

---

# GAAR-I

Generic Arm Assist Robot - Intelligent

---

PROJECT SPRINT #4.

DATE: 19<sup>th</sup> May 2021

ROGER REY MESA #1

JAVIER ALEGRE REVUELTA #2

DANIEL LÓPEZ LARA #3

MOHSIN RIAZ #4

# Table of Contents

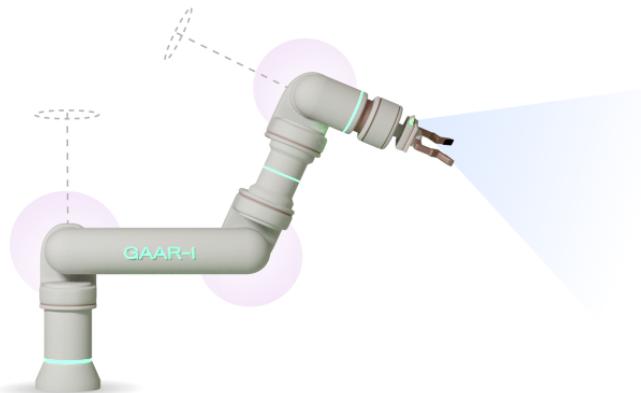
---

<b>Project description</b>	<b>1</b>
<b>Electronic components</b>	<b>3</b>
<b>Hardware Scheme</b>	<b>4</b>
Interconexiones, entradas y salidas	5
<b>Software Architecture</b>	<b>8</b>
Descripción de los módulos	9
<b>Amazing contributions</b>	<b>13</b>
<b>Extra components and 3D pieces</b>	<b>14</b>
Estructura del robot	14
Brazos y conexiones del robot	15
Instrumentos/Objetos 3d	16
<b>Simulation Strategy</b>	<b>17</b>
Diagrama de la estrategia de simulación	19
Máquina de estado: Traer <objeto>	20
Escena del simulador en Coppelia	22
<b>Modules results</b>	<b>26</b>
Visión	26
Cinemática inversa	27
Reconocimiento de voz	29
<b>Foreseen risks and contingency plan</b>	<b>31</b>



# GAAR-I

Generic Arm Assist Robot - Intelligent



## Project description

GAAR-I es un brazo robótico inteligente, asistente de materiales y objetos. A través de un módulo de reconocimiento de voz se le indica un objeto a seleccionar situado en la base del robot, y gracias a la visión por computador identifica el objeto y se lo proporciona al usuario.

Nuestro robot está contextualizado en el ámbito quirúrgico, donde un médico/cirujano pedirá un instrumento que necesite como podría ser el bisturí y se lo acercará.

El funcionamiento del brazo consistirá en una primera fase donde el brazo estará situado en una posición inicial con una cámara apuntando a la base donde se encuentran los instrumentos implicados en la cirugía.

Al emitir una orden de forma oral el cirujano el robot procesa la petición del objeto en demandado y a través de la cámara y algorítmicas de visión por computador se llevará a cabo el reconocimiento del instrumento y dará a la orden al brazo de 5 ejes

para que llegue a esa posición mediante la cinemática inversa. Una vez el instrumento está en la pinza este irá a una posición fija donde sostendrá el objeto a una posición elevada cercana al cirujano para que este pueda coger el objeto deseado sin necesidad de mucho esfuerzo, lo que le permitirá seguir concentrado en la cirugía. En el momento en el que el cirujano coge el instrumento de la pinza este tendrá que emitir otra orden para que se abra la pinza y libere el objeto, una vez liberado el robot volverá a la posición inicial.

En el caso inverso, donde el cirujano quiere dejar un objeto, este tendrá que emitir otra orden para que el brazo vaya a la posición cerca del cirujano y este coloque el objeto en las pinzas, cuando quiera que el brazo cierre la pinza y devuelva el objeto juntos al resto deberá especificarlo.

Las órdenes disponibles que hay son:

- GAAR-I “objeto”: Con esta orden el brazo acercará el objeto al cirujano y esperará a nuevas órdenes. La palabra “objeto” deberá ser sustituida por el instrumento deseado.
- GAAR-I abre: La pinza se abrirá y después de un segundo volverá a su posición inicial.
- GAAR-I devuelve “objeto”: La pinza se cerrará y después de un segundo procederá a dejar el objeto en su posición inicial y volverá a su posición inicial. La palabra “objeto” deberá ser sustituida por el instrumento deseado.
- GAAR-I ven: Con esta orden el brazo se acercará al cirujano con la pinza abierta a la espera de nuevas órdenes.

El conjunto de objetos que va a ser capaz de reconocer serán los siguientes:

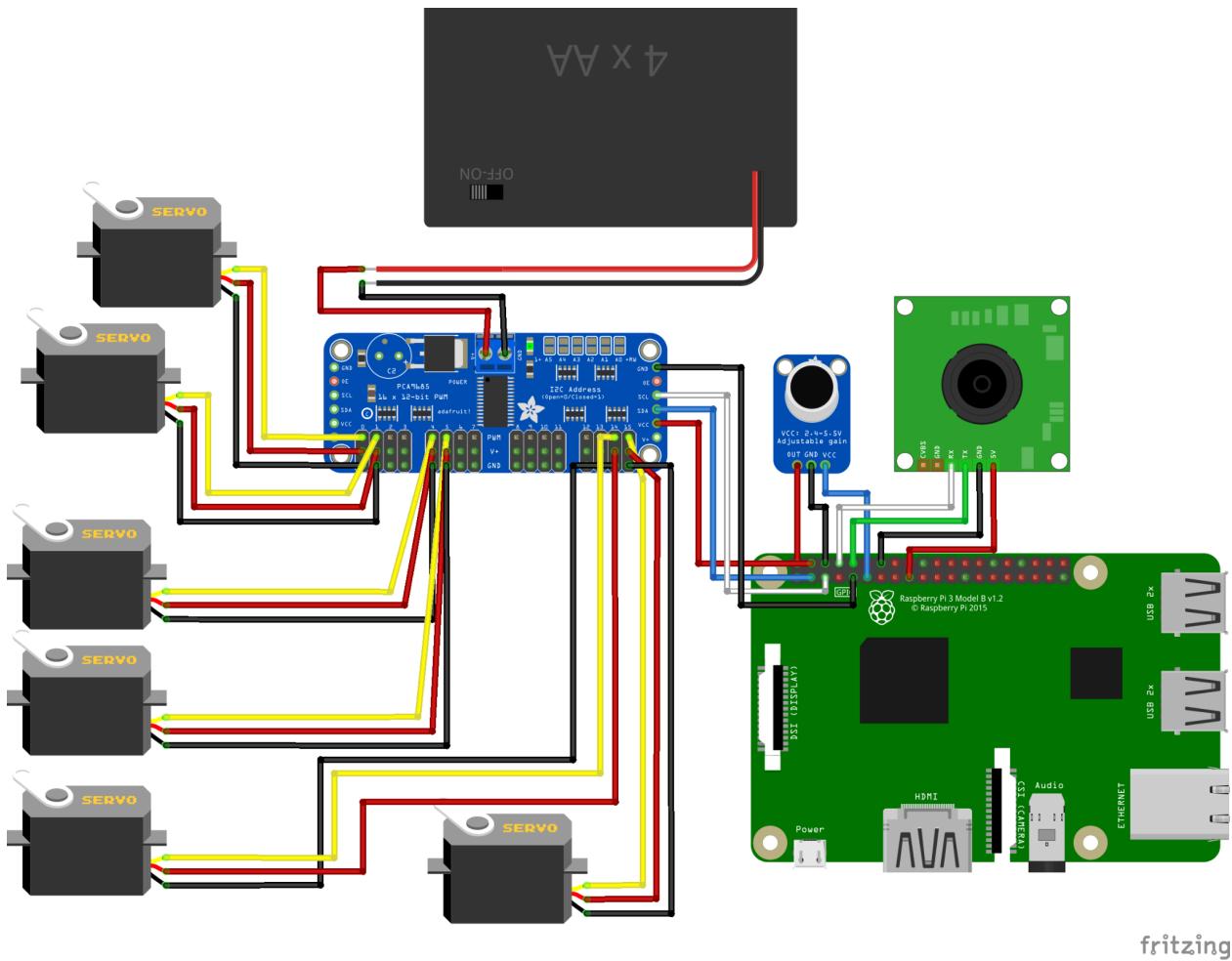
- Pinzas
- Bisturí
- Jeringuilla
- Tijeras

## Electronic components

Esta es la lista de los componentes eléctricos:

- Servo motor, 6 unidades
- Raspberry Pi 3 A+
- Electret Microphone Amplifier - MAX4466 with Adjustable Gain
- Raspberry Pi Camera Module V2
- Driver I2C PCA 9685
- Protoboard
- Cables
- Fuente de alimentación

## Hardware Scheme



## Interconexiones, entradas y salidas

Raspberry Pi Camera Module V2	Raspberry Pi 3 A+
RX	GPIO14
TX	GPIO15
GND	GND
5V	5V

Electret Microphone Amplifier	Raspberry Pi 3 A+
OUT	5V
GND	GND
VCC	GPIO17

Driver I2C PCA 9685	Raspberry Pi 3 A+
GND	GND
SCL	GPIO3
SDA	GPIO2
VCC	5V

Fuente de alimentación	Driver I2C PCA 9685
POS	PWRIN
NEG	GND
Servomotor 1	Driver I2C PCA 9685
Pulse	PWM0
VCC	5V
GND	GND
Servomotor 2	Driver I2C PCA 9685
Pulse	PWM1
VCC	5V
GND	GND
Servomotor 3	Driver I2C PCA 9685
Pulse	PWM4
VCC	5V
GND	GND

