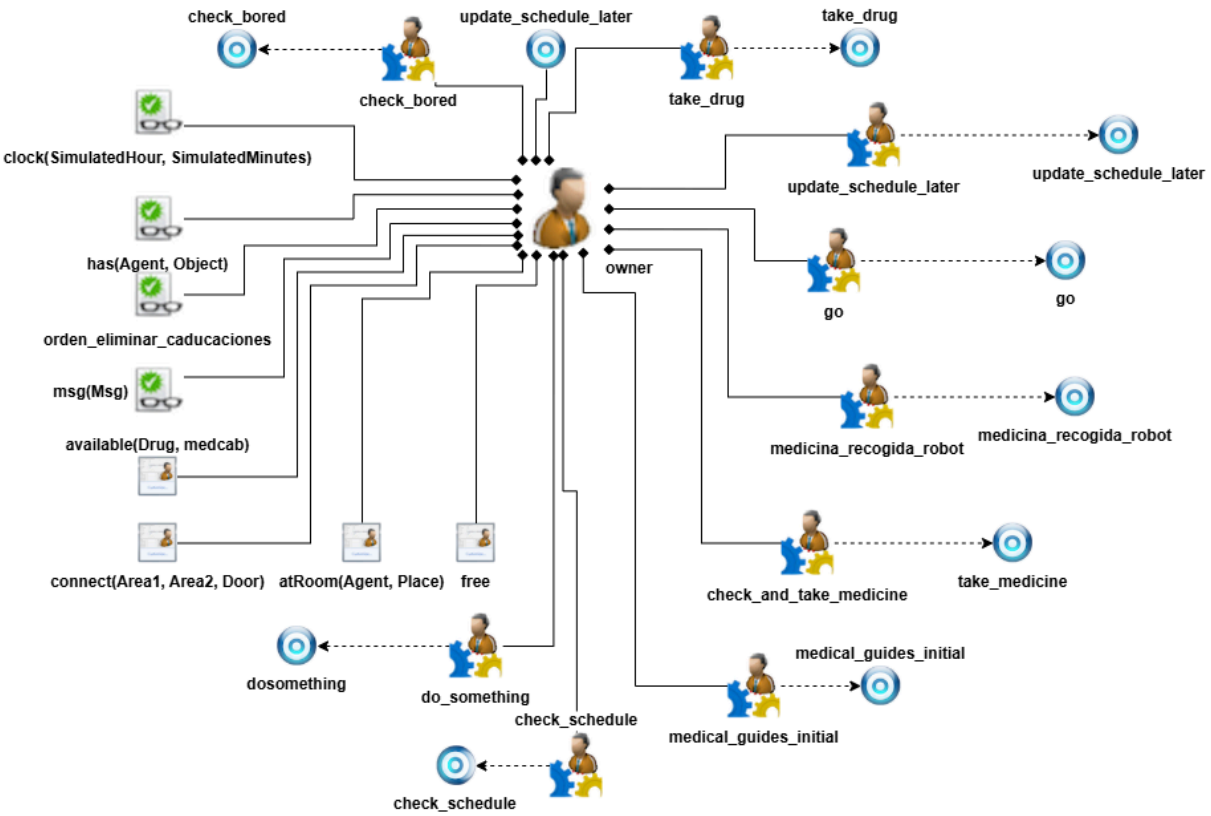

Figura 5: Diagrama de agente de owner

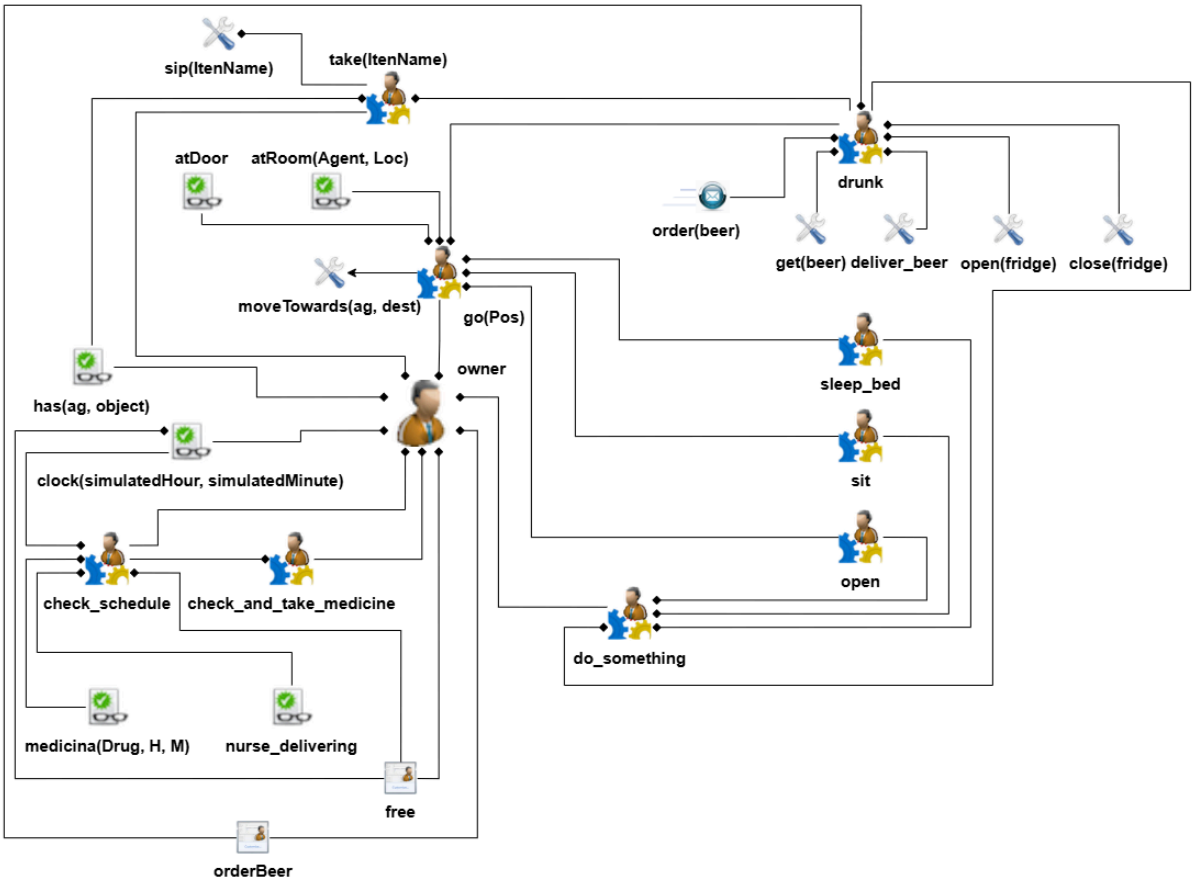

Figura 6a: Diagrama de tareas de owner

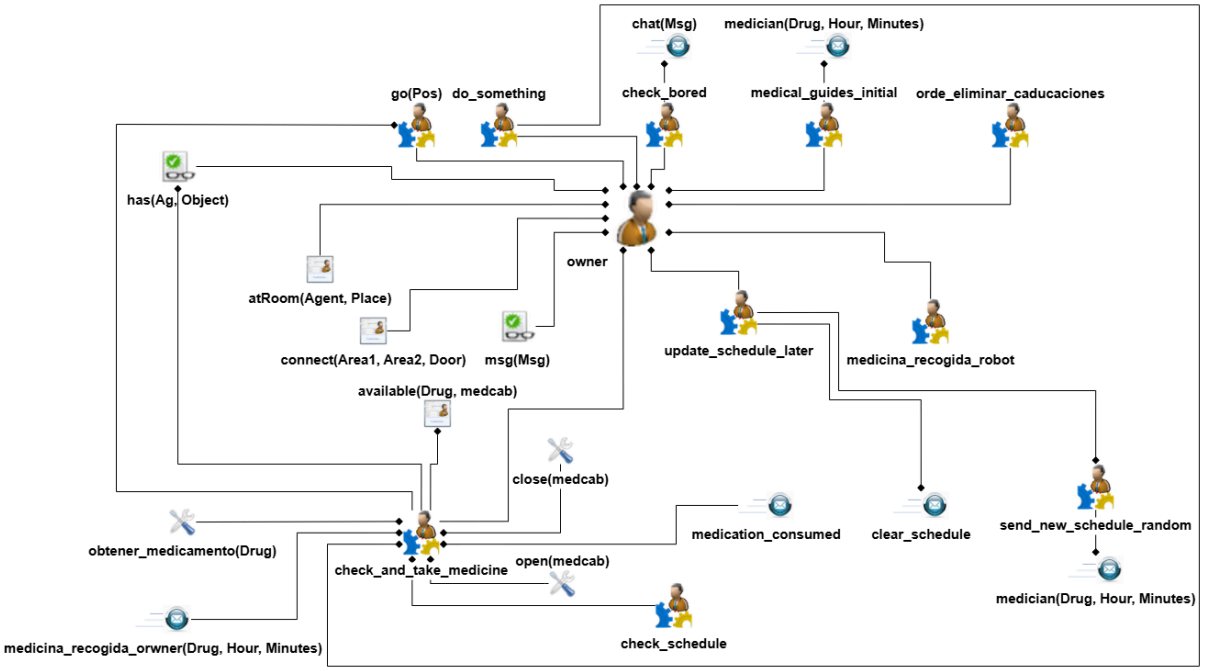

Figura 6b: Diagrama de tareas de owner

Creencias:

Antes de detallar los planes, es fundamental entender las creencias que guían las decisiones del owner:

- **connect**: Define el mapa de la casa, especificando qué habitaciones están conectadas y a través de qué puerta.
- **atRoom**: Creencias que indican en qué habitación de la vivienda se encuentra el agente.
- **atDoor**: Indica si el agente se encuentra en frente de una puerta o no.
- **available**: Conocimiento inicial sobre qué objetos (medicamentos, cerveza) están disponibles y dónde.
- **free**: Creencia que indica si el owner está disponible para iniciar una nueva tarea.

Las anteriores son creencias iniciales; sin embargo, durante la ejecución, el agente puede ir obteniendo lo siguiente:

- **clock**: Percepción que adquiere del entorno, donde indica la hora del entorno simulado en formato HH:MM.
- **has**: Creencia que indica si el owner tiene un objeto en la mano.
- **orderBeer**: Regla que comprueba si durante la ejecución hay que solicitar más cervezas, en caso de que no se encuentre en la base de creencias de owner la creencia available(beer, fridge).
- **msg**: Percepción avisando de que un agente del entorno escribió un mensaje al agente owner.
- **medician**: Percepción que se le añade el propio owner cuando tiene que mandar las pautas al robot enfermera.

Objetivos Iniciales:

El agente owner va a disponer de los siguientes objetivos iniciales:

- **medical_guides_initial**: El agente owner envía unas pautas iniciales al agente enfermera.
- **update_schedule_later**: Permite al owner cambiar dinámicamente las pautas tanto suyas como del agente enfermera al final de cada día simulado.
- **check_bored**: Este objetivo, de forma periódica, indica que el owner puede "aburrirse" y enviar un mensaje al robot para iniciar una conversación con él.
