Proyecto de SI

SI-31

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

Índice de Contenidos

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.
Para eso, los agentes del entorno deben poder moverse libremente, pudiendo sortear obstáculos y pasar de habitaciones a otras a través de puertas, 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*.
En segundo caso, los robots, como autómatas que son, dispondrán de un consumo de energía, que será necesario controlar durante el desarrollo del entorno, evitando que se queden sin batería en medio de tareas.
Para mejorar la visibilidad, se ha desarrollado dos pestañas nuevas que saldrán durante la ejecución: la primera de ellas es un reloj, que marcará una hora simulada dentro del entorno, y la segunda consiste en una pestaña con datos sobre las baterías de los robots indicando cuánto le quedan a cada uno de batería.
En la siguiente imagen podemos observar la simulación del entorno domótico, con sus agentes, objetos y estructura de la casa, además de las pestañas que se comentaron anteriormente.

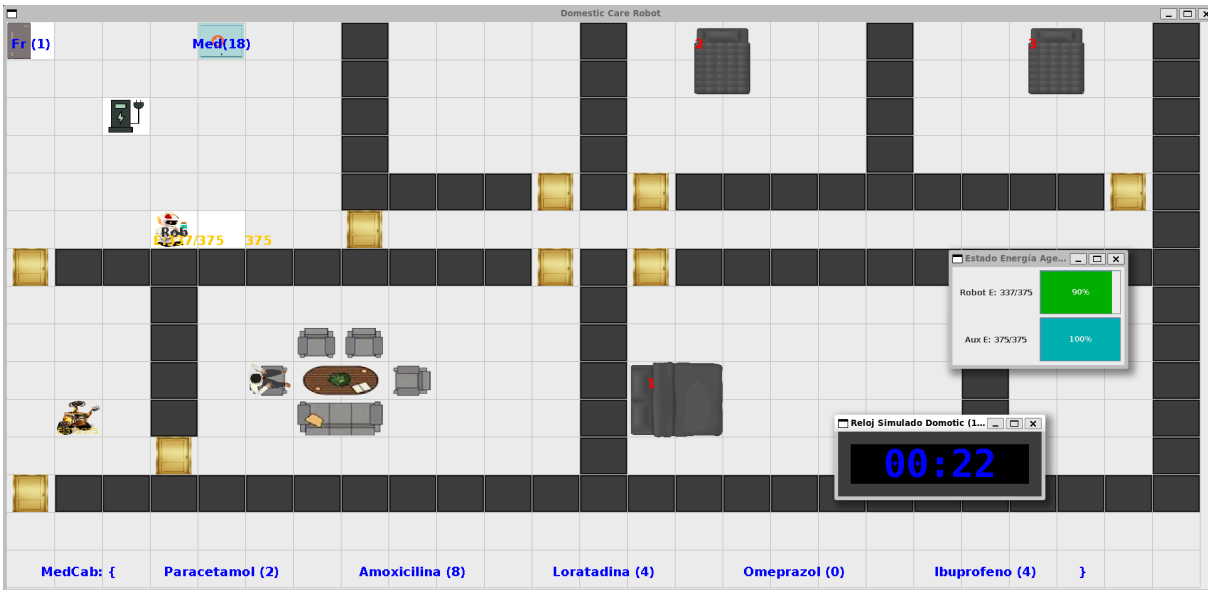

Imagen 1: Simulación de entorno

Diagramas

A partir de ahora, se va a proceder a definir las características del entorno que se están desarrollando, junto las funcionalidades de cada agente que hay en el entorno. Comenzaremos definiendo como hemos diseñado nuestro entorno, que se puede ver en la Figura 1.

Diagrama entorno.