Servomotor 4

Driver I2C PCA 9685

Pulse

PWM5

VCC

5V

GND

GND

Servomotor 5

Driver I2C PCA 9685

Pulse

PWM14

VCC

5V

GND

GND

Servomotor 6

Driver I2C PCA 9685

Pulse

PWM15

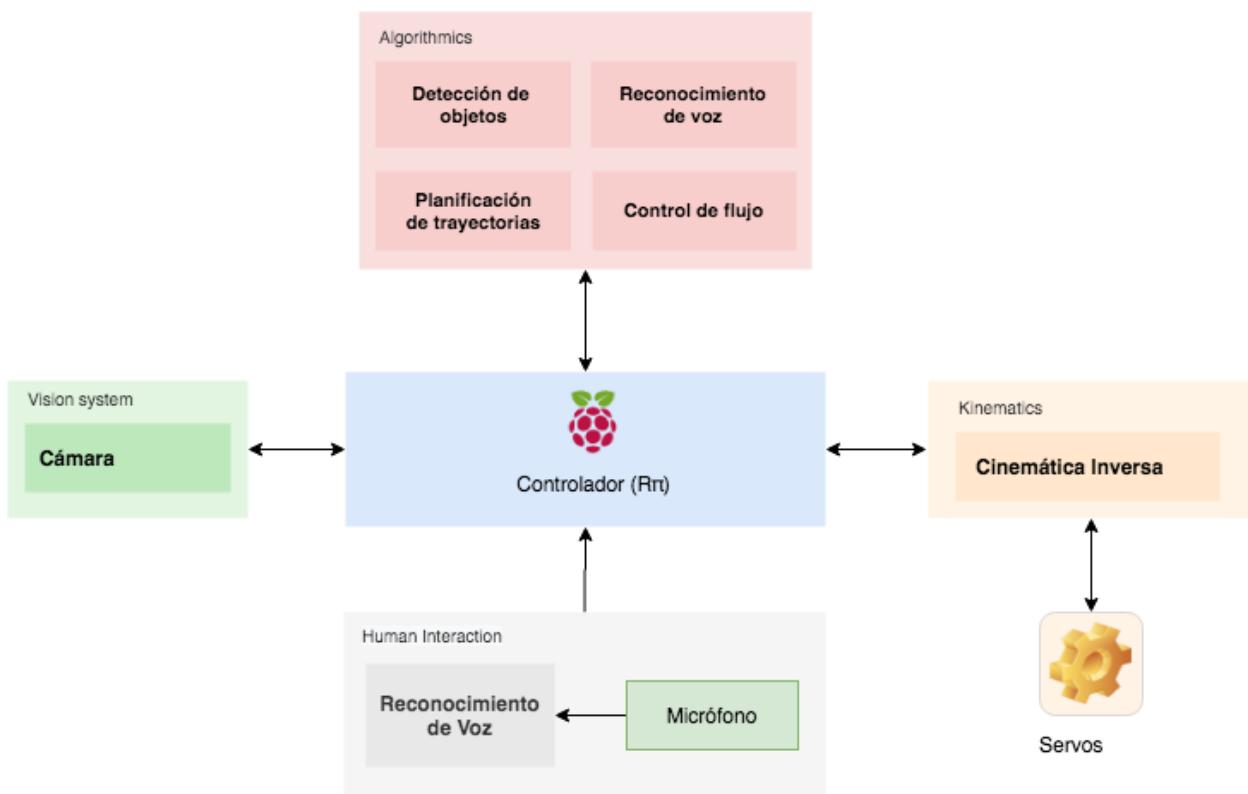
VCC

5V

GND

GND

## Software Architecture



## Descripción de los módulos

Módulo	<b>Reconocimiento de voz</b>
Descripción	<p>Este módulo se dedicará al reconocimiento de voz mediante el micrófono.</p> <p>Para poder pasar de voz a texto se utilizará la librería python <i>SpeechRecognition</i>. Las órdenes vendrán dadas siempre precedidas por la palabra “GAAR-I”, de esta forma al reconocer esta primera palabra se hará una búsqueda en una lista de nombres del resto de la orden.</p> <p>Las órdenes permitidas que siguen al prefijo serán las siguientes:</p> <ul style="list-style-type: none"><li>• “Nombre del objeto”</li><li>• Abre</li><li>• Devuelve “Nombre del objeto”</li><li>• Ven</li></ul> <p>Donde “Nombre del objeto” puede ser cualquiera de los objetos predefinidos que el brazo será capaz de coger (jeringuilla, pinza, bisturí o tijeras).</p>
Entrada	Audio mediante el micrófono.
Salida	Identificador de la orden y el id del objeto.

Módulo	<b>Detección de objetos</b>
Descripción	Este algoritmo se dedicará a captar la imagen de la zona de los objetos y posteriormente mediante técnicas de visión por computador detectar cuál es el objeto que el usuario ha

	<p>indicado.</p> <p>Se utilizará una red YOLOv2 entrenada para clasificar los objetos que capta por cámara. Una vez clasificados los objetos, se selecciona el deseado y se procede a determinar cuál es su orientación y punto por donde se va a agarrar.</p>
Entrada	Imagen de la zona de los objetos.
Salida	<ul style="list-style-type: none"> <li>• Devolver la posición (x,y,z) y la rotación respecto el eje Z, si ha encontrado el objeto.</li> <li>• Devuelve -1, si no ha encontrado el objeto.</li> </ul>

Módulo	<b>Cinemática Inversa</b>
Descripción	<p>A partir de los datos obtenidos del algoritmo de visión o las posiciones ya preprogramadas obtenemos los datos de entrada al algoritmo.</p> <p>Teniendo en cuenta las dimensiones del robot, mediante métodos geométricos y desacoplamiento cinemático mediante los ángulos de Euler, calculamos los ángulos de cada eje para llegar a la posición final deseada.</p> <p>Las operaciones que deberá realizar el robot serán las siguientes:</p> <ul style="list-style-type: none"> <li>• Navegar a la posición deseada.</li> <li>• Recoger o devolver el objeto dependiendo de la orden (abrir y cerrar la pinza).</li> <li>• Recoger un objeto teniendo en cuenta el ángulo de su posición respecto al eje Z.</li> </ul>
Entrada	Posición en 3D.
Salida	Navegar hasta la posición indicada.

Módulo	<b>Planificación de secuencias de movimientos</b>
Descripción	<p>Este algoritmo se dedicará a crear una secuencia de movimientos del robot para poder llevar a cabo una orden.</p> <p>Empezará desde su posición inicial (reposo), ejecutará la orden y volverá a su posición de reposo.</p> <p>Ejemplo:</p> <ul style="list-style-type: none"> <li>• El robot se encuentra en posición de reposo preprogramada.</li> <li>• Se le ordena traer un objeto en concreto.</li> <li>• El robot planificará la siguiente secuencia: <ul style="list-style-type: none"> <li>◦ Ir desde la posición actual a la de la zona de los objetos.</li> <li>◦ Navegar a la posición concreta del objeto a recoger.</li> <li>◦ Realizar las operaciones de la pinza para agarrar el objeto.</li> <li>◦ Navegar a la posición preprogramada para ofrecer el objeto al usuario.</li> <li>◦ Volver a la posición de reposo del robot.</li> </ul> </li> </ul>
Entrada	Identificador de la orden e id del objeto.
Salida	Una secuencia de movimientos del robot.