- **do_something**: Desencadena el plan de vida humana, cuya función es la realización constante de tareas por parte del owner
- **check_schedule**: Este objetivo será el responsable de ejecutar los planes relacionados con los medicamentos, cuya función es comprobar las pautas y decidir si ir o no a por la medicación.

A continuación, se describen los planes clave que el owner utiliza para satisfacer los objetivos anteriores:

Planes de simulación de vida humana

- **do_something**: Este plan es el principal responsable de la simulación de la vida humana que tendrá el agente owner, cuyo objetivo es la realización constante de tareas del owner por toda la casa, interactuando aleatoriamente con los objetos de la vivienda, siempre que no esté involucrado en una tarea de mayor prioridad como la obtención de un medicamento, en cuyo caso abandona la realización de tareas y se centra en los medicamentos. Las tareas que puede ejecutar aleatoriamente bajo este plan son:
    - **open**: Simula una visita, donde el agente se mueve a la puerta de entrada de la vivienda, estando unos segundos en la puerta.
    - **sit**: Selecciona aleatoriamente entre los 4 sillones y el sofá de la sala de estar, sentándose en uno de ellos.
    - **sleep_bed**: Al igual que sit, pero la diferencia es que selecciona una de las 3 camas disponibles de la vivienda para dormir.
    - **drunk**: Es el plan más complejo de esta categoría. El owner tomará una cerveza del frigorífico y se la bebe. Está formado por dos reglas que encadenan una secuencia de acciones:
        1. la primera, si se dispone de cervezas en el frigorífico va a la nevera, la abre **(open(fridge))**, coge una cerveza **(get(beer))**, cierra la nevera **(close(fridge)** y, finalmente, se sienta para consumirla.
        2. La segunda es si no quedan cervezas, donde se piden más cervezas **(order(beer))**, y una vez repuesto se realiza lo comentado en la primera regla. Para poder consumir la cerveza, se va a invocar a la tarea **take**.

