

e_Puck vs COVID-19

Yamil Mateo Rodríguez y José Antonio Antona Díaz

Abril 2020



Figura 1: e_Puck

Índice

1. Enunciado y Objetivo del Trabajo	3
2. Introducción a la idea	3
3. Mapa	4
4. Arquitecturas Basadas en el Comportamiento	5
5. Arquitectura Subsunción	6
5.1. Niveles de Competencia, elementos de supresión y de inhibición	6
5.1.1. Navigate	7
5.1.2. NavigateGym	10
5.1.3. GoLOAD	11
5.1.4. Forage	13
5.1.5. Avoid	14
6. Arquitectura Motor Schemas	15
7. Comparativa de las distintas arquitecturas	17
7.1. Niveles de Competencia y elementos de inhibición	17
7.1.1. Navigate	17
7.1.2. NavigateGym	18
7.1.3. GoLOAD	18
7.1.4. Forage	19
7.1.5. Avoid	19
8. Gráficas	20
9. Conclusión	22
10. Problemas encontrados	22
11. Posibles implementaciones futuras	22

1. Enunciado y Objetivo del Trabajo

Enunciado: Implementar en el simulador IRSIM una arquitectura basada en el comportamiento para la resolución de una tarea a decidir libremente por el grupo. Se deberá implementar una arquitectura subsunción y una motor schemas, para resolver el mismo problema, y realizar una comparativa entre el funcionamiento de ambas.

Objetivo: Se orientará el trabajo al análisis y la presentación de resultados de arquitecturas basadas en el comportamiento, *Subsunción* y *Motor Schemas*. Los resultados se obtendrán a partir de experimentos realizados en el simulador IRSIM. Se evaluará la efectividad y el desempeño conseguido, obteniendo conclusiones en lo que respecta a la conducta deseada del robot y comparándola con la que manifiesta en su desempeño. Los parámetros que se analizan para esta evaluación están relacionados con la evitación de obstáculos, optimización de las trayectorias y el logro de los objetivos. Estos parámetros se van a caracterizar a través de una conducta específica que se va a detallar a continuación.

2. Introducción a la idea

El Gobierno de nuestro país, en la reunión extraordinaria del Consejo de Ministros del 14 de marzo de 2020, aprobó el Real Decreto 463/2020, por el que se declara el estado de alarma para la gestión de la crisis sanitaria ocasionada por el COVID-19. La declaración del estado de alarma limita la libertad de circulación de las personas por espacios públicos, por lo que el e_Puck al igual que el resto de las personas, se quedará confinado en casa. En suma, el e_Puck seguirá varios consejos de la Organización Mundial de la Salud (OMS) para evitar el contagio. Dentro de su casa, con suelo de color negro, realizará distintas actividades como hacer ejercicio, que se referenciará con la luz amarilla. Además, tendrá que satisfacer las necesidades alimentarias de sus e_Puckit@s e ir a por sumistros al supermercado, que se encuentra situado en el lado opuesto a su casa con un suelo gris y una luz azul. Durante el camino se encontrará con diversas personas y obstáculos, con las que tendrá que mantener la distancia de seguridad recomendada. Consecuentemente, al volver con la compra a su casa, si lo necesita, se lavará las manos. Dicha necesidad está marcada por la batería de la luz roja.

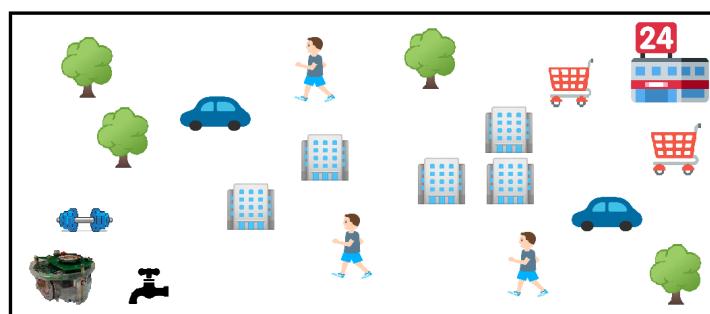


Figura 2: Entorno

3. Mapa

A continuación, se muestra una figura del concepto de mundo descrito en la introducción.

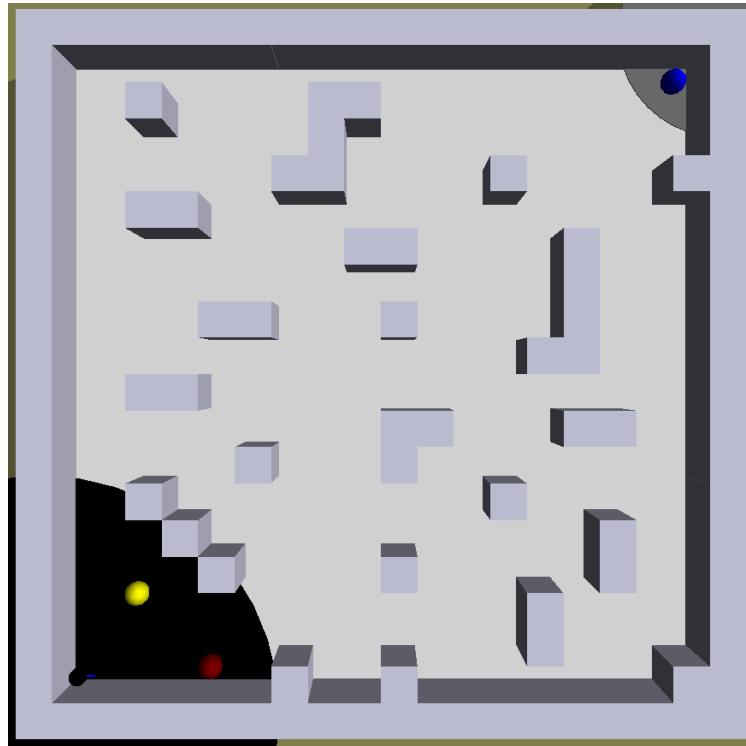


Figura 3: Mapa

- *Suelo negro*: hace referencia a la casa del e_Puck.
- *Luz roja*: hace referencia al puesto de lavado.
- *Luz amarilla*: hace referencia al puesto de gimnasio.
- *Suelo gris*: hace referencia al supermercado.
- *Luz azul*: hace referencia a un foco de atracción para el suelo gris.