Módulo	<b>Control de flujo (controlador)</b>
Descripción	<p>Este módulo se dedicará a controlar y coordinar el resto de módulos para poder efectuar las órdenes deseadas por el usuario.</p> <p>Esta parte será la encargada de:</p>

- 
- Poner en marcha al robot y realizar la secuencia de inicialización y pruebas.
  - Tener integrados los algoritmos de voice-to-text.
  - Tener integrado el algoritmo de planificación de la secuencia de movimientos.
  - Ejecutar la secuencia de movimiento usando la cinemática inversa.
  - Usar algoritmos de visión para ubicar los objetos.
  - Controlar el estado de la secuencia de movimientos.
  - Ver si la orden se ha completado para volver a la posición de reposo.
  - Esperar a la siguiente orden.
-

## Amazing contributions

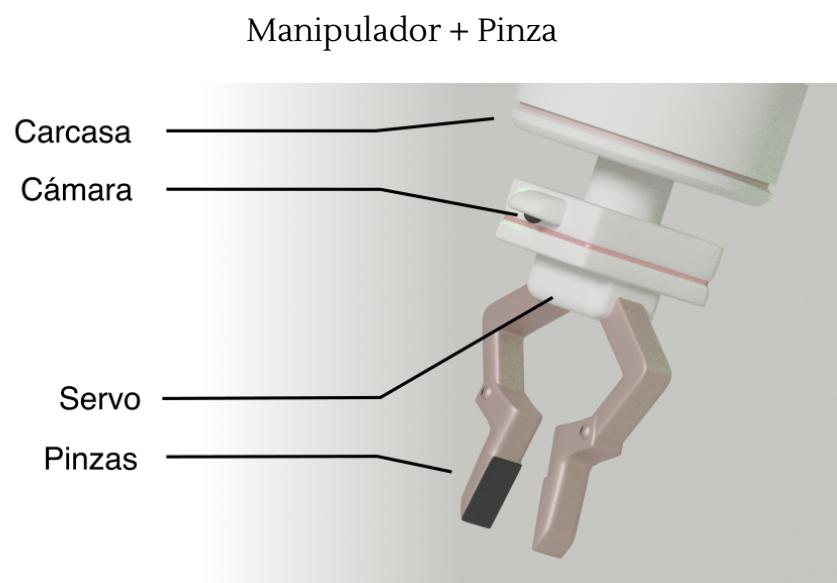
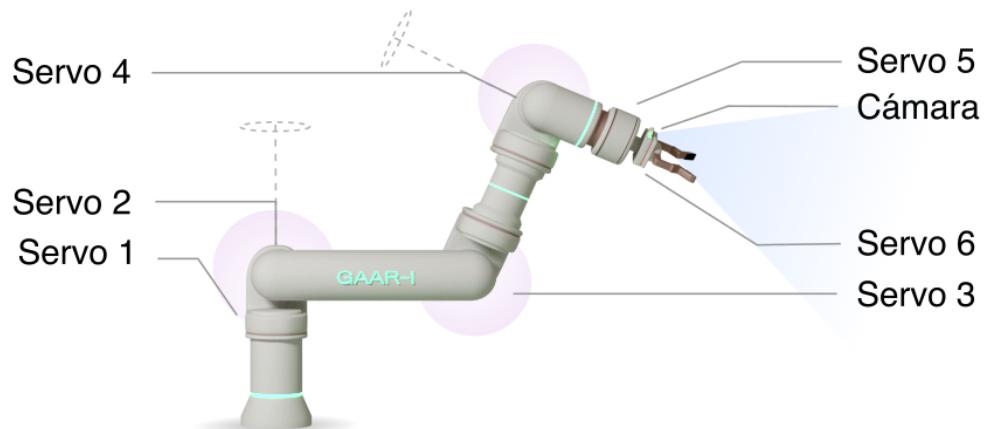
Estas son las contribuciones increíbles que aporta nuestro proyecto al mundo, sobre todo en el ámbito quirúrgico y medicinal:

- Automatización del proceso de asistencia de objetos a un cirujano/médico.
- Reducción de personal durante los procesos quirúrgicos en una sala de operaciones.
- Mayor precisión a la hora de asistir al cirujano/médico acercando los instrumentos que sean necesarios.
- Gracias al control por voz, se permite dar órdenes al brazo robótico y tener libres las manos al mismo tiempo.
- Visión por computador integrada con una inteligencia artificial que permite determinar cuál es el objeto deseado, permitiendo que la colocación de los objetos no esté totalmente predefinida y tenga cierto grado de flexibilidad.
- Evita la pérdida de tiempo y errores humanos derivados de la presión, distracción o cansancio. ya que el brazo robótico reconoce y encuentra el objeto rápidamente.