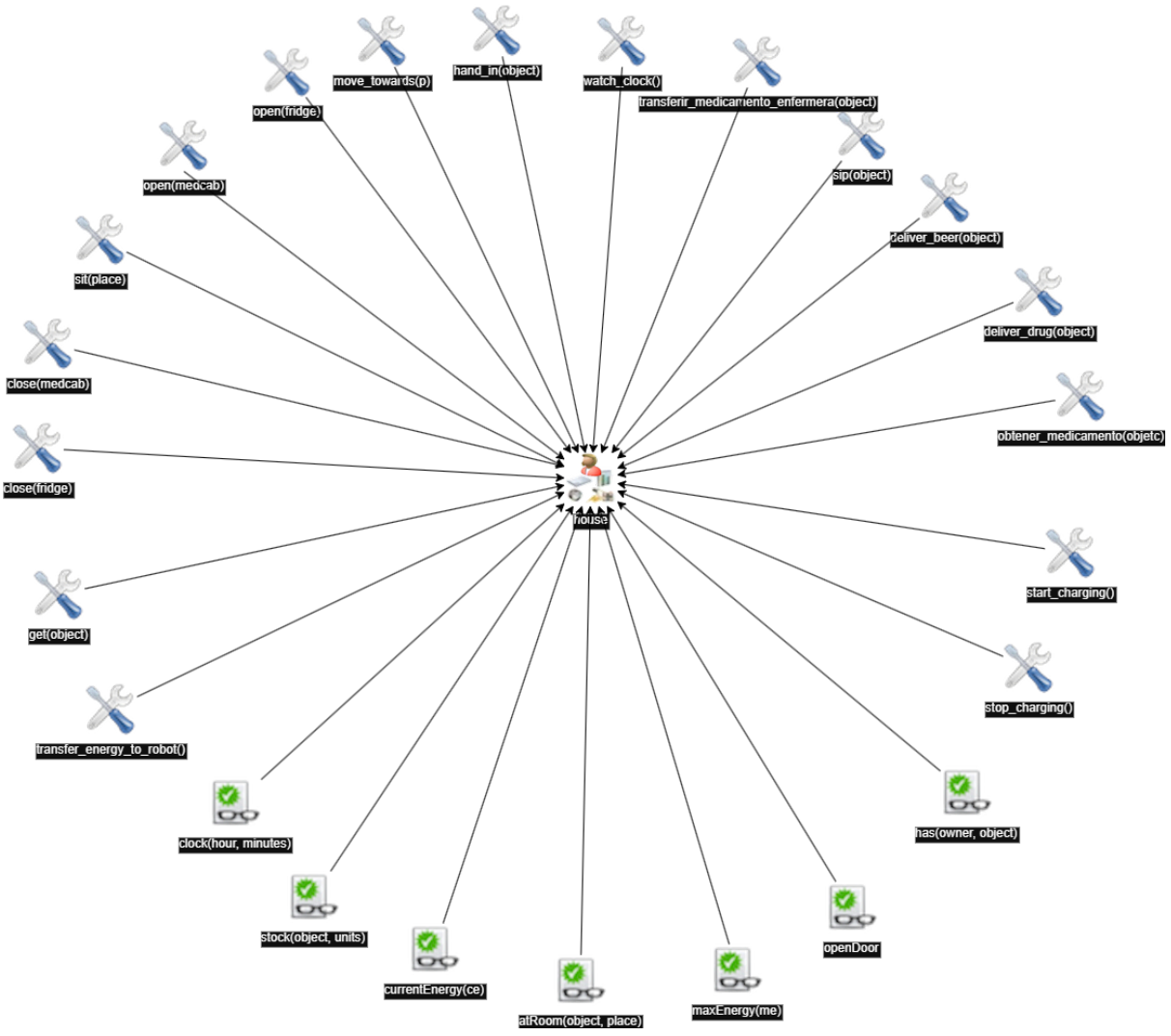

Figura 1: Diagrama de entorno

En dicha Figura 1, observamos el propio entorno House, con el que interactúan nuestros agentes: owner, robot, auxiliar y supermarket. 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 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.

Diagrama organización para robot doméstico

Una vez introducidas las creencias y las acciones que pueden realizar los agentes en el entorno que nos ofreció la Figura 1, vamos a proporcionar una organización para realizar el objetivo de darle el medicamento al owner, que nos proporciona la Figura 2.

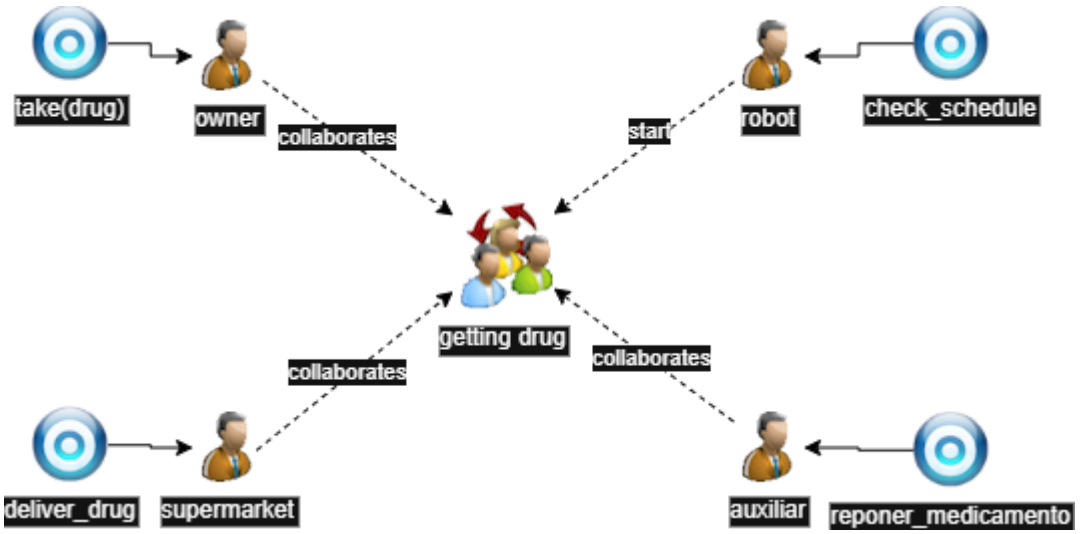

Figura 2: Diagrama de organización para robot doméstico

En la Figura 2 podemos observar que, disponemos de los agentes owner, robot, auxiliar y supermarket, y entre éstos van a colaborar para lograr un objetivo, en este caso qué owner obtenga la medicación para tomar (Interacción **getting drug**). Se puede observar, que quien inicia la acción es el agente robot, con el objetivo **check_schedule**, verificando que hay una medicación que necesita entregar a owner. Por su parte, el resto de agentes colaboran para realizar la acción, siendo auxiliar el que vaya a por el medicamento y se lo lleve a robot para entregarlo a owner(objetivo **reponer_medicamento** en caso de que robot no le de batería suficiente para hacer la tarea completa), o bien supermarket, quien entrega más medicación en caso de que auxiliar le haya comunicado que se agotó o caduco (objetivo **deliver_drug**).

Diagrama de objetivos ideal para robot doméstico

A continuación, en la Figura 3 podemos observar el flujo de objetivos ideal que sería necesario para llevar a cabo la entrega de medicación al owner.