4. Arquitecturas Basadas en el Comportamiento

Las arquitecturas basadas en el comportamiento constituyen una evolución de las implementaciones reactivas, las cuales se basan en modelos biológicos y la psicología de los seres vivos para explicar el comportamiento observado en distintos organismos. Para dichas arquitecturas se implementan estrategias basadas en dos fases, percepción del entorno y actuación. Los movimientos del robot se guían a partir de la información que recogen los sensores en cada instante. Es decir, las acciones del robot se basan en un acoplamiento directo entre sensores y actuadores, mediante bucles rápidos de realimentación.

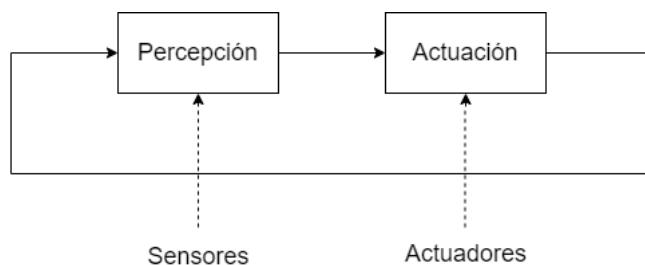


Figura 4: Arquitectura Reactiva

Los comportamientos permiten establecer un modelo descentralizado de inteligencia, con el cual se puede llevar a cabo un comportamiento específico a partir de subcomportamientos básicos. Para llevar a la acción nuestro comportamiento global deseado, se define el concepto de *Coordinador*, como un sistema capaz de interactuar con el entorno y los diversos estímulos que se detecten, es decir, para conseguir información del entorno y actuar en consecuencia.

El *Coordinador* se encargará de interactuar con el entorno a través de una tabla de prioridades, que dispondrá de tantas columnas como subcomportamientos contenga la arquitectura y varias columnas, que van a variar en función de la arquitectura elegida: *Subsunción* o *Motor Schemas*. Una de las columnas estará reservada para la activación de un flag. Dichos flags se activarán o no como consecuencia de la información recibida por los sensores. Cada subcomportamiento tendrá una prioridad asignada, con la que el *Coordinador* trabajará a la hora de decidir qué subcomportamiento o cuáles deben imponerse sobre los demás.

5. Arquitectura Subsunción

Las arquitecturas de *Subsunción* se construyen añadiendo capas de subcomportamientos unas sobre otras. Dichos subcomportamientos se denominan niveles de competencia o capas. Cada nivel de competencia se ejecutará de manera paralela al resto y se encargará de llevar a cabo una tarea específica, que junto con el resto de subcomportamientos, llevarán a cabo el comportamiento global deseado. Además, dichos niveles de competencia van a interaccionar entre sí modificando las entradas y salidas a través de dos elementos de control, inhibidores y supresores. Cabe destacar, que esta arquitectura se va a codificar en **Iri1**.

Cada nivel de competencia consiste en un conjunto de módulos asíncronos conectados entre sí. En esta arquitectura entra en juego la tabla de coordinación, un elemento muy importante que va a determinar el efecto de cada comportamiento en el robot. Para la arquitectura de subsunción estudiada en esta asignatura, la tabla de coordinación tiene esta forma:

	Velocidad lineal rueda izquierda	Velocidad lineal rueda derecha	Flag de activación
Comportamiento	{-1000, ... ,1000}	{-1000, ... ,1000}	1/0

Figura 5: Tabla coordinación

5.1. Niveles de Competencia, elementos de supresión y de inhibición

A continuación, se va a detallar como se usan los niveles de competencia implementados y las relaciones entre ellos, a través de inhibidores y supresores que se muestran en el siguiente esquema:

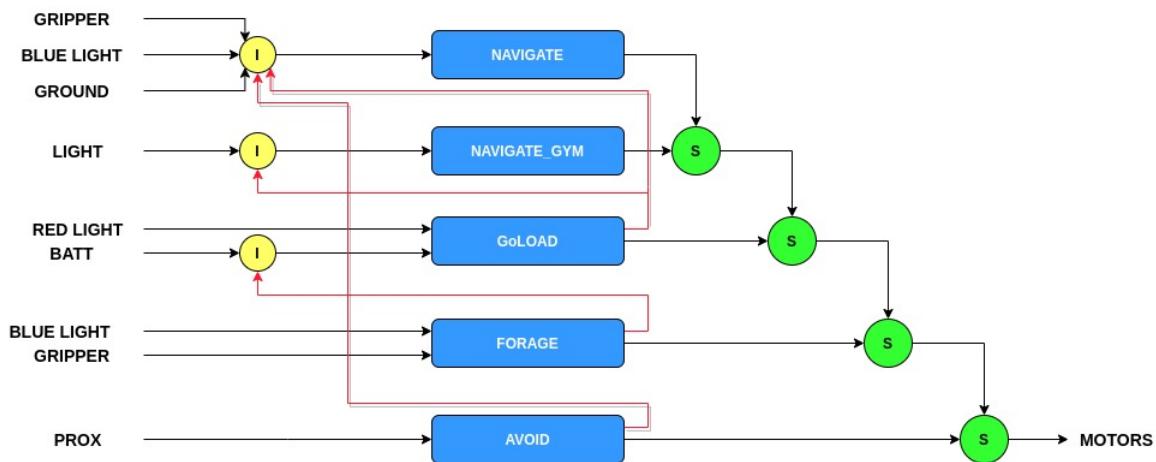


Figura 6: Niveles de Competencia subsunción

La función principal de los supresores es proporcionar una escala de prioridad a los diferentes niveles de competencia, dicha prioridad permitirá llenar las tablas de coordinación en cada "*Simulation step*", tras la ejecución de cada uno de los comportamientos y consecuente alteración de variables. En este momento, entra en acción la función real del supresor, ejecutándose dicha tabla en un orden de prioridad fijado previamente, del comportamiento más prioritario al menos prioritario. Con dicho orden se consigue que una vez se detecte activo uno de los niveles de competencia, el e_Puck reaccionará a lo impuesto ese nivel. En nuestro proyecto los niveles de prioridad se ordenan de la siguiente manera:



Figura 7: Prioridad de estados

En cuanto a los inhibidores, su función principal será omitir las condiciones de las que un determinado comportamiento depende, de manera que no se cumpliran dichas condiciones y no se ejecutará dicho comportamiento. En nuestro caso tenemos tres inhibidores: "*Forage to GoWash*", "*GoWash to Navigate*", "*Avoid to Navigate*".