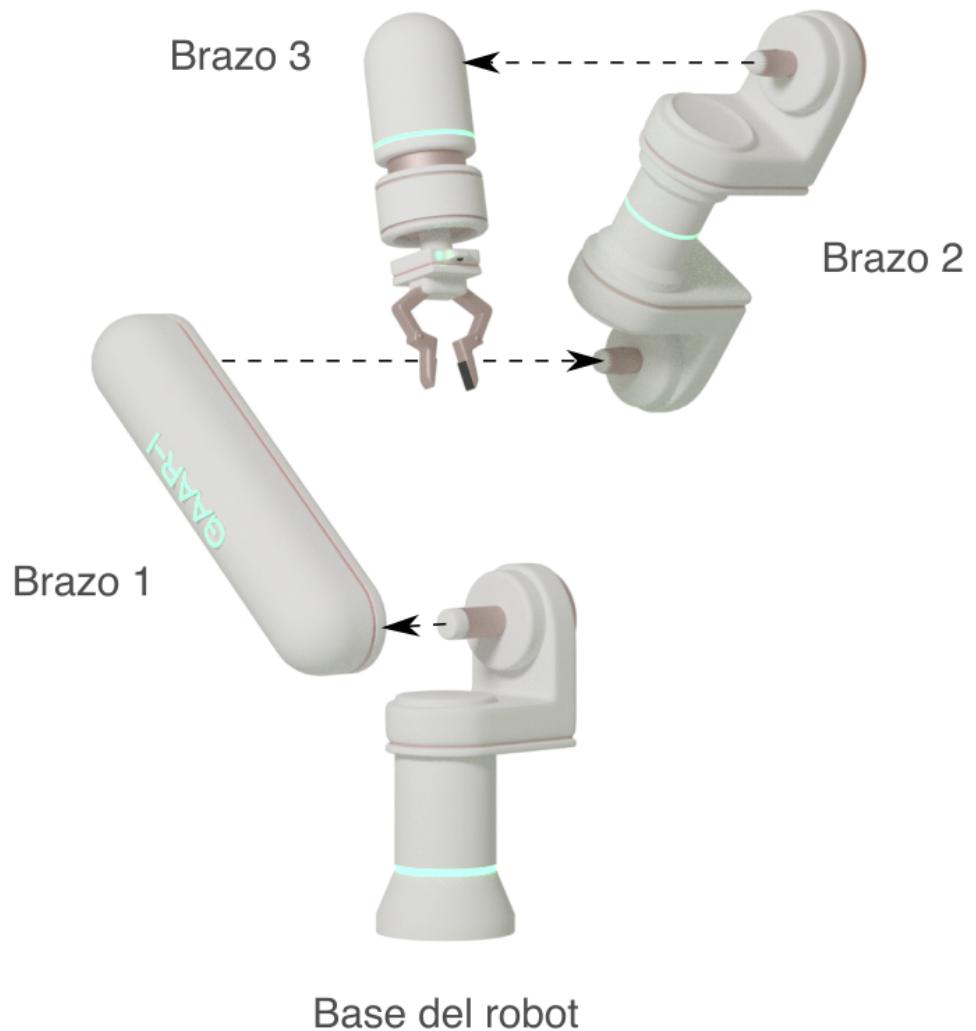
Consideramos que la realización de todos las características que hemos presentado, constituyen un proyecto sobre un brazo robótico inteligente completo y operativo como asistente de materiales y objetos en el ámbito quirúrgico y medicinal. Pensar en añadir más funcionalidades solo adornará el proyecto, pero realmente no le aportaría más valor, ya que para el problema que queremos resolver esta solución lo resuelve a la perfección. Además, consideramos que GAAR-I podría aportar valor al mercado y ser de gran utilidad en su ámbito.

## Extra components and 3D pieces

### Estructura del robot



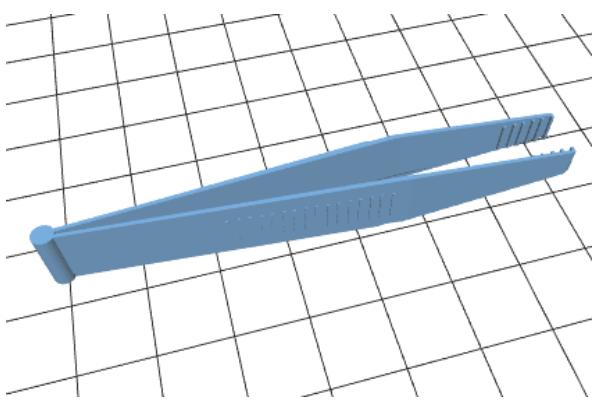
## Brazos y conexiones del robot



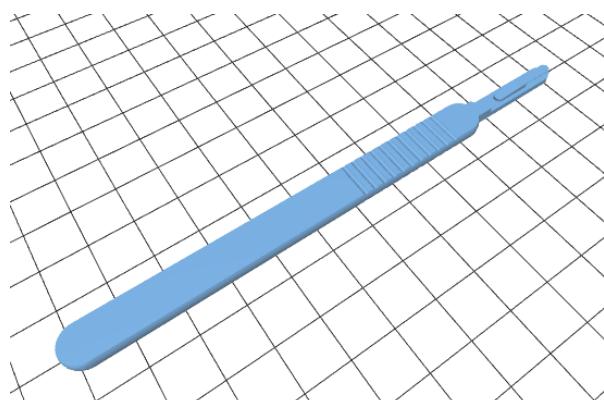
## Instrumentos/Objetos 3d

GAAR-I está contextualizado en el ámbito quirúrgico, por ello los objetos con los que interactúa también pertenecerán a ese mismo ámbito. Se han utilizado modelos de varios instrumentos quirúrgicos tanto para la simulación como para el entrenamiento de la red neuronal encargada de identificarlos dentro de una imagen tomada por la cámara del robot.

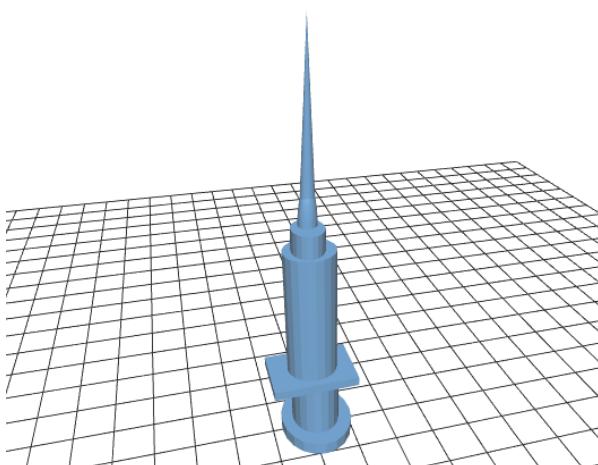
Pinzas



Bisturí



Jeringuilla



Tijeras



## Simulation Strategy

Los diseños para la escena de la simulación se realizan mediante tinkercad. La escena, estará ambientada en una sala de operaciones donde se realiza una operación. Dispondremos de varios objetos para realizar una simulación lo más realista posible, como son una camilla con un paciente tumbado encima, una mesita quirúrgica donde se entregarán los objetos que se ordene el cirujano/médico, una mesa sobre la que estará GAAR-I, en la misma mesa estarán depositados los objetos que se le pedirán a GAAR-I y más elementos de atrezzo para la escena como paredes, una puerta, estanterías, botiquines, lámparas e incluso aparatos tecnológicos . El robot estará posicionado de manera estratégica para poder asistir óptimamente al cirujano/médico, que estará frente al paciente.