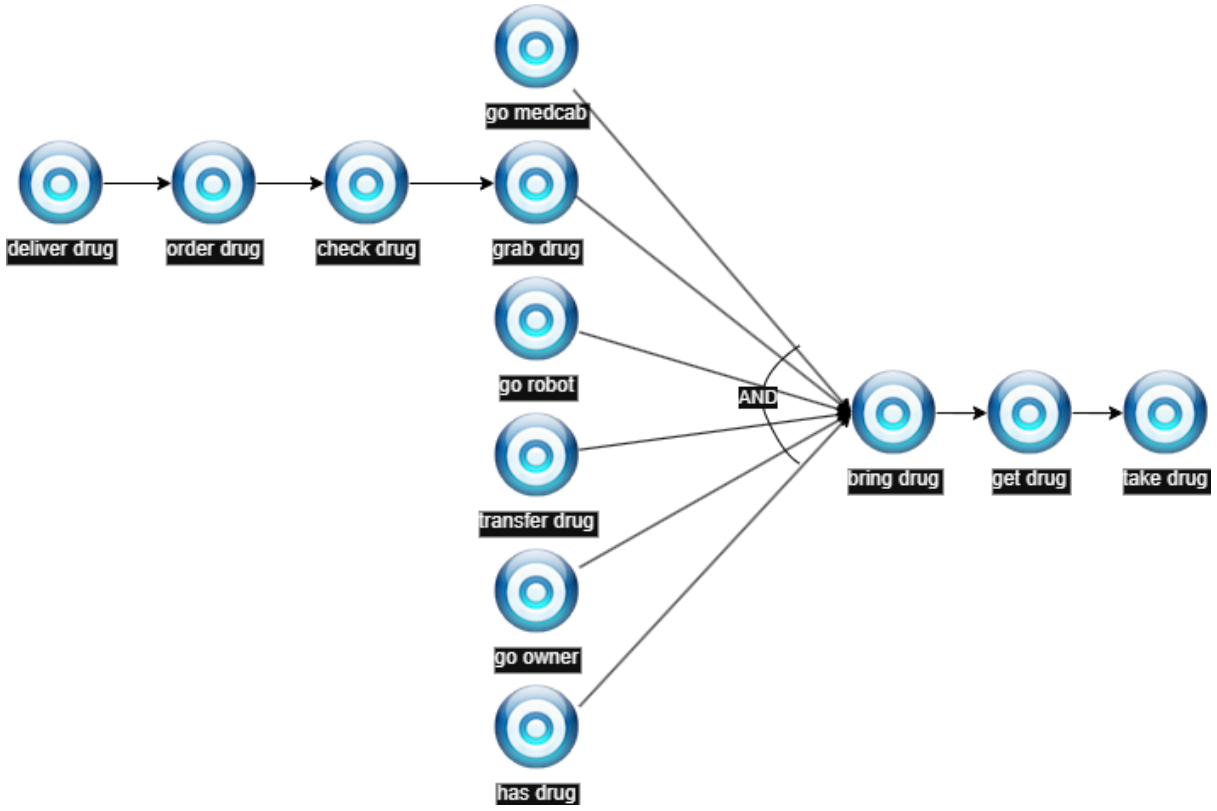

Figura 3: Diagrama de objetivos ideal para robot doméstico

Para cumplir el objetivo de que owner tome la medicina (Objetivo **take drug**) necesitamos que robot le de la medicina a owner (Objetivo **get drug**). Pero para realizar eso, el robot precisa de que obtenga la medicina (**bring drug**). Eso se hace de lo siguientes: al cajón de medicamentos (**go medcab**), coger un medicamento (**grab drug**), va junto robot (**go robot**), se lo da para entregarlo(**transfer drug**), ir junto a owner (**go owner**) y entregarlo (**has drug**). Por otra parte, cuando estemos en el cajón de medicamentos para obtener el medicamento, hay que comprobar que queden medicamentos (**check drug**), y en caso de que no sea el caso, pedirlos (**order drug**), donde estos se entregarán a robot cuando supermarket satisfaga el objetivo **deliver drug**.
Tal y como se acaba de describir los objetivos ideales de la Figura 3, podemos observar como cuatro de los agentes (owner, robot, auxiliar y supermarket) se relacionan entre sí para satisfacer un objetivo concreto.

Diagrama de objetivos para robot doméstico

Partiendo del diagrama de la Figura 3, podemos obtener un nuevo diagrama, descomponiendo más las tareas, como es la Figura 4.

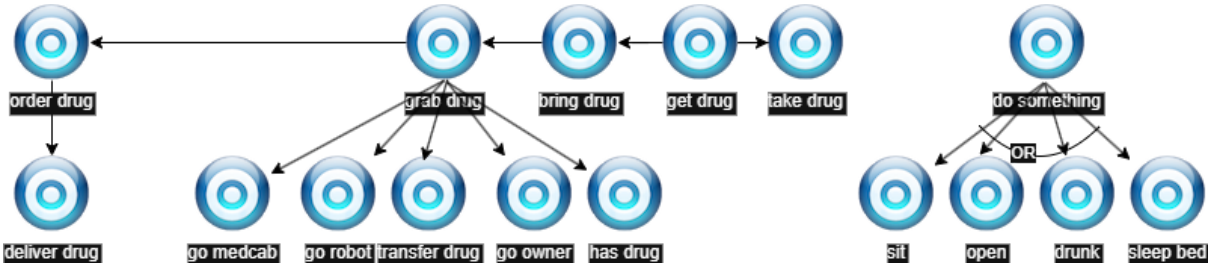

Figura 4: Diagrama de objetivos para robot doméstico

Teniendo los mismos objetivos ya mencionados y explicados en la Figura 3, partimos que para qué owner tome la medicación, debe de llevarla y darla de auxiliar a robot, y este la lleva y entrega a owner. Por otra parte, si no quedan medicamentos, se deben pedir y supermarket realizará dicho envío de medicamentos.
Además, owner estará haciendo algo (**do_something**), que puede ser atender una visita (**open**), sentarse en una silla o sofá (**sit**), dormir un rato en una de las camas (**sleep_bed**) o tomar una cerveza del frigorífico (**drunk**).

Diagrama de interacción para robot doméstico

Ya vistos los diagramas de objetivos anteriores, podemos observar que la Figura 2, describe un diagrama de organización, donde los agentes ejecutan unos objetivos para realizar la tarea, pero no sabemos qué interacciones puede haber entre ellos cuando realizan los objetivos, y esa interacción entre agentes es lo que vamos a explicar de la siguiente figura.