Planes de Actualización para Pautas de Medicamentos:

- **medical_guides_initial**: Su única función es enviar unas pautas iniciales al robot enfermera, añadiendo también dichas pautas con las creencias medician a su propia base de conocimiento.
- **update_schedule_later**: El objetivo de este plan es borrar las pautas que queden en la base de creencias de los agentes para su futura actualización de estas mismas, donde primero envía un mensaje al robot para que borre sus pautas antiguas con la acción externa clear_schedule, luego borra las suyas propias y por último invoca a send_new_schedule_random.
- **send_new_schedule_random**: Este plan permite al owner modificar dinámicamente sus pautas al final de cada día de simulación, donde forma nuevas pautas aleatorias que son enviadas al robot enfermera y añadidas al propio owner mediante creencias medician.

Planes de Toma de Medicación:

- **check_schedule**: este es el objetivo central que gestiona la pauta médica del owner. Se ejecuta periódicamente para comprobar, a través de la percepción clock, si existe alguna pauta que deba tomarse a esa hora. Si encuentra una coincidencia y el medicamento no está caducado, desencadena el siguiente objetivo **check_and_take_medicine**: el cual es responsable de decidir cómo se obtendrá la

dosis, pudiendo dar lugar a los diferentes escenarios de obtención de medicamentos ya descritos en la sección 3.4.

- **check_and_take_medicine**: Este último plan simula la decisión humana del owner sobre cómo actuar ante una pauta, donde contiene la lógica para decidir aleatoriamente cuál de los tres escenarios (iniciativa propia, carrera o espera pasiva) se ejecutará, tal y como se describió en la sección 3.4.
    - ❖ **Si la decisión es "Competir" o "Ir Solo"**, el plan se propone el objetivo go y ejecuta la secuencia de acciones para recoger y consumir la medicina, notificando finalmente al robot con el mensaje medication_consumed.
    - ❖ **Si la decisión es "Esperar"**, el plan delega la tarea por completo al robot, y owner se mueve a un lugar de descanso mientras espera.

Plan de Consumo de Objetos:

- **take**: Este plan se activa a través del objetivo **take**, que es invocado por otros planes como drunk o después de recibir un medicamento. Su función es ejecutar la acción interna **sip** para simular el consumo y, una vez finalizado, eliminar la creencia **has**.

Planes de Navegación:

- **go**: Es una serie de planes que gestionan el movimiento paso a paso. Utilizando las creencias **connect, atRoom** y **atDoor**, junto la acción interna **move_towards**, responsable de que mueva el agente a través del algoritmo A*, explicado ya su funcionamiento en la sección 3.1. Este plan cuenta con las siguientes reglas de movimiento:
    - ○ Si el objetivo destino se encuentra en la misma habitación que el owner, se mueve directo a este.
    - ○ Si el destino se encuentra en una habitación contigua a la que está el owner, y este agente está en la puerta que interconecta habitaciones, se mueve directo al destino.
    - ○ Si el destino se encuentra en una habitación contigua a la que está el owner, y este agente no está en la puerta que interconecta habitaciones, se mueve a la puerta de conexión entre las habitaciones.
    - ○ El destino se encuentra en una habitación que conecta con una habitación intermedia, y esta conecta con la habitación donde se encuentra el agente owner, donde sabemos que el agente no está en la puerta de su habitación, por lo tanto, se moverá a la puerta para salir de la habitación.
    - ○ El destino se encuentra en una habitación que conecta con una habitación intermedia, y esta conecta con la habitación donde se encuentra el agente owner, donde sabemos que el agente está en la puerta de su habitación, por lo tanto, se moverá a la puerta de la habitación intermedia.
    - ○ No se mueve si el owner está en el destino

Planes Reactivos a Mensajes:

- **medicina_recogida_robot**: Este plan se activa cuando el robot le informa que ha recogido la medicina en el escenario de "la carrera" por la toma de medicamentos. La respuesta del owner es abandonar su propia intención de ir a por ella, eliminar la

pauta de su base de creencias y añadir **nurse_delivering** para ponerse en modo de espera, hasta que el robot le entregue la medicina.

- **orden_eliminar_caducaciones**: Al recibir este mensaje del auxiliar, el owner elimina de su base de creencias cualquier creencia sobre medicamentos caducados, actualizando así su conocimiento.
- **has**: Este plan reactivo es fundamental. Se dispara automáticamente en cuanto el owner recibe un objeto y la creencia **has** se añade a su base de conocimiento. Su única función es desencadenar inmediatamente el objetivo **take**, asegurando que el agente siempre intente consumir cualquier objeto que reciba.

4.3.2 Supermarket

El agente supermarket será el responsable y encargado de proporcionar las medicinas o cervezas que se hayan pedido, para su reposición.
Las Figura 7 y 8 representan los diagramas de agente y de tareas del agente supermarket.

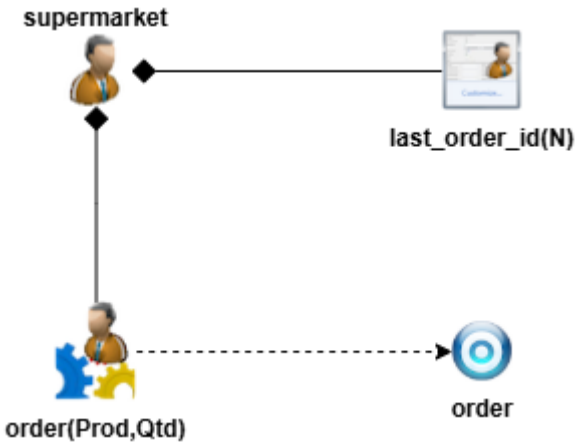

Figura 7: Diagrama de agente de supermarket

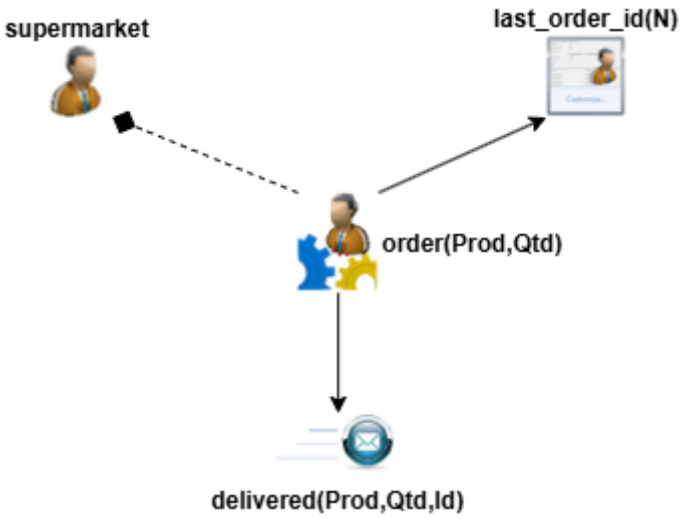

Figura 8: Diagrama de tareas de supermarket

Creencias:

El agente supermarket únicamente dispondrá de la siguiente creencia.