5.1.1. Navigate

Este comportamiento es el menos prioritario en la arquitectura de subsunción realizada. "*Navigate*", va a hacer que el robot se comporte de manera distinta en función de las siguientes condiciones. Se distinguen dos subcomportamientos.

- **Navegar hacia luz azul:** este subcomportamiento orienta al robot hacia la luz azul si no esta yendo hacia ella, para conseguir la corrección de la orientación, se realiza un giro en sentido horario, siempre que la parte izquierda del robot perciba más luz que la derecha y uno en sentido antihorario en caso de que los sensores de la parte derecha del robot perciban más luz. Sin embargo, depende de otros sensores, inhibidores y variables:
 - *Sensores de luz azul:* para apuntar correctamente hacia la luz se hace uso de los sensores de luz azul, concretamente de los sensores 0 y 7, siempre que alguno de su valores sea nulo se va a corregir la posición del robot, constituyendo así la primera condición para este subcomportamiento.
 - *Sensores de suelo:* estos sensores logran que el robot cuando esté en casa no quiera irse a por la luz azul tras dejar un paquete, o que una vez que salga de la casa o suelo negro no intente volver a menos que necesite cargar su batería. Por lo tanto, cuando el robot no esté en suelo blanco, no entrará en este subcomportamiento.

- *Sensores de suelo con memoria*: estos sensores logran que no haya un conflicto sobre la orientación del robot una vez este entra en el estado de Forage, en este estado los sensores de suelo con memoria tienen valor unidad, al haber pisado suelo gris. Aunque "Forage" tiene prioridad respecto a "Navigate" de no ser por esta condición una vez el robot satisfaciera la condición de orientación contraria a la luz azul establecida por "Forage", entraría en conflicto directo con este subcomportamiento que lo orientaría hacia la luz en el siguiente '*Simulation step*' quedándose parado.
- *Inhibidor 'Wash to Navigate'*: este inhibidor solventa un conflicto similar al que solucionan los sensores de suelo con memoria, pero en este caso ese conflicto se genera por la orientación que establece el comportamiento "GoLoad Wash" orientándose hacia la luz roja cuando no tiene batería. Si no fuera por este inhibidor, el robot se quedaría parado en situaciones en las que estuviera apuntando hacia la luz roja pero no hacia la luz azul. Por lo tanto, siempre que el inhibidor esté activado, este subcomportamiento no funcionará.
- *Variable 'justStopped'*: la principal función de esta variable es evitar que el robot entre en un bucle en el que intente avanzar hacia la luz azul y se choque con la misma pared repetidamente. Por lo tanto, cuando se choca y entra en "Avoid" además de orientarse en contra del obstáculo, el valor de 'justStopped' se incrementa, con dicho incremento se consigue que no intente ir hacia la luz directamente debido a que se va a volver a chocar. Consecuentemente, es necesario que el valor de esta variable sea 0 para que entre en este subcomportamiento.

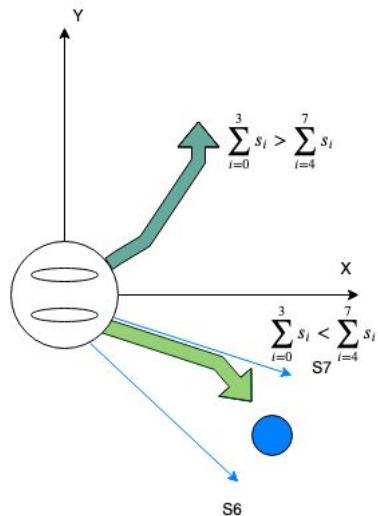


Figura 8: Navigate to blue light situation

- **Navegar recto:** este subcomportamiento hace que el robot continue en la dirección a la que apunta, proporcionando la misma velocidad en ambas ruedas. Siempre que no se cumpla alguna de las condiciones necesarias en el anterior subcomportamiento se cargarán en la tabla de coordinación los valores impuestos por este. Además, existe cierta dependencia de los siguientes parámetros:

- *Inhibidor 'Avoid to Navigate'*: este inhibidor se encarga de limitar el decremento de la variable '*justStopped*', ya que en el procesamiento que realiza el método *ExecuteBehaviours()* se recorre todo el controlador viéndose afectadas las variables comunes entre comportamientos al modificarse en varios, de hecho, es en este procesamiento donde se actualiza el valor de los inhibidores. Por lo tanto, de no ser por este inhibidor y a la vista del controlador si en un '*Simulation step*' se incrementará la variable en "*Avoid*" porque ha habido un choque, se decrementaría tras el procesamiento de "*Navigate*".
- *Variable '*justStopped*'*: en este subcomportamiento se decrementa el valor de la variable. El motivo de este decremento es permitir que tras salir del bucle de choque contra una pared vuelva a ir a la luz. Cabe destacar, que el valor de '*justStopped*' va desde 0 hasta N.