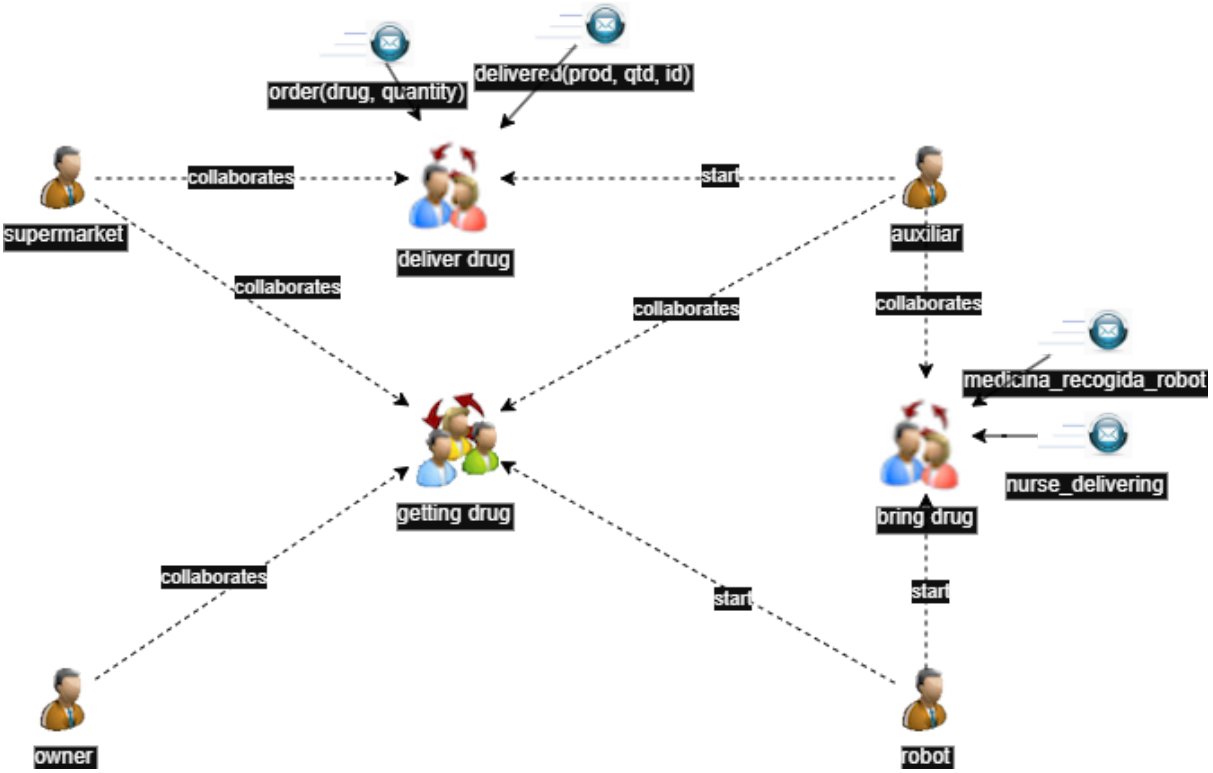

Figura 5: Diagrama de interacción para el robot doméstico

En la Figura 5, podemos observar la interacción para que el owner consuma el medicamento (**getting drug**), donde es iniciado por robot y colaboran tanto owner, supermarket y auxiliar. Pero fijándonos bien, y como se puede deducir de las explicaciones de los diagramas de objetivos descritos en las Figuras 3 y 4, se puede ver que robot inicia una conversación (**bring drug**) con auxiliar, en caso de que no le de la energía suficiente para ir al cajón a por la medicina. Auxiliar colabora para llevarle la medicina al robot, y cuando la obtenga, robot informa al owner de que le esta llevando la medicina. Otra situación de la que se habló varias veces, es el caso de que haya que pedir medicinas, ya sea porque caducaron o porque no quedan existencias, iniciando una conversación con supermercado (**deliver drug**), donde le mandará un mensaje del pedido, y supermarket enviará un mensaje indicando la entrega del pedido.

Con esto, finalizamos las explicaciones relacionadas a la organización del entorno, con sus creencias y percepciones; los objetivos a cumplir y las interacciones entre agentes. Los siguientes diagramas describen más detalladamente las funciones de los 4 agentes del entorno.

Owner

El agente owner representa al propietario en el entorno diseñado, cuyo objetivo será estar haciendo cosas por la casa; recibiendo medicación del agente robot, llegando incluso a poder tomar medicación el mismo, siempre que le comunique a robot que ha tomado (o puede que se le haya olvidado tomarla) para verificar este último que es cierto.
Las Figuras 6 y 7 representan el diagrama de agente de owner y el diagrama de tareas de owner respectivamente.

Diagrama agente.

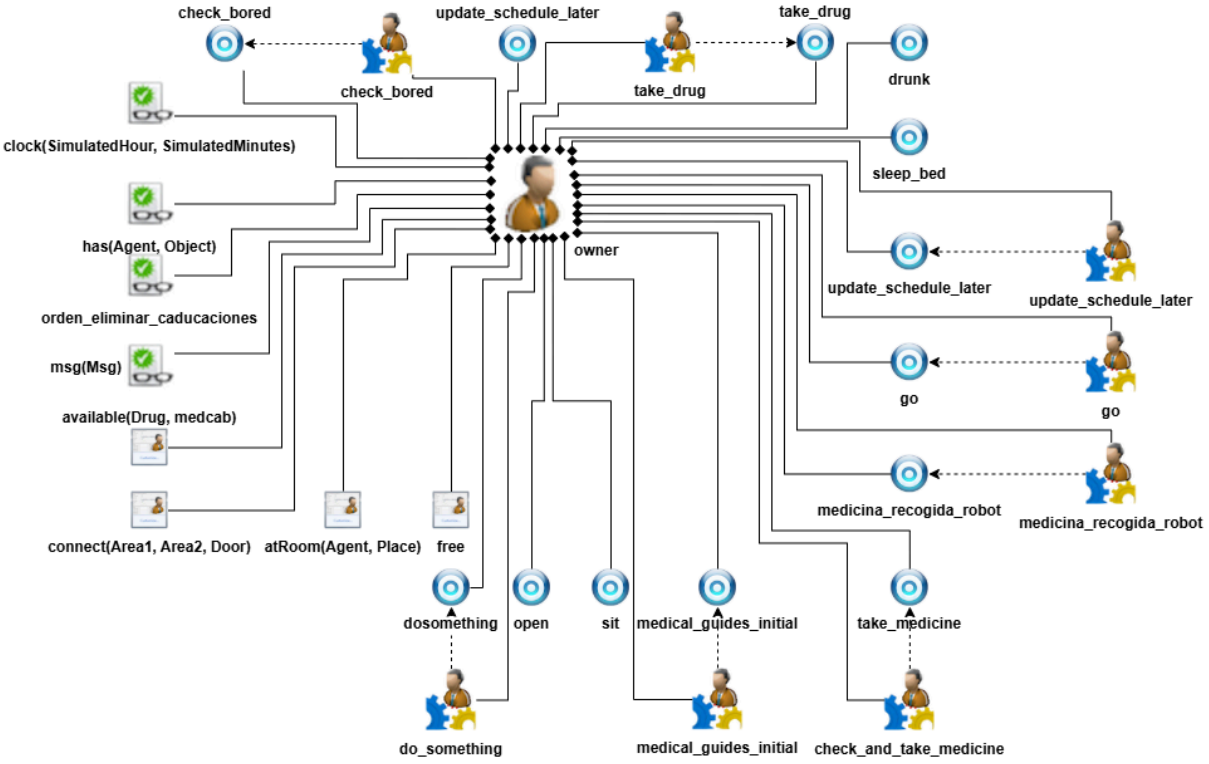

Figura 6: Diagrama de agente de owner

Diagrama de tareas.