- **last_order_id**: Creencia inicial que representa el número del último pedido.

En cuanto a los objetivos iniciales, este agente no tiene ningún objetivo inicial, ya que su función es esperar a que lo contacten. Por otro lado, si tendrá un plan asociado.

Planes de envío de productos:

- **order**: Cuando un robot solicita una cantidad de productos al agente supermarket, este envía los productos al lugar de recogida establecido en el entorno, enviando un mensaje al agente solicitante de los productos para avisarle de la entrega. También actualiza la creencia last_order_id, incrementando en 1 y guardándola en la base de creencias del agente.

4.3.3 Robot enfermera.

La función del agente robot, también denominado enfermera, es la de actuar como el principal cuidador del owner. Es un agente proactivo que no solo se encarga de la dispensación de la medicación según las pautas establecidas, sino que también tiene la responsabilidad de verificar activamente que el owner cumple con su tratamiento. Además, debe gestionar sus propios recursos, como la energía, para garantizar que siempre esté operativo para sus tareas asistenciales.

Las Figuras 9, 10a y 10b representan el diagrama de agente del robot y sus diagramas de tareas, que se han separado para mayor claridad.

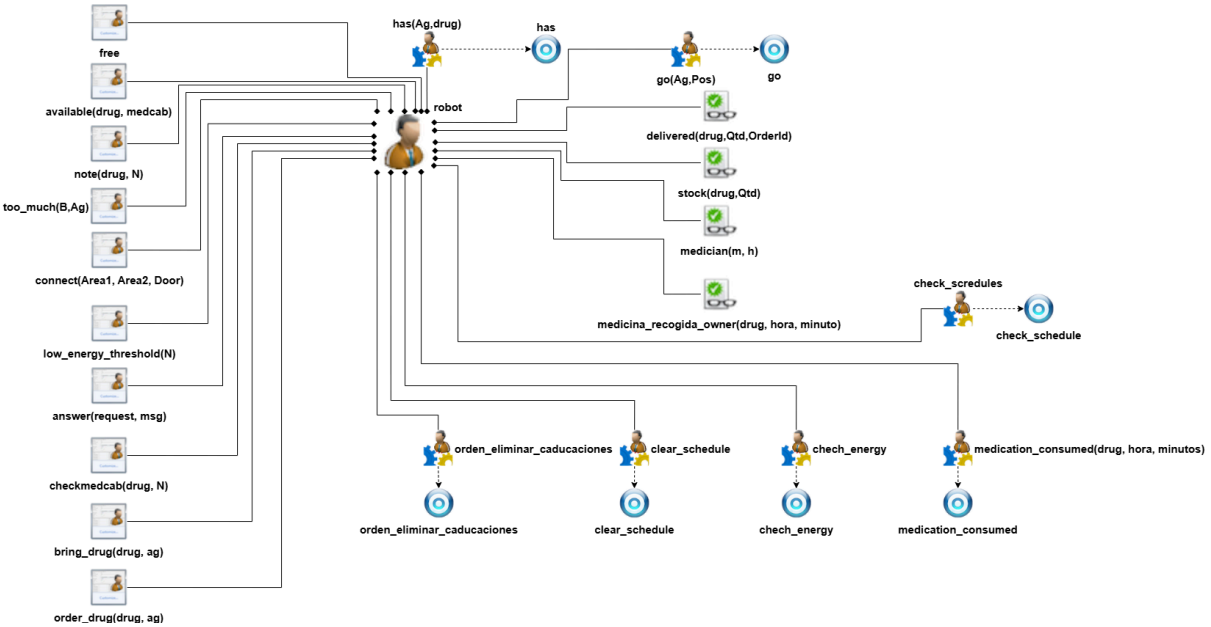

Figura 9: Diagrama de agente de robot enfermera

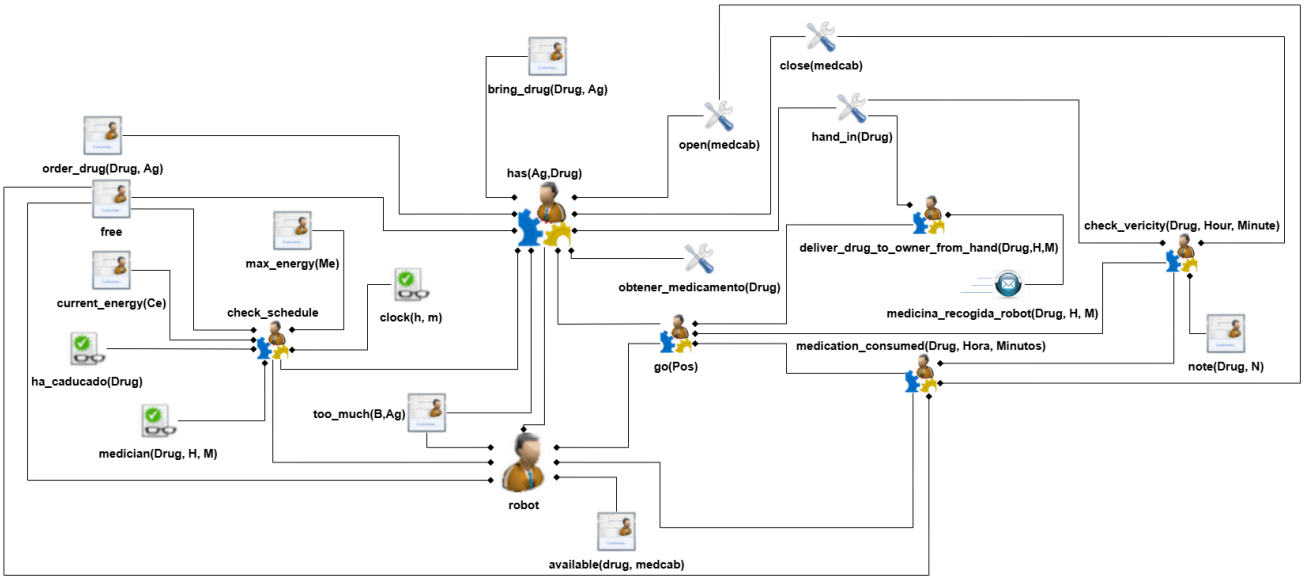

Figura 10a: Diagrama de tareas de robot enfermera

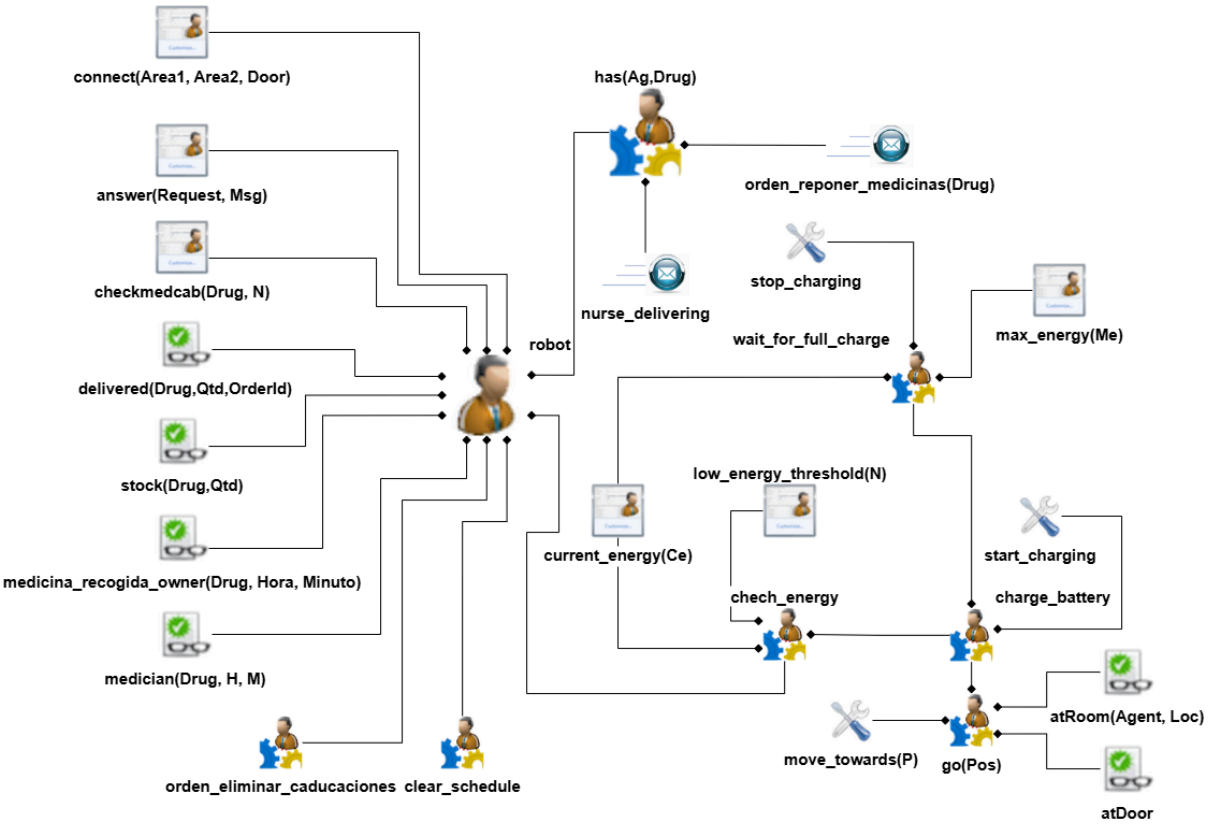

Figura 10b: Diagrama de tareas de robot enfermera

Creencias Clave:

El comportamiento del robot se basa en un conjunto de creencias que guían sus decisiones. Inicialmente dispone de las siguientes:

- **connect**: Define el mapa de la casa, especificando qué habitaciones están conectadas y a través de qué puerta.
- **atRoom**: Creencias que indican en qué habitación de la vivienda se encuentra el agente.
- **atDoor**: Indica si el agente se encuentra en frente de una puerta o no.
- **free**: Indica si el robot está disponible para iniciar una nueva tarea.
- **available**: Conocimiento sobre la disponibilidad de medicamentos.
- **note**: Creencia interna que el robot utiliza para "recordar" la cantidad de un medicamento que había antes de la verificación, para poder compararla con el stock actual.
- **low_energy_threshold**: Umbral numérico que, al ser superado por la energía actual, desencadenaría la necesidad de recargar.