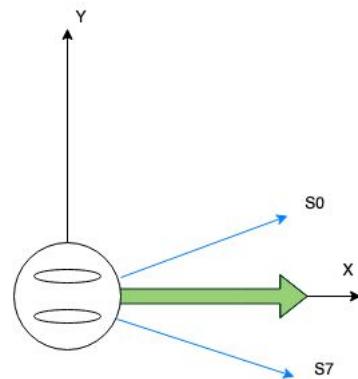


Figura 9: Navigate straight situation

5.1.2. NavigateGym

En este comportamiento, cuya prioridad sucede a "Navigate" básico, se hace uso de los sensores de luz para hacer al e_Puck girar en torno a la luz amarilla. La implementación para lograr esto se detalla en [1]. La luz amarilla se enciende y se apaga intermitentemente con una frecuencia determinada y ajustable. Los factores condicionantes de este nivel de competencia son:

- *Inhibidor 'Wash to Navigate'*: este inhibidor cancela el comportamiento cuando se activa, resuelve el conflicto donde el robot, al no tener batería avanza hacia la luz roja pero se encuentra la luz amarilla de camino. En este caso, se quiere un avance directo, que se consigue con la orientación hacia la luz roja en "Wash" y con el progreso directo que se realiza en "Navigate" en estas condiciones, por lo tanto, debido a que la prioridad es mayor frente a "Navigate", hay que limitar un posible movimiento si se encontrara con la luz amarilla para que no se desviara del camino más corto a la luz roja.
- *Valor 'Navigate_Gym_Threshold'*: este valor marca el umbral de proximidad por el cual los sensores de la parte derecha o izquierda entrará en acción para comenzar la acción de giro. Por debajo de este umbral el nivel de competencia no cargará nada en la tabla de coordinación.

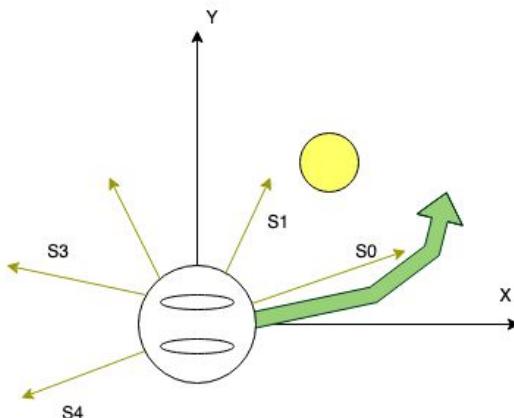


Figura 10: Gym situation

5.1.3. GoLOAD

En este comportamiento, cuya prioridad sucede a "NavigateGym", se hace uso tanto de los sensores de luz roja como de la batería asociada a dicha luz. El ciclo de funciones de este comportamiento se basa en una carga y descarga de la batería asociada a la luz roja. Respecto a la carga, se realiza mediante unos sensores de carga en el robot y un coeficiente de carga el cual se va sumando al valor de la batería en cada 'Simulation step' una vez esos sensores se encuentran con la luz roja. Respecto a la descarga, se tiene un coeficiente de descarga que se va restando a la batería a medida que no detectan la luz. Los factores condicionantes de este comportamiento son los siguientes:

- *Inhibidor 'Forage to Wash'*: este inhibidor cancela el comportamiento cuando se activa, resuelve el conflicto que se crea cuando el robot cumple las condiciones de "Forage", en este caso si se descargara la batería durante dicho comportamiento, el robot no dejaría de ir en sentido contrario a la luz azul para ir en sentido a la luz roja. Este concepto es análogo al conflicto que resuelve el inhibidor 'Wash to Navigate' pero en este caso dando prioridad al comportamiento que va en contra de la luz azul, "Forage".
- *Valor 'Battery_Wash_Threshold'*: este valor marca el umbral de máxima descarga permitida. Cuando la batería está por debajo de este umbral, el comportamiento se activa y se va hacia la luz roja, cargando la batería cuando se encuentre a una distancia menor o igual que la de visión del sensor de carga . En caso contrario no se realiza ninguna acción.
- *Sensores de luz roja*: para apuntar correctamente hacia la luz se hace uso de los sensores de luz roja, concretamente de los sensores 0 y 7, siempre que alguno de sus valores sea nulo se va a corregir la posición del robot orientando al robot hacia la luz roja si no está yendo hacia ella, para conseguir la corrección de la orientación se realiza un giro en sentido antihorario, siempre que la parte izquierda del robot perciba más luz que la derecha y uno en sentido horario en caso de que los sensores de la parte derecha del robot perciban más luz.

Cuando se cumplen las condiciones para este comportamiento, se activa a nivel bajo la señal "Wash to Navigate" que inhibe los comportamientos "Navigate" y "NavigateGym", se activa el flag y se cargan los valores lineales de la velocidad de cada rueda en la tabla de coordinación.

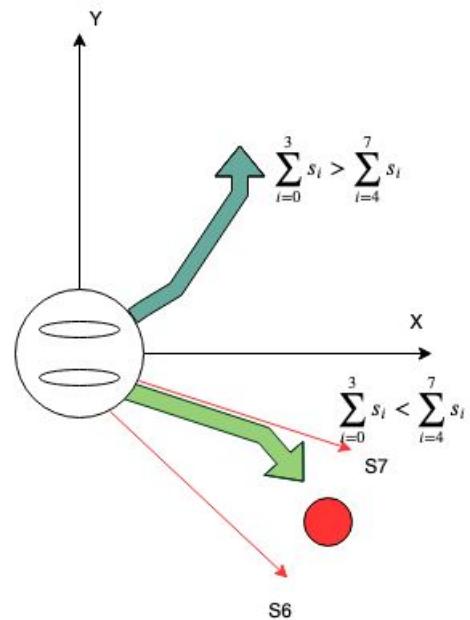


Figura 11: Load situation

5.1.4. Forage

En este comportamiento, cuya prioridad sucede a "GoLOAD", se hace uso de los sensores de luz azul. Su función principal se encarga de orientar en contra de la luz azul que se posiciona encima del suelo gris como referencia sensorial a distancia. Los factores condicionantes de este comportamiento son los siguientes:

- *Sensor de suelo con memoria*: este inhibidor cancela el comportamiento cuando se activa, resuelve el conflicto que se crea cuando el robot cumple las condiciones de "Forage", en este caso si se descargara la batería durante dicho comportamiento, el robot no dejaría de ir en sentido contrario a la luz azul para ir en sentido a la luz roja. Este concepto es análogo al conflicto que resuelve el inhibidor 'Wash to Navigate' pero en este caso dando prioridad al comportamiento que va en contra de la luz azul, "Forage".
- *Sensores de luz azul*: para orientar al robot en contra de la luz azul si está yendo hacia ella se hace uso de los sensores de luz azul, concretamente los sensores 3 y 4, siempre que alguno de su valores sea nulo se va a corregir la posición del robot, para orientarlo se realiza un giro en sentido antihorario, siempre que la parte izquierda del robot perciba más luz que la derecha y uno en sentido horario en caso de que los sensores de la parte derecha del robot perciban más luz.