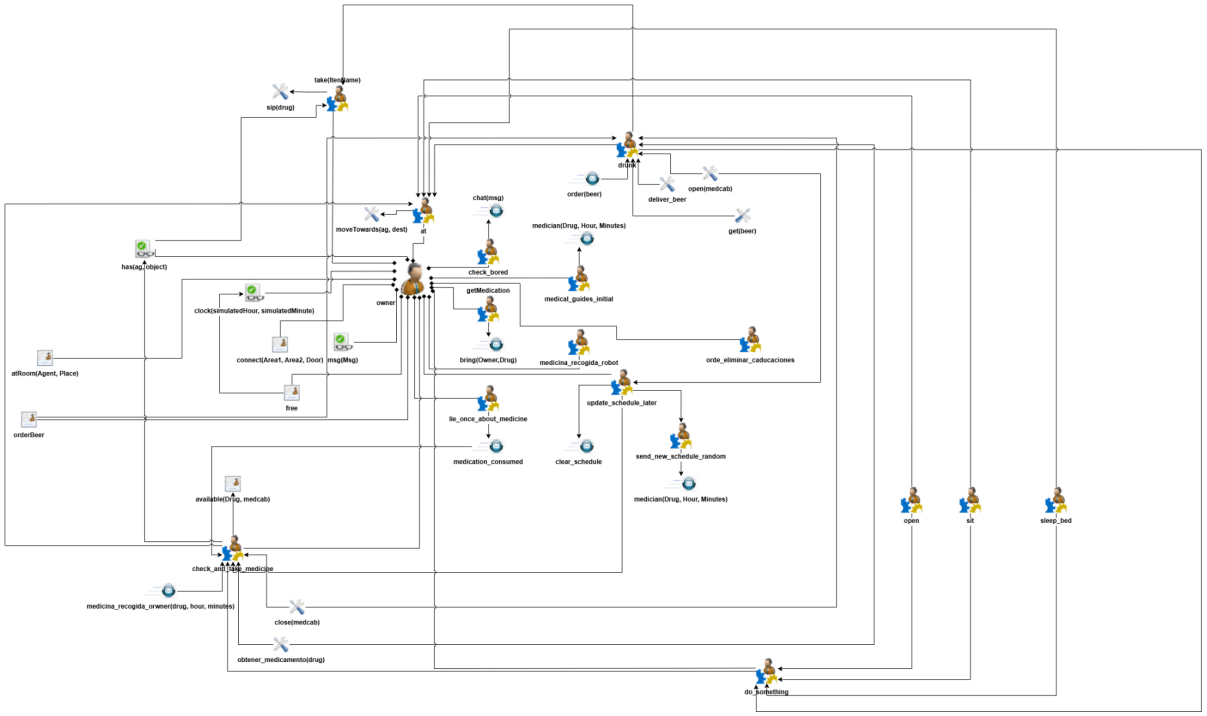

Figura 7: Diagrama de tareas de owner

El agente owner va disponer de los siguientes planes iniciales:

- **medical_guides_initial**, que satisface el objetivo **medical_guides_initial**, como se muestra en la Figura 6, y este envía a enfermera las pautas de medicación (**medician(Drug, H, M)**).
- **update_schedule_later**, satisfaciendo el objetivo **update_schedule_later**, que invoca la tarea **send_new_schedule_random**, donde actualiza las pautas borrando las que quedarán y mandando nuevas (**medician(Drug, H, M)**).
- **check_bored**, ejecutando el objetivo **check_bored**, donde envía mensajes a robot, interactuando algo (**chat(msg)**).
- **do_something**, que ejecuta objetivo **do_something**, cuya función es que owner esta siempre (quitando cuando valla a por algún medicamento) haciendo algo. Las tareas que puede ejecutar aleatoriamente son:
    - **open**, donde va a la puerta y simula durante segundo la atención a alguien.
    - **sit**, que selecciona aleatoriamente un objeto para sentarse unos segundos.
    - **sleep_bed**, seleccionando aleatoriamente una cama para dormir un rato.
    - **drunk**, donde va a la nevera, la abre (**open(fridge)**), coje una cerveza (**get(beer)**), cierra la nevera (**close(beer)**), y se va al sofá para tomarla. Si no quedan, las pide a supermarket (**order(beer)**) y repone yendo al punto de entrega, volviendo al fridge y reponiéndolas (**deliverbeer(Object, N)**) Para tomarla ejecuta la acción **take(Object)** (siempre que no tenga algo ya en la mano [**has(ag, Object)**]), que ejecuta la acción interna **sip(beer)** Estas relaciones se pueden ver en la Figura 7.
- **check_schedule**, que comprueba con el **clock(H,M)** y que no esté caducada, si va o no a por la medicina, y eso se hace con la tarea

check_and_take_medicine(Drug, Hour, Minutes) si decide: ir owner y robot a por ella, en ese caso owner va al cajón de medicamentos. Si llega owner primero, lo abre, coje el medicamento, cierra y le envia un mensaje a robot para que compruebe (**medication_consumed**); si va el robot a por la medicina y que owner no, entonces le envia mensaje de que espera (**esperar_medicina**) o si va owner a por ella, en ese caso va al cajón, coje la medicina, se la toma y envia un mensaje a robot para que lo compruebe si es verdad (**medication_consumed**).

Supermarket

El agente supermarket será el responsable y encargado de proporcionar las medicinas o cervezas que se hayan pedido, para su reposición.
Las Figuras 8 y 9 representan los diagramas de agente y tareas respectivamente.

Diagrama agente.

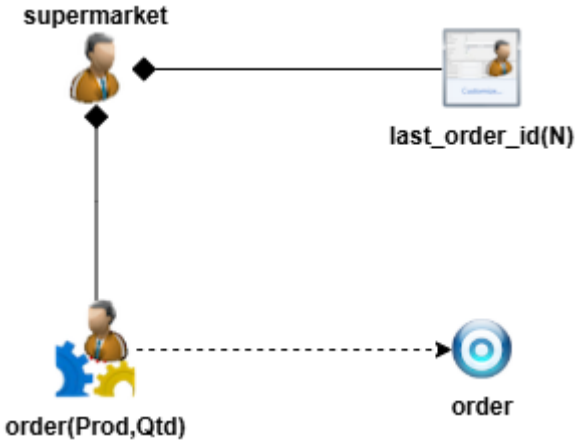

Figura 8: Diagrama de agente de supermarket

Diagrama tareas.

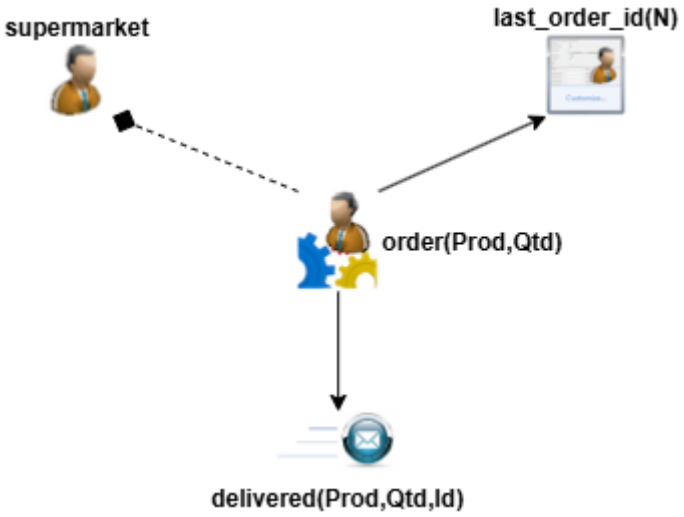

Figura 9: Diagrama de tareas de supermarket

Los Diagramas presentados a continuación son sencillos de entender. Posee una creencia inicial **last_order_id(N)** que representa el último número de pedido y un plan **+!order(Prod, Qtd)[source(Ag)]** que satisface el objetivo de procesar un pedido.
El evento desencadenante de este plan tiene tres variables: el nombre del producto (Prod), la cantidad (Qtd) y el agente que desencadenó el evento.