La simulación se realizará mediante la interconexión entre Coppelia y Script de Python.

- **Coppelia:** herramienta utilizada que representará la simulación. Podremos observar la escena sobre el comportamiento del robot ante los comandos que se le vayan indicando.
- **Python Script:** aquí es donde todo el código que conforma la inteligencia artificial del robot se ejecutará. Por tanto, se encargará de ejecutar el código necesario para procesar las entradas de voz, determinar cuál es el objeto requerido, determinar la posición donde se encuentra, hacer que el robot lo coja y se lo entregue al cirujano.

La estrategia de simulación va a ser la siguiente, dividida en 2 situaciones:

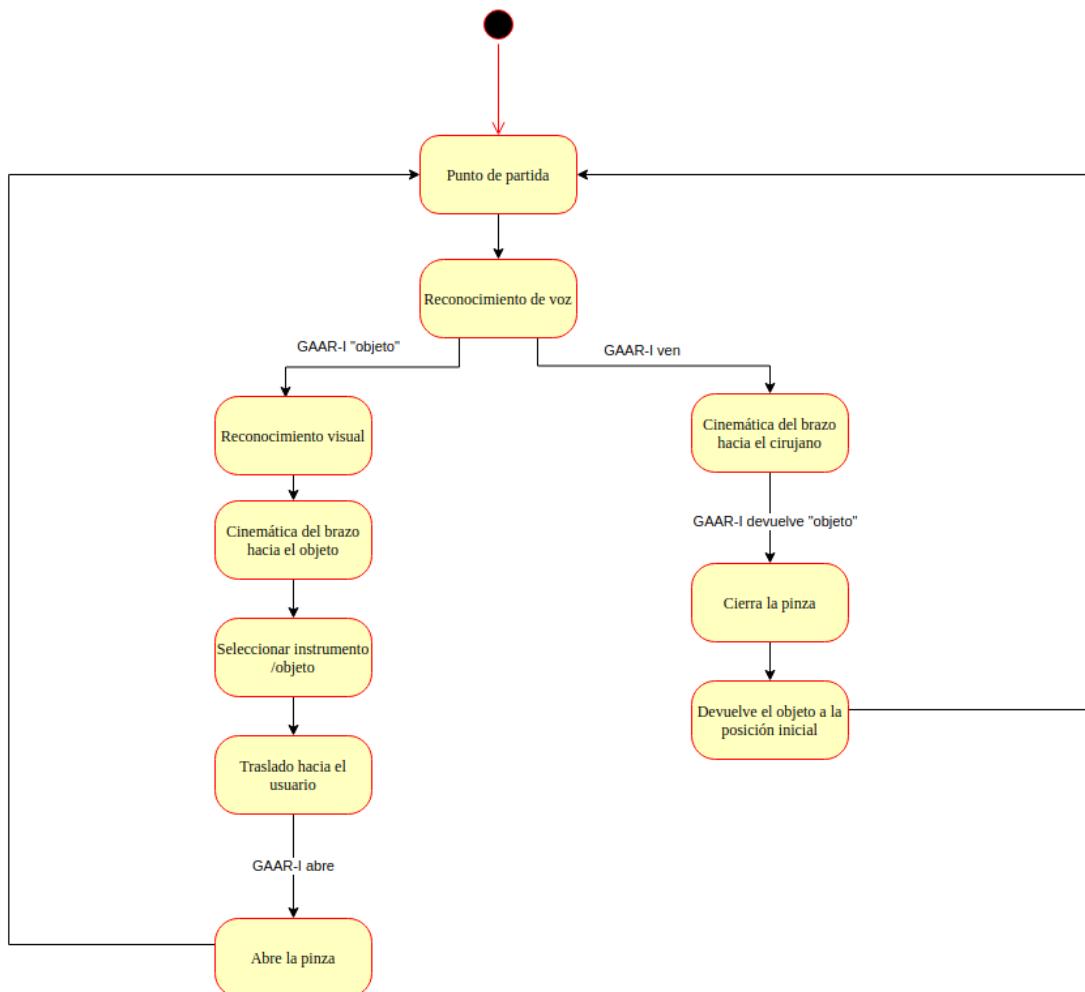
- **El cirujano quiere un instrumento:**
  - **Punto de partida:** GAAR-I, siempre está en un punto inicial fijo con la pinza abierta, esperando las órdenes del usuario.
  - **Reconocimiento de voz:** mediante un micro indicaremos cuál es el instrumento requerido mediante el comando "GAAR-I <<objeto>>". En Python, donde estará implementada la inteligencia artificial capaz

de reconocer y procesar la voz, se determinará cuál es el objeto que se ha solicitado.