- **answer**: Creencias que sirve de respuestas ante mensajes de conversación que llegan al robot enfermera.

Además de estas, el agente puede adquirir durante su ejecución las siguientes reglas o creencias:

- **checkMedCab**: Regla para comprobar si el agente owner ha consumido o no una medicina. Es importante esta regla para el plan check_vericity que se explica más adelante.
- **medician**: Almacena la pauta médica recibida del owner, que es la base para su objetivo check_schedule.
- **stock**: Percepción que actualiza la cantidad exacta de un medicamento al abrir el botiquín.
- **order_drug**: Creencia que indica la necesidad de pedir más unidades de un medicamento al supermarket.
- **bring_drug**: Creencia que se añade cuando el robot ha recogido un medicamento y está en proceso de entregarlo.
- **medicina_recogida_owner**: Mensaje que recibe del owner cuando este recoge la medicina por su cuenta.
- **delivered**: Mensaje del agente supermarket que confirma la entrega de un pedido.
- **clock**: Percepción del tiempo en formato HH:MM.
- **current_energy**: Percepciones sobre su estado energético actual.
- **max_energy**: Percepciones sobre su estado energético máximo.

Objetivos Iniciales:

El robot se inicia con dos objetivos fundamentales que se mantienen activos durante toda la simulación:

- **check_energy**: Monitorizar constantemente su nivel de batería para asegurar que nunca se quede sin energía en mitad de una tarea.

- **check_schedule**: Vigilar el reloj para cumplir con las pautas de medicación a la hora precisa.

Por último, definiremos los diferentes planes que se han elaborado para el robot enfermera, junto las acciones de cada plan:

Planes de Gestión de Energía:

- **check_energy**: Este plan se activa si el robot está libre (free) y su energía actual (current_energy**)** es inferior a al umbral indicado (low_energy_threshold). Su única función es desencadenar el objetivo **charge_battery** para iniciar el proceso de recarga.
- **charge_battery**: Contiene la lógica para ir al punto de carga y ejecutar la acción **start_charging**: comenzando la recarga.
- **wait_for_full_charge**: Una vez que está cargando, este plan se ejecuta hasta que la energía actual de enfermera (current_energy) coincide con la energía máxima posible (max_energy), momento en el que ejecuta la acción externa **stop_charging**. Si es interrumpido, la penalización por carga parcial (descrita en la sección 3.2) se aplicaría. Si el auxiliar estaba esperando, le envía un mensaje para notificarle que el cargador está libre.