Cuando se ejecuta, el plan debe consultar la base de creencias para conocer el último número de orden e incrementar ese número, actualizando la base de creencias. Este recoge el pedido y entrega el producto, y debe enviar un mensaje al cliente **delivered(Product, Qtd, OrderId)** diciéndole que el producto ha sido entregado junto con la cantidad y el número de orden del pedido.

Robot enfermera.

La función del agente robot será la de proporcionar las pautas de medicación que debe tomar el owner. Puede ser que a veces owner tome o no medicaciones, entonces debe verificar que sea correcto, y en caso contrario, entregarle la medicación. Además, contará con energía limitada, que deberá recargarse en un cargador compartido (si está ocupado espera).
Las Figuras 10 y 11 representan los diagramas de agente y tareas respectivamente.

Diagrama agente.

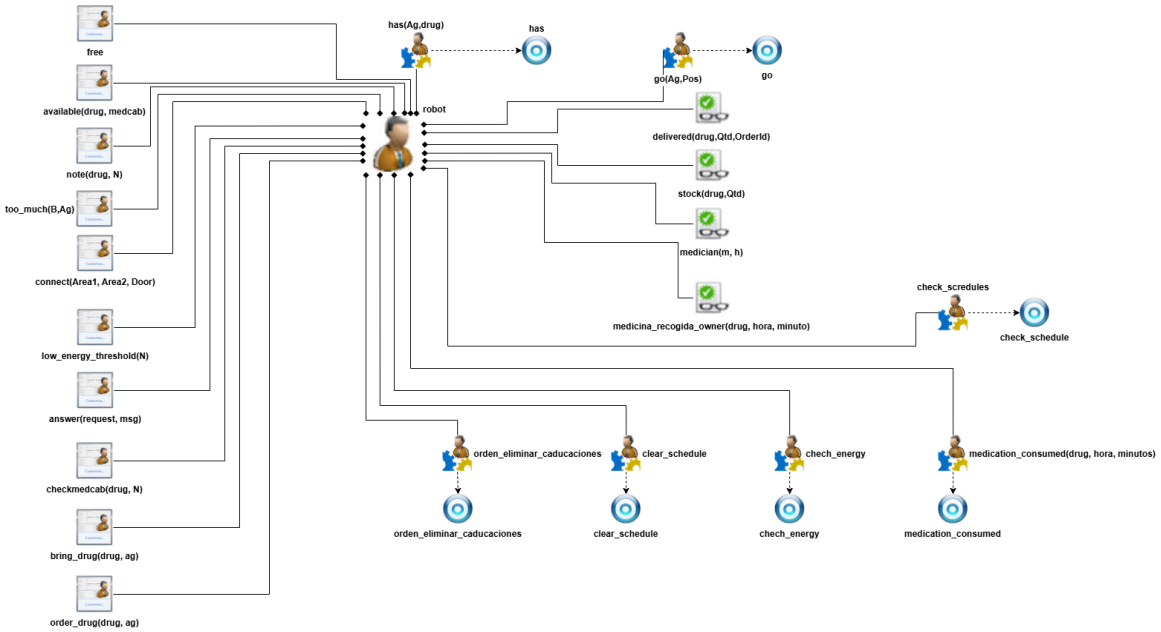

Figura 10: Diagrama de agente de robot enfermera

Diagrama tareas.