Cuando se cumplen las condiciones para este comportamiento, se activa a nivel bajo la señal "Forage to Wash" que inhibe "GoLOAD", se activa el flag y se cargan los valores lineales de la velocidad de cada rueda en la tabla de coordinación.

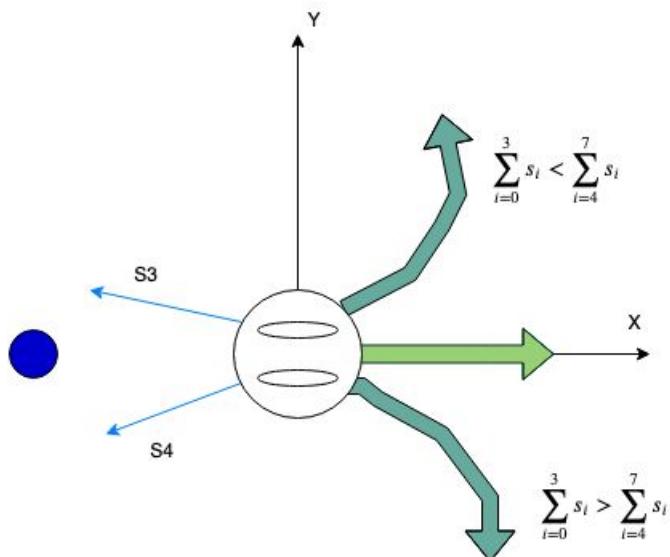


Figura 12: Forage situation

5.1.5. Avoid

Este comportamiento es el nivel de competencia de mayor prioridad. Su función principal es evitar el choque con las paredes y obstáculos del mapa. Para ello se comprueba si la cercanía excede el umbral de proximidad '*Proximity_Threshold*', si lo excede, se calcula un ángulo de repulsión ϕ y velocidad de repulsión, como el módulo de la velocidad de cada rueda, a partir de las siguientes expresiones:

$$\left\{ \phi = \text{arc tg} \frac{\sum_{i=0}^7 \sin(\text{proxDirection}_i) * \text{prox}_i}{\sum_{i=0}^7 \cos(\text{proxDirection}_i) * \text{prox}_i} \right\} \quad (1)$$

$$\left\{ r = \sqrt{(\text{SPEED} * \cos \frac{\phi}{2} - \phi)^2 + (\text{SPEED} * \cos \frac{\phi}{2} + \phi)^2} \right\} \quad (2)$$

$$r = 1 \text{ when } \text{prox} > \text{Proximity_Threshold}, 0 \text{ otherwise} \quad (3)$$

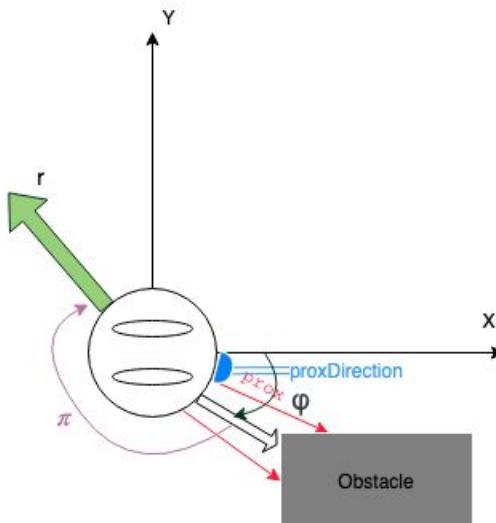


Figura 13: Avoid situation

Cabe destacar, que cada vez que se supere el umbral de proximidad se aumentará el valor de la variable '*justStopped*' para contabilizar las veces que se ha chocado repetidamente sin avanzar. Cuando se cumplen las condiciones para este comportamiento y el robot evita un obstáculo, se activa a nivel bajo la señal '*Avoid to Navigation*' que inhibe el decremento erróneo de la variable '*justStopped*' en el nivel de competencia "*Navigate*", se activa el flag y se cargan los valores lineales de la velocidad de cada rueda en la tabla de coordinación.

6. Arquitectura Motor Schemas

La arquitectura *Motor Schemas* se concibe como una arquitectura basada en el comportamiento ya que se pretende conseguir un comportamiento específico a través de múltiples procesos, que reaccionan a estímulos del entorno gracias a diversos sensores que interactúan con dicho entorno. Los sensores permiten a cada proceso responder al entorno y contribuir de forma independiente a la acción global que se pretende conseguir, es decir, el comportamiento específico para el que ha sido diseñado.

No necesariamente se tienen que procesar todos los datos que nos proporcionan los sensores, solo aquellos que van a contribuir de manera significativa a la tarea que deseamos desempeñar en cada instante, para llevar a cabo la acción global.

Los esquemas motores deben conducir al robot a interactuar con el entorno en el que se encuentre y satisfacer el comportamiento para el que ha sido diseñado. Para ello, se implementarán niveles de competencia, estableciendo una jerarquía entre ellos. Los niveles de competencia se encargarán de que el robot pueda desplazarse o rotar según las condiciones.