Planes de Gestión de Medicamentos:

- **check_schedule**: Este es el plan central del robot enfermera. Comprueba si existe alguna pauta de medicación que tenga asignada la hora actual de la simulación. Si encuentra una coincidencia, la medicación a llevar no está caducada y tiene energía suficiente, desencadena el objetivo **has**, que inicia el proceso de entrega al agente owner.
- **has**: Este plan gestiona la obtención del medicamento. Podemos diferenciar cinco reglas para este plan:
    1. El primer plan se ejecuta cuando tenemos existencias del medicamento a entregar y el agente robot está libre. Por lo tanto, se mueve al cajón de medicamentos, lo abre **(open(medcab))**, coge la medicina **(obtener_medicamento)**, cierra el cajón de medicamentos **(close(medcab))** y se lo entrega al owner con **hand_in.**
    2. En caso de que el robot esté libre pero no haya unidades de este medicamento, se ejecuta esta segunda regla, donde se piden al agente supermarket, recogiéndose en el punto de entrega y reponiendo dicha medicina. A continuación se ejecutan los pasos de la regla uno.
    3. El tercer plan se produce cuando el robot está ocupado, donde espera cuatro segundos y vuelve a intentarlo.
    4. El cuarto plan sucede cuando se alcanza el límite máximo de medicamentos permitidos para ese día, informando al agente owner (está implementado, pero no operativo en el proyecto).
    5. El quinto plan es un tratamiento ante cualquier error inesperado que suceda.

- **medication_consumed**: Este plan reactivo se activa cuando el owner notifica que ha tomado una medicina. Su función es iniciar el proceso de verificación, desencadenando el objetivo **check_vericity**.
- **check_vericity**: Contiene la lógica de verificación. El robot va al botiquín, lo abre y compara el stock actual con la cantidad que tenía anotada en su creencia **note**. Si la cantidad ha disminuido, actualiza la creencia y da la tarea por finalizada. En caso contrario, concluye que el owner no la tomó, por lo que el robot coge él mismo la medicina y se la entrega.
- **clear_schedule**: Este plan reactivo se activa cuando recibe la orden del owner de borrar las pautas. Su función es eliminar todas las creencias **medician** de su base de conocimiento, debido a que se van a dar nuevas pautas.
- **deliver_drug_to_owner_from_hand**: Gracias a la 'Entrega estándar del robot respaldada por el auxiliar', forma de entrega de medicamentos explicada en la sección 3.4, el agente robot, siempre que esté libre, ejecuta este plan cuando el agente auxiliar le informa que ha traído la medicina solicitada y se la transfiere. Gracias a eso, el robot se desplaza hacia el owner y le entrega la medicina al owner por la acción externa hand_in. Finalmente, se envía un mensaje a owner informando que la medicina fue recogida por el agente robot (medicina_recogida_robot).

Planes de Navegación:

- Al igual que el owner, utiliza el plan **go** para moverse por el entorno, usando las creencias connect, atRoom y atDoor, junto la acción interna move_towards para moverse entre habitaciones.

Planes Reactivos a Mensajes:

- **orden_eliminar_caducaciones**: Al recibir este mensaje del auxiliar, el robot elimina de su base de conocimiento las creencias relacionadas con los medicamentos caducados, manteniéndose sincronizado con el estado real del inventario.

4.3.4 Robot auxiliar

El agente auxiliar tendrá la funcionalidad de traer más medicación de supermarket, en caso de que esté caducada. Además, contará con energía limitada, al igual que robot, e incluso traerá la medicina del cajón de medicamentos a robot, en caso de que no le de la energía suficiente a robot.

Las Figuras 11, 12a y 12b representan su diagrama de agente y los diagramas de tareas respectivamente, de nuevo estos últimos separados para una mayor claridad.