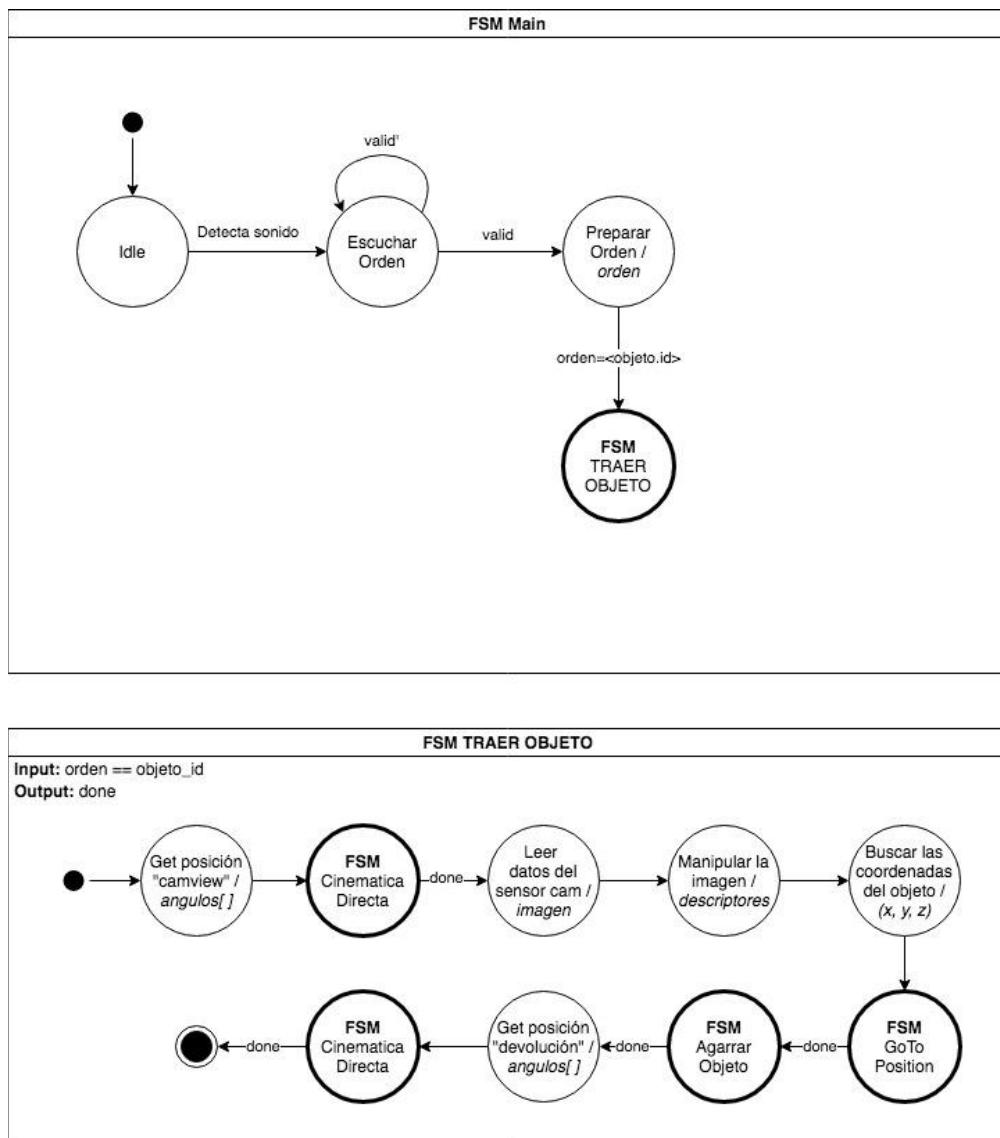
- **Reconocimiento visual:** desde el Coppelia se llevará la imagen al Python, donde la inteligencia artificial procesa la imagen y determinará dónde se encuentra el objeto.
  - **Cinemática del brazo:** se realizará la cinemática inversa para determinar los ángulos necesarios de cada motor de GAAR-I para llegar a la posición necesaria. Desde Python se enviarán las órdenes para que el robot se mueva a esa posición. En un diccionario, asociaremos el nombre del instrumento con la posición a la que se encuentra.
  - **Seleccionar instrumento/objeto:** el robot una vez situado encima de la pieza (a una altura fija determinada) comenzará a cerrar las pinzas hasta que el objeto quede sujetado.
  - **Traslado hacia el usuario:** el brazo una vez haya recogido la pieza se desplazará a la posición fijada determinada, donde el cirujano podría recogerlo. Una vez el cirujano esté preparado para coger esa pieza, le indicará que abra la pinza con el comando por voz pertinente.
  - **GAAR-I vuelta a origen:** posteriormente al paso anterior, GAAR-I volverá a la posición de inicio, preparado para volver a iniciar el primer paso punto de partida.
- 
- **El cirujano quiere devolver un instrumento:**
    - **Punto de partida:** GAAR-I estará en el punto inicial del que siempre parte con la pinza abierta.
    - **Reconocimiento de voz:** mediante el micro indicaremos que queremos devolver el objeto con el comando “GAAR-I ven” y se procesa.