A continuación, se muestran los niveles de competencia que utilizaremos para llevar a cabo nuestro comportamiento:

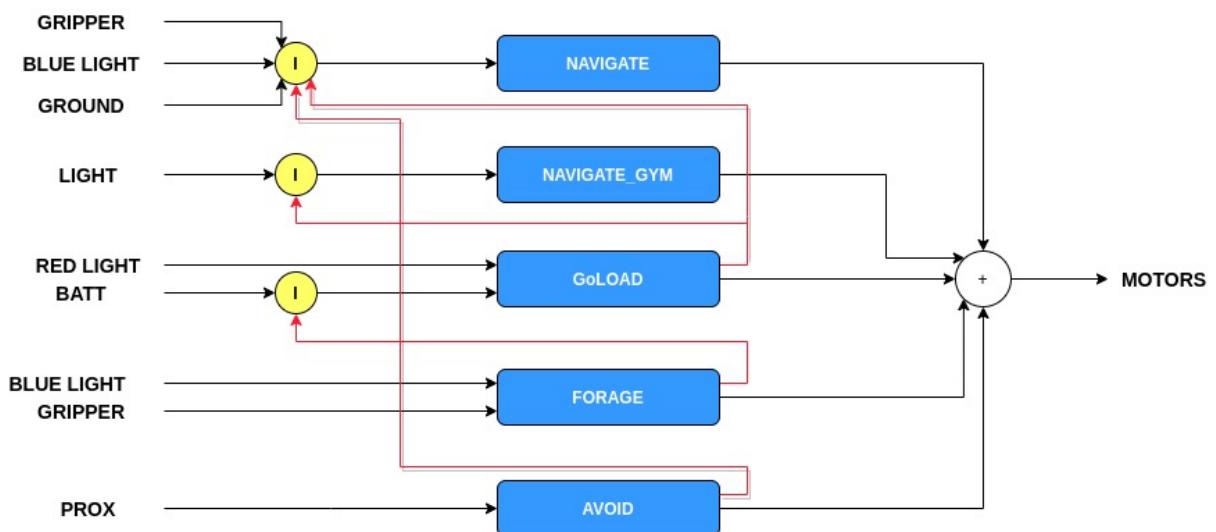


Figura 14: Niveles de Competencia motor schemas

Los múltiples procesos o subcomportamientos que se deben llevar a cabo, se encargarán de proporcionar al *Coordinador* la intensidad y la orientación, campo vectorial, con la que se debe desplazar el robot para satisfacer dicho subcomportamiento. En la arquitectura de esquemas de motores varios niveles de competencia pueden estar activos al mismo tiempo, gracias a un mecanismo de combinación de todos los campos vectoriales activos. La dirección que seguirá el robot se determinará sumando los vectores que se recogerán en una tabla que se explicará más adelante.

El sistema de esquemas de motores se basa, a nivel de software, en la famosa estrategia *Divide y vencerás*. Cada subcomportamiento o proceso se programará como una función independiente en la que el *Coordinador* se encargará de extraer la información necesaria para conocer si dicho proceso debe contribuir o no, a través de *flags* que estarán activados o desactivados. Cabe destacar, que esta arquitectura se va a codificar en **Iri2**.

La función del *Coordinador* se basa en una tabla de coordinación como la descrita anteriormente en el apartado 5 de la página 6, un array bidimensional con tantas filas como niveles de competencia y tres columnas. Con la diferencia de que en las arquitecturas de subsunción se recogen las velocidades de las ruedas y el flag de activación, en la arquitectura de esquemas de motores se trabaja con campos vectoriales, por lo que se recogerán, además del flag de activación, la intensidad con la que se debe contribuir cada comportamiento y la orientación en la que debe dirigirse el robot. A continuación se muestra dicha tabla de activación:

	Ángulo	Módulo	Flag de activación
Comportamiento	$\{0, \dots, 2\pi\}$	$\{0, \dots, 1\}$	1/0

Figura 15: Ejemplo tabla de activación Motor Schemas

7. Comparativa de las distintas arquitecturas

La principal diferencia entre las arquitecturas presentadas a lo largo del proyecto se centra en el Coordinador y en como este coordinador recoge información a través de los subcomportamientos en la tabla de prioridades. Al recoger distinta información la manera en la que se relacionará con los actuadores será diferente ya que tiene que adaptar dicha información a los parámetros pedidos por los actuadores, en nuestro caso, las ruedas.

7.1. Niveles de Competencia y elementos de inhibición

Los conceptos detrás de cada comportamiento son muy similares en ambas arquitecturas. Además, puesto que el objetivo es llevar a cabo el mismo comportamiento en ambas arquitecturas las prioridades asignadas a cada nivel de competencia son las mismas que en la arquitectura de subsunción. De la misma manera, no hay cambios ni los inhibidores ni en sus posiciones: "*Forage to GoWash*", "*GoWash to Navigate*", "*Avoid to Navigate*".

7.1.1. Navigate

La principal diferencia se encuentra en los valores proporcionados al Coordinador, que serán un vector dirección hacia la propia luz azul y el valor máximo de los sensores de luz azul en el caso del subcomportamiento "*Navegar hacia luz azul*". Para el subcomportamiento "*Navegar recto*" se proporcionará un vector de manera que el robot siga en línea recta, en la dirección que apuntaba en el instante inmediatamente anterior. Las condiciones para cada subcomportamiento dentro de "*Navigate*" serán las mismas.

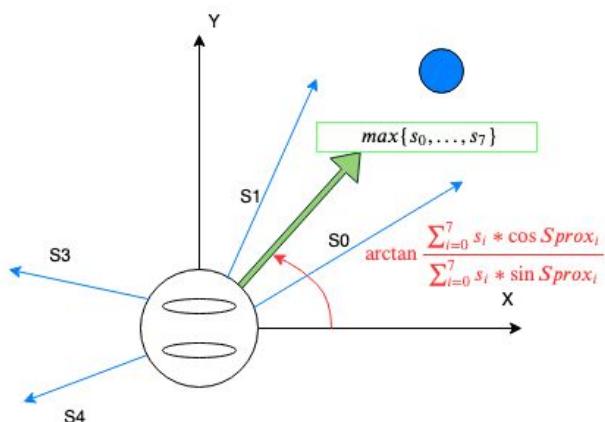


Figura 16: Navigate situation in Motor Schemas

7.1.2. NavigateGym

Para la arquitectura Motor Schemas, "NavigateGym" proporcionará un vector dirección hacia la luz amarilla y el valor máximo de los sensores de luz amarilla. La condición para que se tendrá que cumplir para que se active el flag será que la luz total captada por los sensores de luz amarilla superen el umbral "NAVIGATE_GYM_THRESHOLD" y que el inhibidor "Wash to Navigate" esté activo a nivel alto.