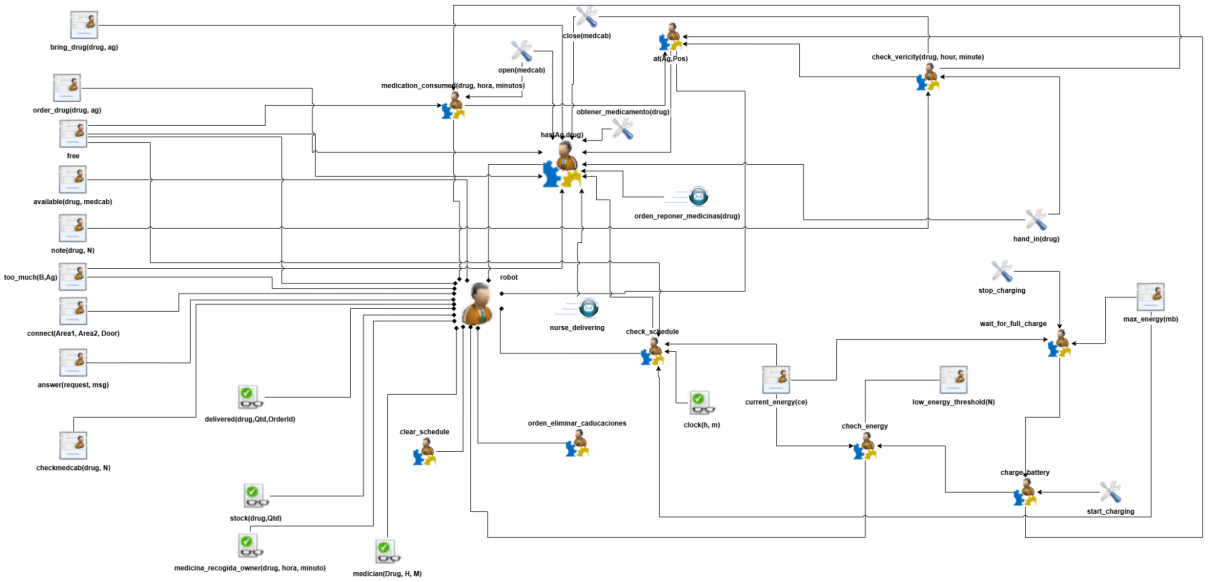

Figura 11: Diagrama de tareas de robot enfermera

El agente robot doméstico inicialmente recibirá las pautas de medicación de owner (**medician(Drug, Hour, Minutes)**) que las almacenará como creencias. Además, tiene como objetivos iniciales **check_energy** y **check_schedule**. La tarea **check_energy** se encarga de, si está libre el robot enfermera y la batería que tiene actualmente (creencia **current_energy(CE)**) es inferior al umbral **low_energy_threshold**, y siempre que robot no esté cargando, ejecuta el plan **charge_batery**, cuya función es moverse al cargador. Esto se hace gracias a la tarea **at** que llama a la función interna **move_towards(P)** para desplazarse. Después de llegar al cargador, llama a la función interna **start_charging** para comenzar a cargar. A no ser de que sea interrumpido, come se puede ver en la Figura 11, hará la tarea **wait_for_full_charge**, donde cargará hasta el máximo de batería (percepción **max_energy(me)**), ejecutando **stop_charging** una vez cargado al máximo, mandandole un mensaje a enfermera para que añada la ciencia **auxiliar_cargado**, en caso de que estuviera esperando.
En caso de que se quede sin energía el robot doméstico, se envía un aviso al auxiliar **robot_needs_energy** para que lo recargue.
El objetivo **check_schedule**, comprueba si existe alguna pauta que haya que entregar, siempre comprobando que haya energía suficiente para entregar, donde si se cumplen dichas condiciones, se invoca la tarea has, como se muestra en la Figura 11, invocando al objetivo **has**. Si no quedan de esos medicamentos (**order_drug(Ag)**), enviará un pedido a supermarket (**order(Drug, N)**), donde lo recogerá en la zona de entrega, regresa al cajón de medicamentos y repone. Una vez repuestos, envía un mensaje a owner de que va ir entregarle medicina (**nurse_delivering**), abre el cajón (**open(medCab)**, obtiene un medicamento (**obtener_medicamento(Drug)**), cierra la nevera (**close(medcab)**), se dirige al owner con at y le entrega la medicina(**hand_in(drug)**). En caso de no tener que reponer ya que quedan existencias, omite el paso de reponer y entrega directamente, como se explicó anteriormente.

Si owner consumió una medicina, robot enfermera debe ir a comprobarlo, con la tarea **medication_consumed(Drug, Hour, Minutes)**, yendo al cajón de medicamentos y comprobando gracias al objetivo **check_vericity(Drug, Hour, Minutes)**, tal como se relacionan en la Figura 11. Al abrir el cajón, el entorno añade una percepción **stock(Drug, N)** que es el número de medicamentos que hay, y lo que hacemos es compararlo con la creencia **note(Drug, N)**, si ha habido cambios o no, y en caso de que sí, robot cogerá la medicina, actualizará sus notas y la entregará a owner. En caso contrario, solo actualiza su creencia **note(Drug, N)**.
Por último, en caso de que la robot enfermera se quede sin energía y auxiliar le comunique que le va entregar la medicina, se moverá a owner y le entregará la medicación, donde irá luego a recargar gracias a **check_energy**.

Robot auxiliar

El agente auxiliar tendrá la funcionalidad de traer más medicación de supermarket, en caso de que esté caducada o no quede. 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 12 y 13 representan su diagrama de agente y tareas respectivamente.

Diagrama agente.

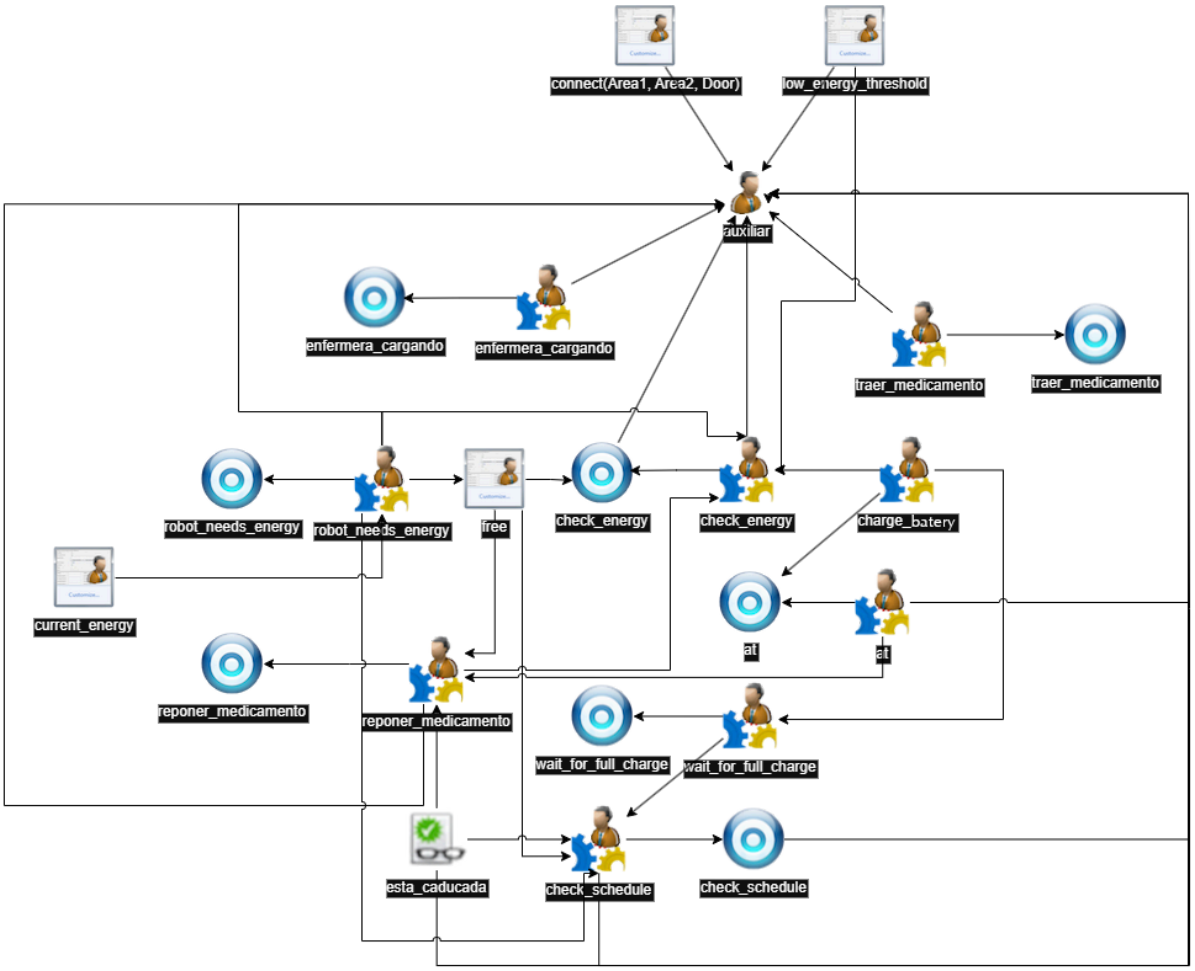

Figura 11: Diagrama de agente de auxiliar

Diagrama tareas.