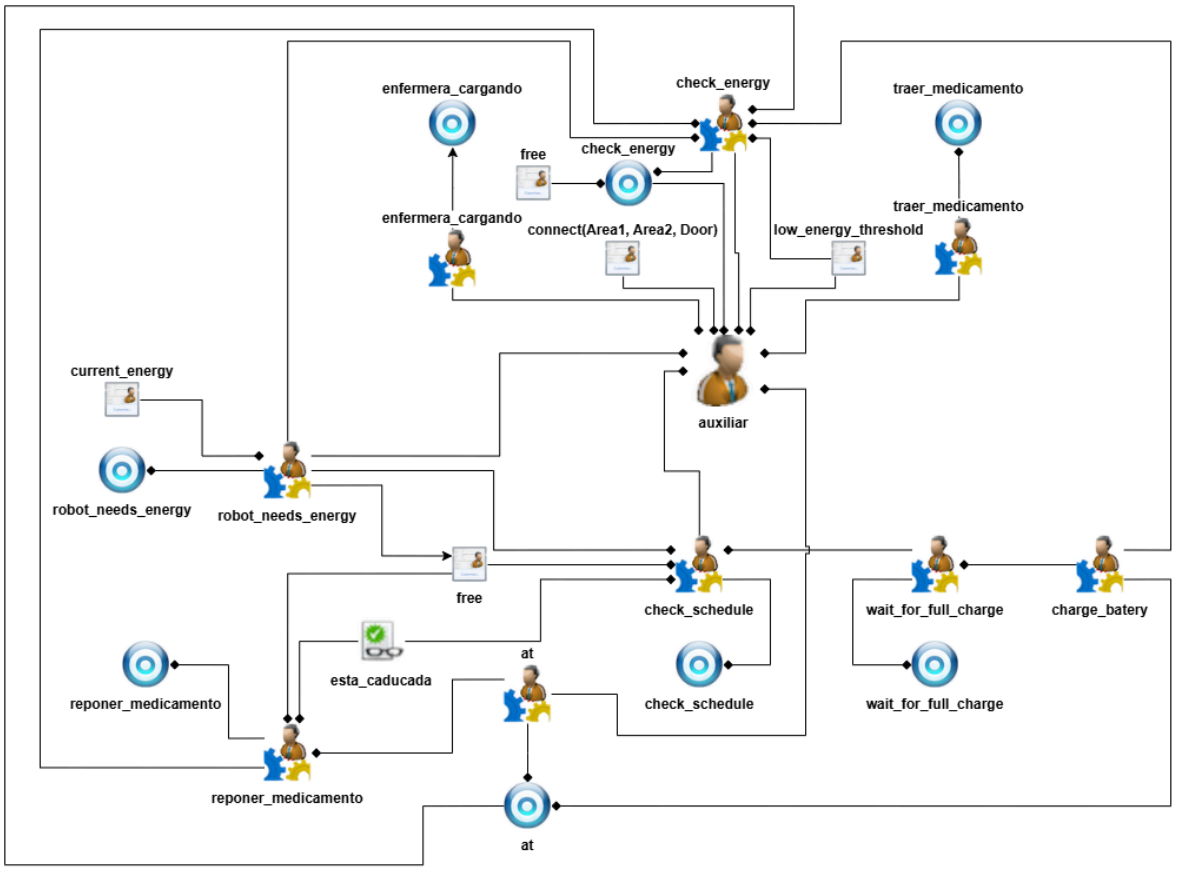

Figura 11: Diagrama de agente de auxiliar

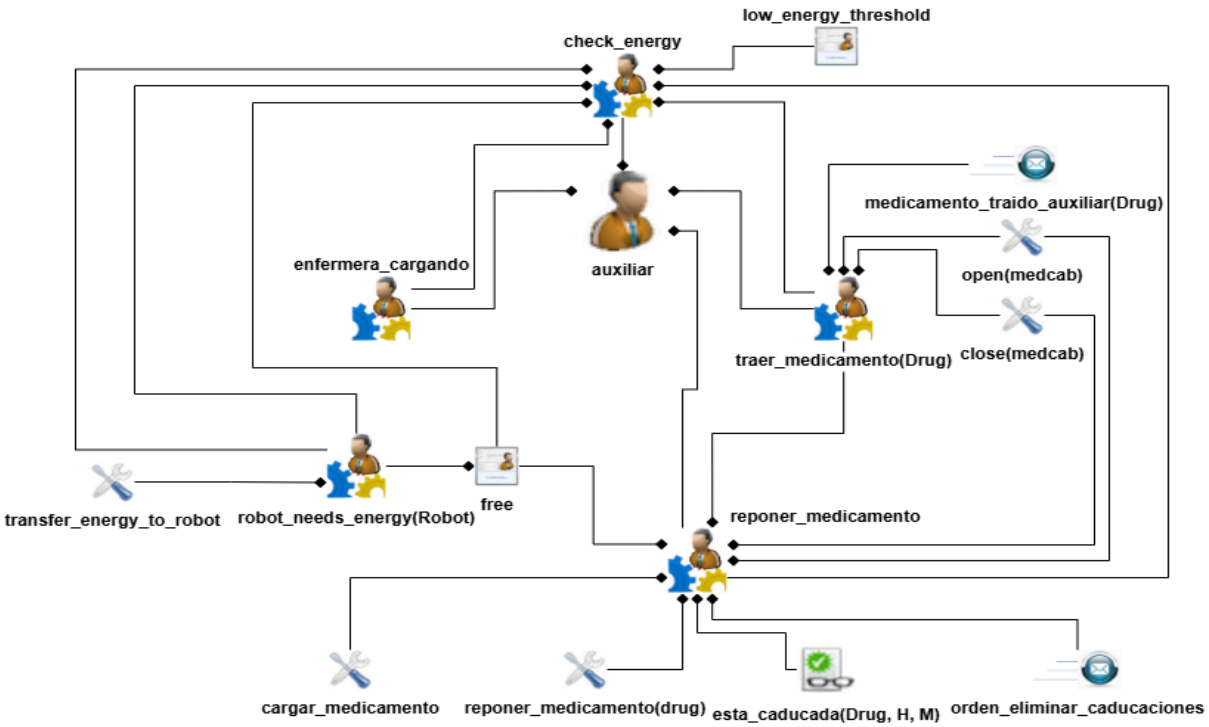

Figura 12a : Diagrama de tareas de auxiliar

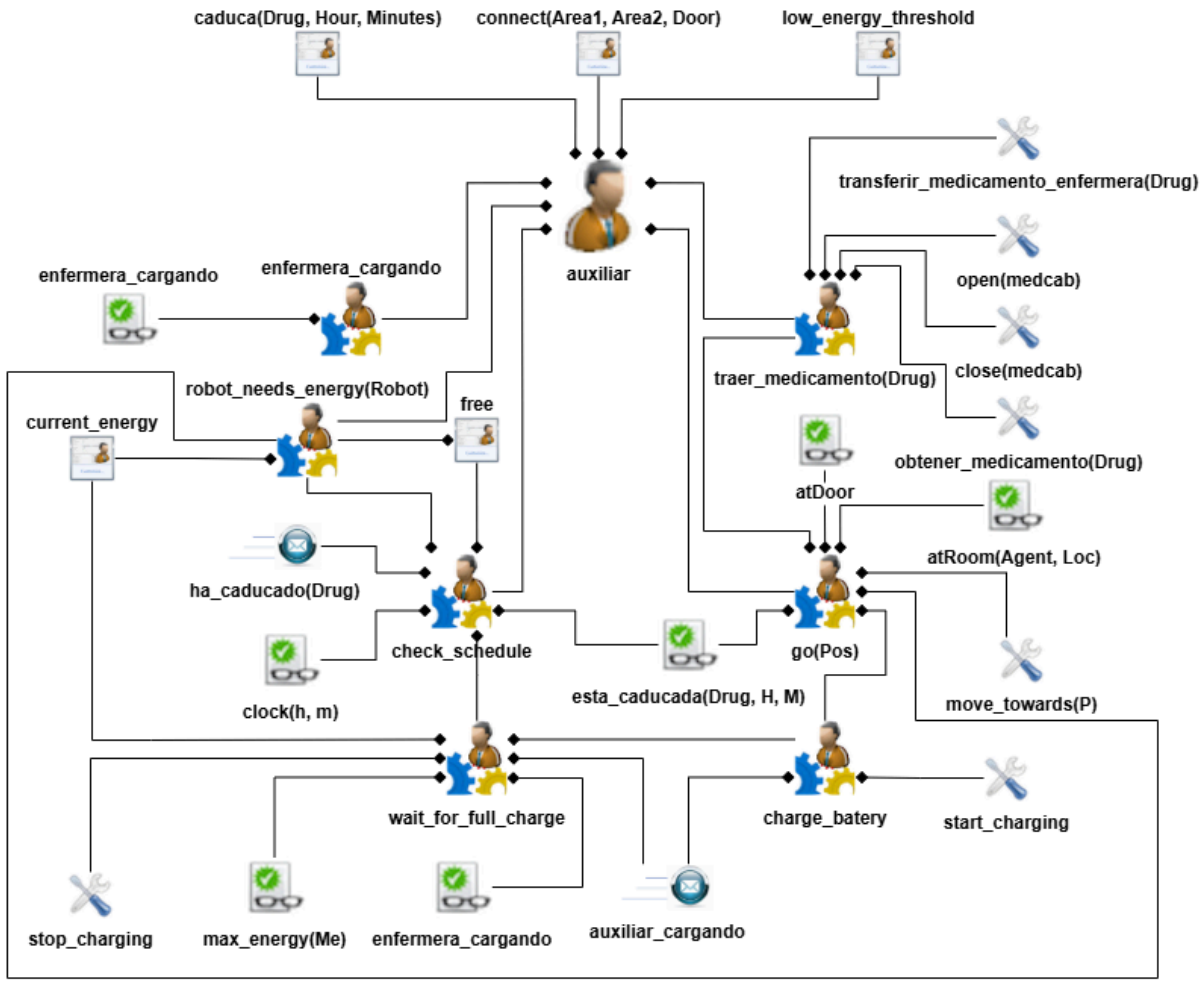

Figura 12b: Diagrama de tareas de auxiliar

Creencias:

Para el agente auxiliar, inicialmente dispondremos de las siguientes creencias:

- **connect**: Define el mapa de la casa, especificando qué habitaciones están conectadas y a través de qué puerta.
- **atRoom**: Creencias que indican en qué habitación de la vivienda se encuentra el agente.
- **atDoor**: Indica si el agente se encuentra en frente de una puerta o no.
- **caduca**: Creencias que simbolizan cuantas horas y minutos le quedan a cada medicamento para caducar.
- **low_energy_threshold**: Umbral que dispone auxiliar, indicándonos que vaya a cargar si al revisar su energía actual, esta es inferior al umbral.
- **free**: Creencia que indica si el agente auxiliar se encuentra libre.

Adicionalmente, se pueden añadir durante la ejecución las siguientes creencias:

- **max_energy**: Creencia que indica la carga máxima disponible a cargar para el auxiliar.
- **clock**: Creencia que recibe del entorno, indicando la hora simulada en formato HH:MM.
- **current_energy**: Creencia indicando la energía actual que posee el auxiliar.

- **enfermera_cargando**: Creencia de que el cargador está ocupado por enfermera.
- **esta_caducada**: Se añade a la base de conocimientos cuando hay un medicamento caducado.

Objetivos Iniciales:

Inicialmente, el agente auxiliar dispondrá de los siguientes objetivos:

- **check_energy**: Objetivo que comprueba constantemente la energía que dispone el robot auxiliar, desencadenando los planes de gestión de energía.
- **check_schedule**: Objetivo que gestiona las caducidades de los medicamentos, en caso de que haya algún medicamento caducado.

Finalizando, el agente auxiliar tendrá diferentes planes para ejecutar:

Planes de gestión de caducidad de medicamentos

- **check_schedule**: Este plan es el eje central de la gestión de caducidad de medicaciones. Se puede agrupar en tres reglas:
    1. La primera y la regla principal, donde si el auxiliar está libre (free) y le llega una creencia del entorno indicando la hora, este comprobará si existe algún medicamento que esté caducado. Si existe, notifica con un mensaje a la enfermera con la creencia **ha_caducado** indicando que el medicamento a comprobar está caducado, y se llama al plan **reponer_medicamento**.
    2. La segunda regla sucede cuando el agente auxiliar está ocupado, donde espera unos milisegundos y vuelve a intentarlo.
    3. La tercera regla ocurre cuando el agente no ha recibido la creencia de la hora, por lo que espera y vuelve a intentarlo.

- **reponer_medicamento**: Plan para reponer un medicamento que ha caducado. El auxiliar se dirige al cajón de medicamentos, donde retira el medicamento caducado (cargar_medicamento) y se obtendrá nuevo medicamento. Una vez obtenido, abrimos el cajón de medicamentos **(open(medcab))**, lo reponemos con la acción interna **reponer_medicamento** y cerramos el cajón **(close(medcab))**. Una vez finalizado, enviamos mensajes a los agentes owner y robot informando que eliminen las creencias de medicaciones caducadas que poseen.

Planes de gestión para medicamentos:

- **traer_medicamento**: Este plan tiene la funcionalidad de ayudar al robot auxiliar a llevarle la medicación cuando se le solicite. El auxiliar se moverá al cajón de medicamentos, donde lo abrirá, obtendrá el medicamento que solicita la enfermera y cierra el cajón. Luego se mueve junto al robot enfermera y le transfiere el medicamento al robot a través de la acción interna **transferir_medicamento_enfermera**: notificando a enfermera que se le pasó el medicamento deseado.

Planes de gestión de energía

- **check_energy**: Este plan se activa si auxiliar se encuentra libre (free) y su energía actual (current_energy**)** es inferior al umbral indicado (low_energy_threshold**)**. Desencadena el objetivo **charge_battery** para iniciar el proceso de recarga.
- **charge_battery**: El auxiliar se mueve hasta el cargador y comienza la carga con la acción interna **start_charging**.
- **wait_for_full_charge**: Una vez que está cargando, este plan se ejecuta hasta que la energía actual de auxiliar (current_energy) coincide con la energía máxima posible (max_energy), momento en el que ejecuta la acción externa **stop_charging** para detener la carga. Si es interrumpido, la penalización por carga parcial descrita en la sección 3.2 se aplicaría. Si la enfermera estaba esperando, le envía un mensaje para notificarle que el cargador está libre.
- **robot_needs_energy**: Plan que se activa si el agente robot le envía un mensaje notificando que se quedó sin energía y necesita que el agente auxiliar le transfiera energía. Podemos desglosarlo en 3 reglas:
    1. La primera y la principal, sucede si el agente auxiliar está libre y posee energía suficiente para transferir. En ese caso, se desplaza hacia el agente robot y le transfiere parte de su energía para que pueda seguir gracias a la acción interna **transfer_energy_to_robot**. Una vez hecho, revisa si necesita cargar energía o si hay que cambiar medicaciones caducadas.
    2. En caso de que esté libre pero no posea la suficiente energía para transmitir, hace caso omiso a la petición del robot y continúa realizando otras tareas.
    3. Si en cambio el auxiliar está ocupado, espera cinco segundos y lo vuelve a intentar.

Planes de Navegación

- **go**: Al igual que el owner y enfermera, utiliza el plan **go** para moverse por el entorno, usando las creencias connect, atRoom y atDoor, junto la acción interna move_towards para moverse entre habitaciones.

Planes reactivos a mensajes

- **esta_caducada**: Si al agente auxiliar le llega una notificación de que una medicación está caducada, procederá a llamar a la tarea reponer_medicamento para su reposición, eliminando la creencia caduca de su base de creencias.

5. Bibliografía

Bordini, R., Hübner, J., & Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons. Obra de referencia para el desarrollo de sistemas multiagente con Jason. Aporta los fundamentos del modelo BDI y del lenguaje AgentSpeak(L).

MOOVI – Plataforma de docencia virtual de la Universidad de Vigo
Plataforma utilizada para el acceso al enunciado del proyecto, documentación de apoyo, recursos docentes y entregas.

Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson. Texto académico clásico que aborda los fundamentos de la inteligencia artificial, incluidos los modelos de agentes, heurísticas y algoritmos de planificación.