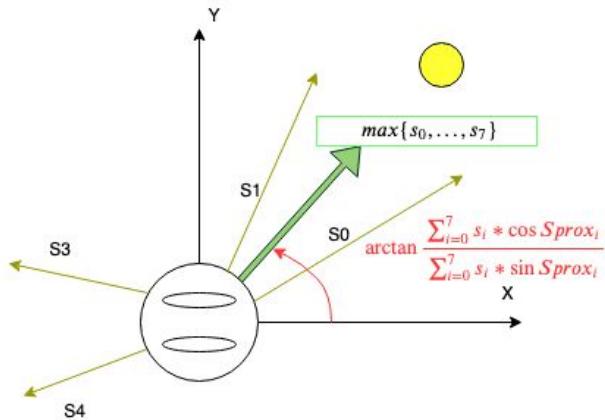


Figura 17: Navigate gym situation in Motor Schemas

7.1.3. GoLOAD

La principal diferencia se encuentra en los valores proporcionados al Coordinador, que serán un vector dirección hacia la luz roja y el valor máximo de los sensores de luz roja. Las condiciones dentro de "GoLOAD" serán las mismas.

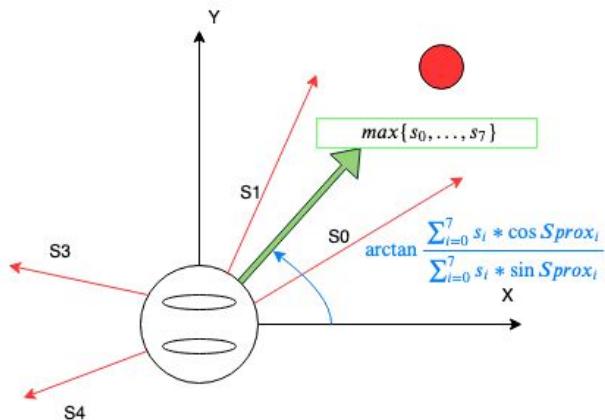


Figura 18: GoLOAD situation in Motor Schemas

7.1.4. Forage

La principal diferencia se encuentra en los valores proporcionados al Coordinador, que serán un vector dirección en contra a la luz azul y el valor máximo de los sensores de la luz azul. La condición dentro de "Forage" que se tendrá que cumplir para que se active el comportamiento será que el e_Puck haya estado en la zona gris activándose el valor del sensor de suelo con memoria y que la variable 'justStopped' sea cero.

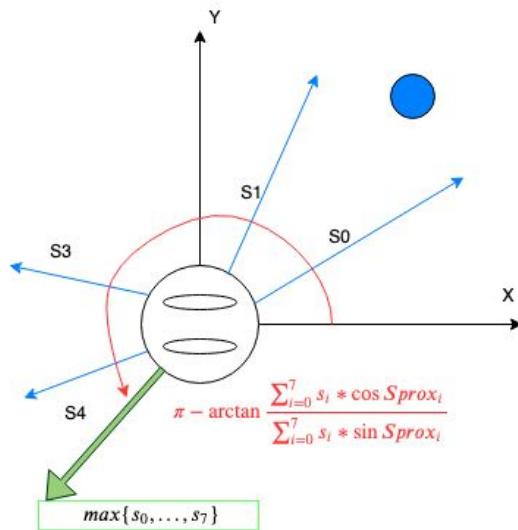


Figura 19: Forage situation in Motor Schemas

7.1.5. Avoid

La principal diferencia se encuentra en los valores proporcionados al Coordinador que serán un vector dirección en contra del obstáculo detectado por los sensores de proximidad y el valor máximo de dichos sensores. Las condiciones dentro de *Avoid* serán las mismas. La explicación gráfica coincide con en el apartado 5.1.5 de la página 14

8. Gráficas

En esta sección se va a mostrar el flujo de activación de ambas arquitecturas:

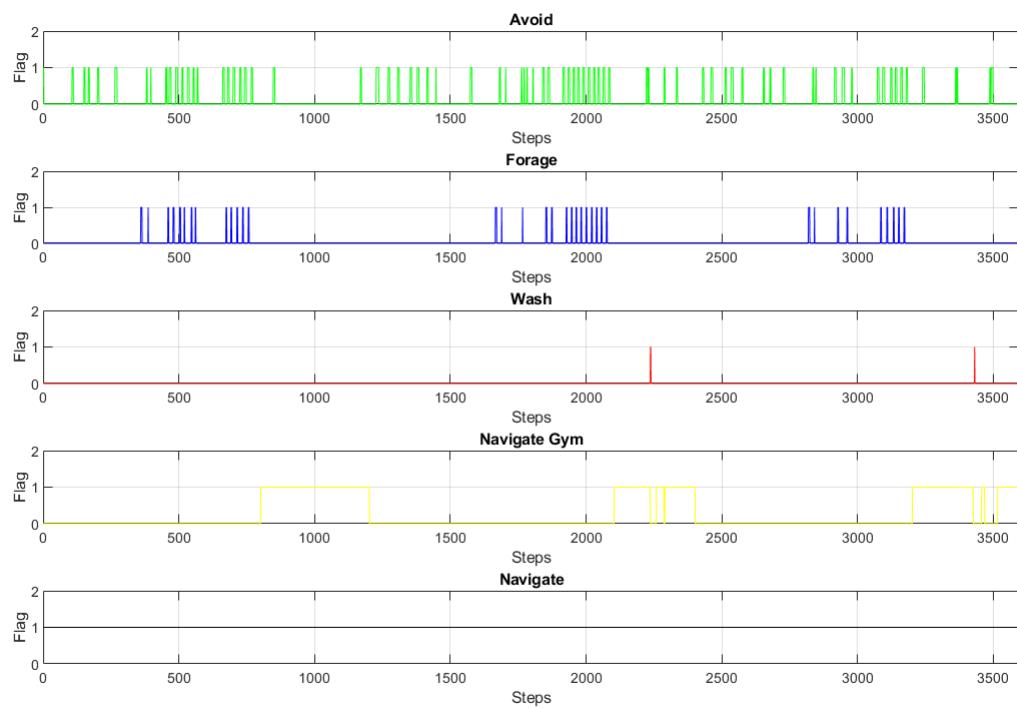


Figura 20: Gráfica de activación de los niveles de competencia en subsunción

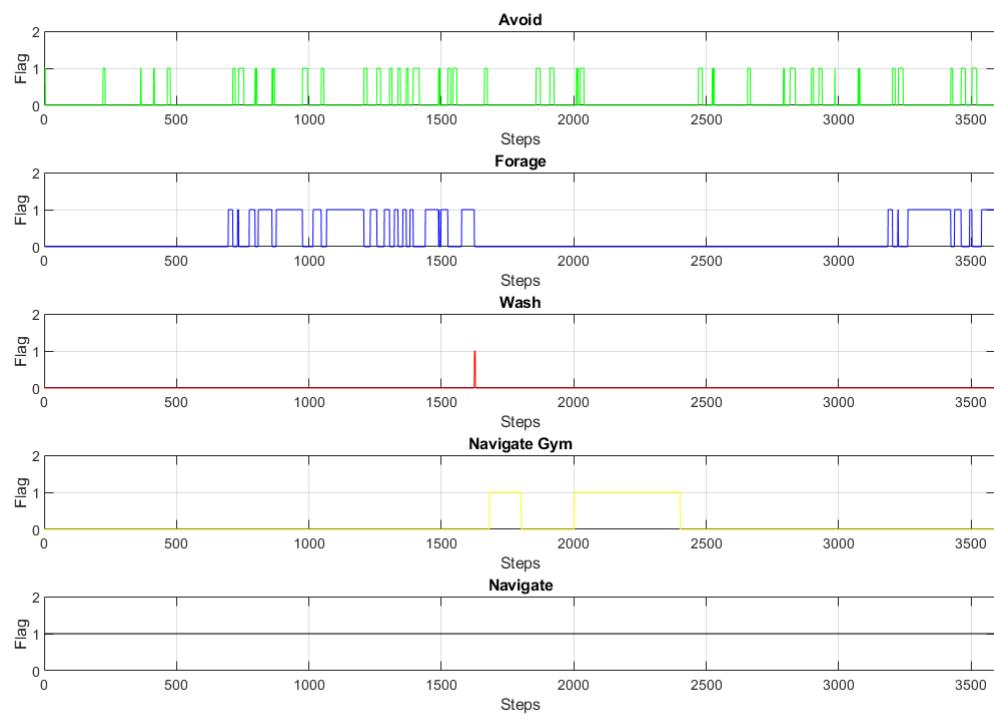


Figura 21: Gráfica de activación de los niveles de competencia en motor schemas

9. Conclusión

Anteriormente, se han planteado dos arquitecturas basadas en el comportamiento, con las cuales los robots dotan de *inteligencia* a la hora de responder a estímulos captados por los sensores que dispone. Además, se han podido ver las diferencias entre dichas arquitecturas a nivel interno para una misma funcionalidad global y la complejidad que requiere implementar la misma funcionalidad en ambas arquitecturas.

Por otro lado, se puede observar que a pesar de intentar simular un comportamiento humano ante la crisis sanitaria que vivimos en España actualmente, un robot nunca podrá asemejarse totalmente a un ser humano ya que existen múltiples factores que "no podemos programar". En nuestro caso podemos ver como cada persona tiene la libre elección de hacer caso a las recomendaciones de la OMS, y cada ser humano podrá hacer más o menos caso en determinados momentos imprevisiblemente, por posibles factores externos que puedan ocurrir. Un robot no dota de dicha posibilidad, ya que ha tenido que ser programado anteriormente para cumplir dicha funcionalidad, por lo que se pierda la imprevisibilidad humana. Además, entran en juego factores sentimentales o emocionales de los que no dispone un robot y que implican un cambio significativo en las reacciones.

10. Problemas encontrados

Los principales problemas que nos hemos encontrado han sido:

- Replicación de los comportamientos en ambas arquitecturas.
- Resolución de conflictos entre atracción y repulsión impuesta por "*Navigate*" y "*Forage*" con la luz azul.
- Resolución de conflictos entre atracción y repulsión impuesta por "*GoLOAD*" y "*Forage*" con las luces roja y azul.
- Control del flujo de modificación de variables globales en *ExecuteBehaviours()*

11. Posibles implementaciones futuras

En línea con nuestra idea de simular un comportamiento humano ante el estado de alarma y las limitaciones que estas implican, se podrían añadir niveles de competencia que nos permitieran acudir a un hospital, situado en una esquina del entorno, y realizar un análisis cada cierto intervalo de tiempo, dicha funcionalidad se podría implementar con un nuevo sensor de luz y una nueva batería. Llevando la complejidad más allá, se podría desarrollar un experimento en el que hubiera varios robots siguiendo los mismos niveles de competencia pero añadiendo un comportamiento más que hiciera que tuvieran una distancia de seguridad entre ellos y que si por algún casual la superaran automáticamente se tendrían que ir a lavar, esto se lograría mediante los sensores de contacto y los de proximidad que actualmente se usan.

Referencias

- [1] IRIN. *Solución al voluntario* 2. Febrero, 2020
- [2] IRIN. *Manual del simulador irsim*. Febrero, 2020
- [3] Motor Schema Based Navigation For a Mobile Robot (R. C. Arkin 1987)
- [4] Alejandro Hossian, Gustavo Monte y Verónica Olivera. Análisis del Comportamiento de Robots Móviles con RNA. Un Acercamiento desde el Paradigma Reactivo.
- [5] José Daniel Hernández Sosa. Adaptación computacional en sistemas percepto-efectores. Propuesta de arquitectura y políticas de control. Marzo, 2003
- [6] Ignacio Herrero Reder. Arquitectura de comportamientos reactivos para agentes robóticos basada en CBR. Noviembre, 2015