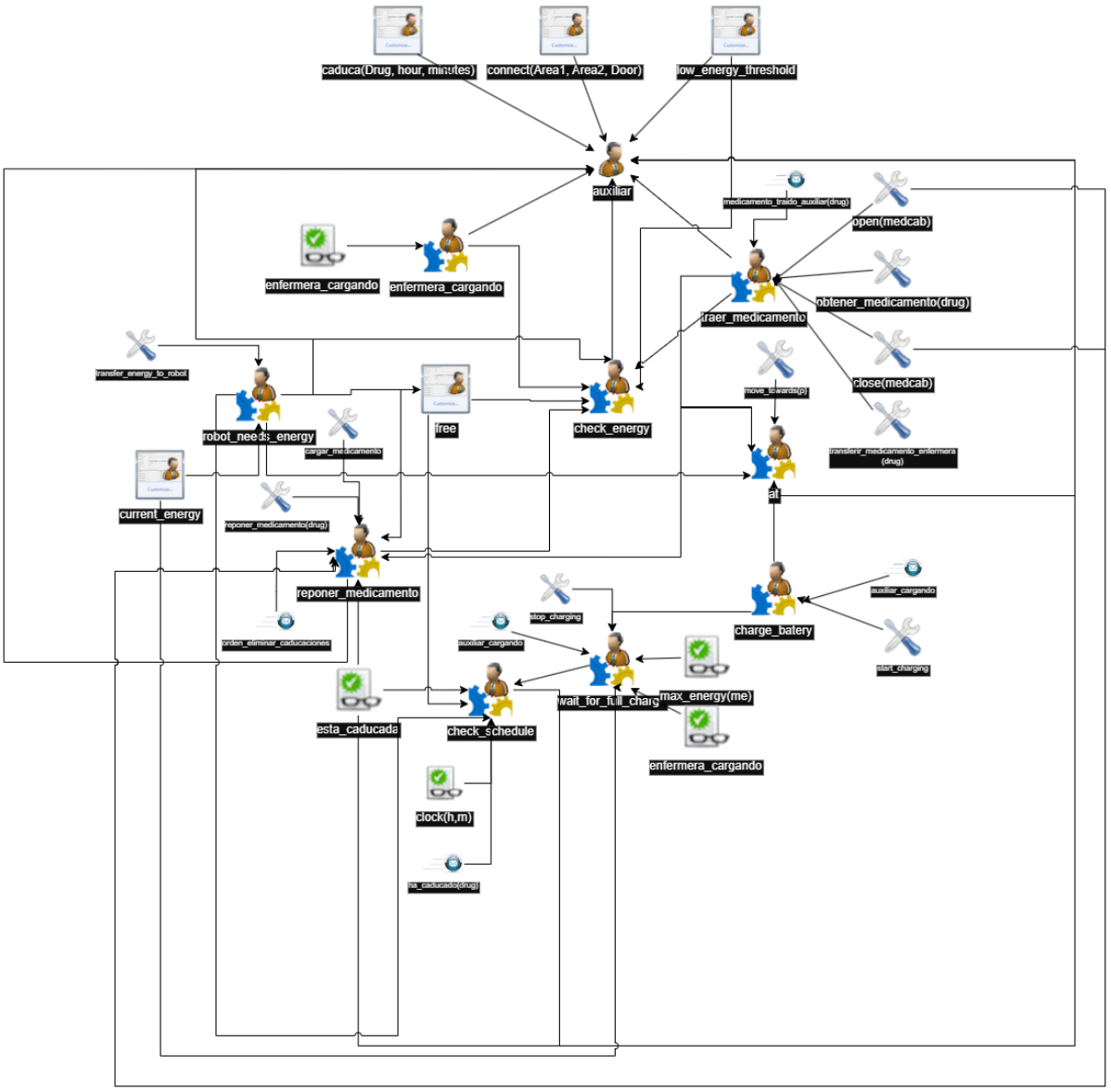

Figura 12: Diagrama de tareas de auxiliar

Empezaremos definiendo sus creencias iniciales:

- **connect(Area1, Area2, Door)**: Creencia para saber la distribución del entorno.
- **caduca(Drug, hour, minutes)**: Nos indica cuánto tiempo le queda a la medicina para que caduque y sea inservible.
- **low_energy_threshold**: Umbral de referencia para la batería de auxiliar, donde si no está realizando nada y la batería es inferior al umbral, vaya a recargar al cargador.
- **free**: Indica si el auxiliar está libre o ocupado.

El auxiliar tendrá como planes iniciales **check_energy** y **check_schedule**, que satisfacen los objetivos **check_energy** y **check_schedule**, como se ve en la Figura 11. La tarea **check_energy** es la misma que se describe en el agente robot enfermera, donde se va a

recargar en caso necesario (**charge_batery**), bien cargando hasta el final (**wait_for_full_charge**) o si es interrumpido para realizar una tarea prioritaria.
La tarea **check_schedule**, como se observa en la Figura 10, invoca el objetivo **check_schedule**, y este, gracias a la percepción **clock(H,M)** del entorno, comprueba si existe una/varias medicación/es que estén caducadas. En el caso de encontrar alguna, como se muestra en la Figura 12, informa a los agentes robot y owner de que la medicina está caducada (**ha_caducado(Drug)**), y se añade a sí mismo la creencia **esta_caducada(Drug)**, ejecutando la tarea **reponer_medicamento** que invoca al objetivo **reponer_medicamento** como muestra la Figura 11, donde se mueve al cajón de medicamentos, abre el cajón de medicamentos (**open(medcab)**), quita los medicamentos de la nevera, obtiene más pidiendo al supermarket, que los va a recoger en el punto de entrega y los añade a la nevera(**reponer_medicamentos(Drug)**), cerrándola (**close(medcab)**), enviando mensajes a owner y robot informando de que se eliminó la medicina caducada (**orden_eliminar_caducaciones**). Finalmente comprueba si tiene que ir a cargar.
Existe la posibilidad de que robot se quede sin batería, en cuyo caso, se ejecuta **robot_needs_energy** para suministrarle (moviéndose a robot con **at**) energía al robot, siempre y cuando le quede a auxiliar energía suficiente para cargar de nuevo, transfiriendo con **transfer_energy_to_robot**. Una vez hecho, comprueba si hay que cargar o reponer medicinas caducadas.

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.