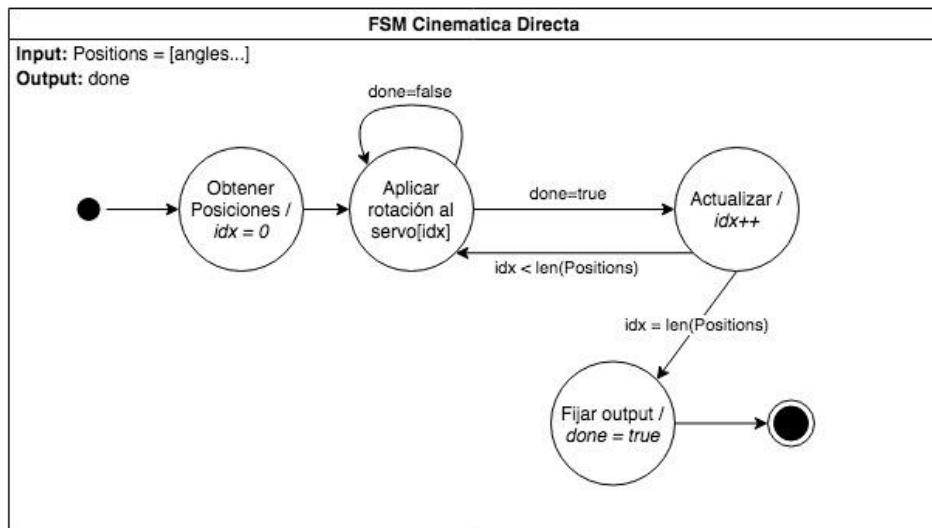
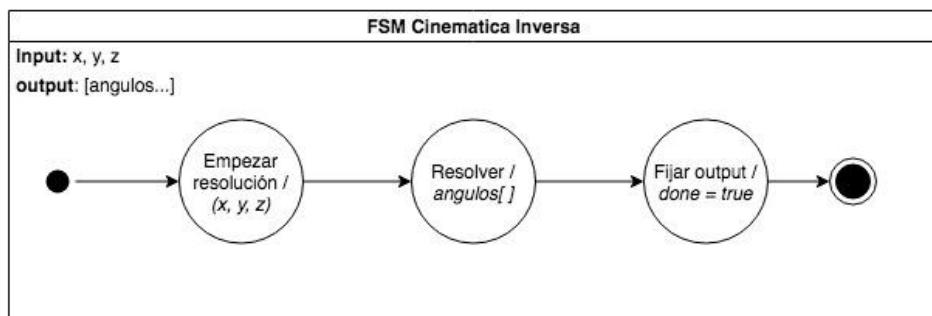
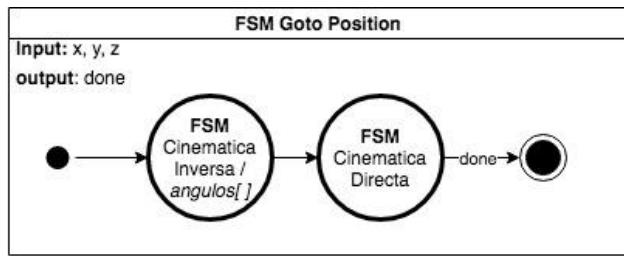
- **Movimiento brazo:** el brazo aplicará los ángulos predefinidos para realizar la acción de devolver y colocarse en la posición en la que va a recibir el objeto.
- Se colocará el objeto en la pinza del brazo y con el comando “GAAR-I devuelve <<instrumento>>”. Gracias al diccionario, se podrá determinar la posición asociada a ese instrumento.
- GAAR-I irá a esa posición y dejará el objeto en la base.
- **GAAR-I vuelta al origen:** GAAR-I vuelve al punto de partida.

### Diagrama de la estrategia de simulación



## Máquina de estado: Traer <objeto>



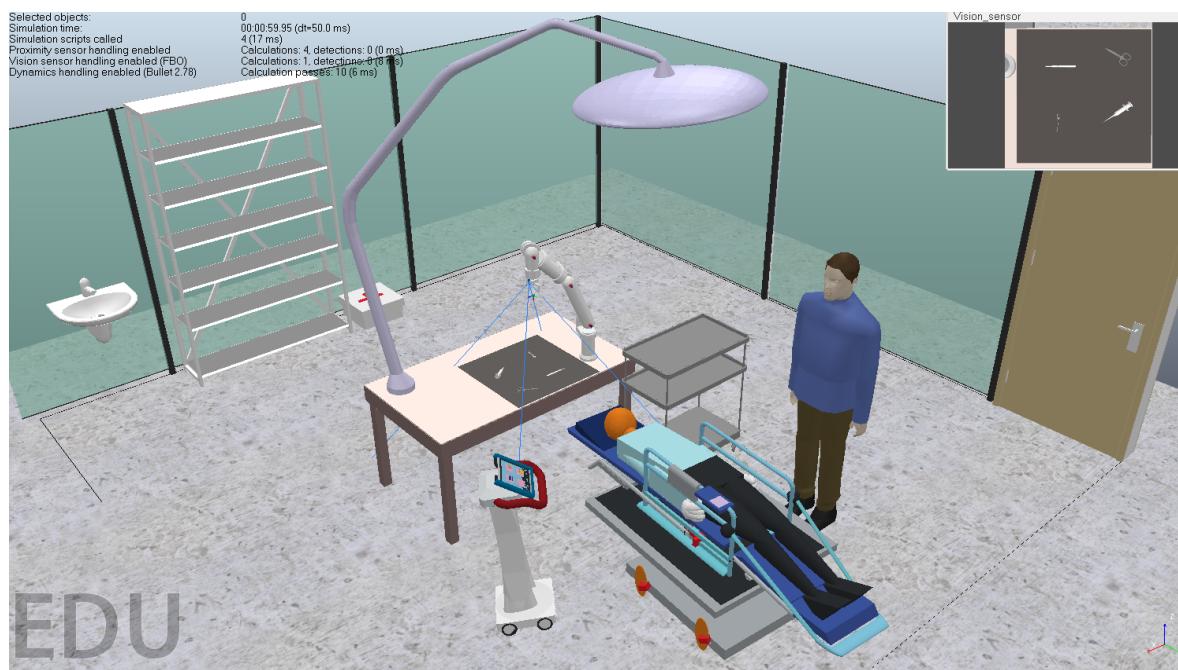


## Escena del simulador en Coppelia

Tal y como se ha mencionado anteriormente, se disponen de varios elementos de atrezzo para conseguir una simulación lo más realista posible dentro de una sala de operaciones. Los elementos más destacados serían:

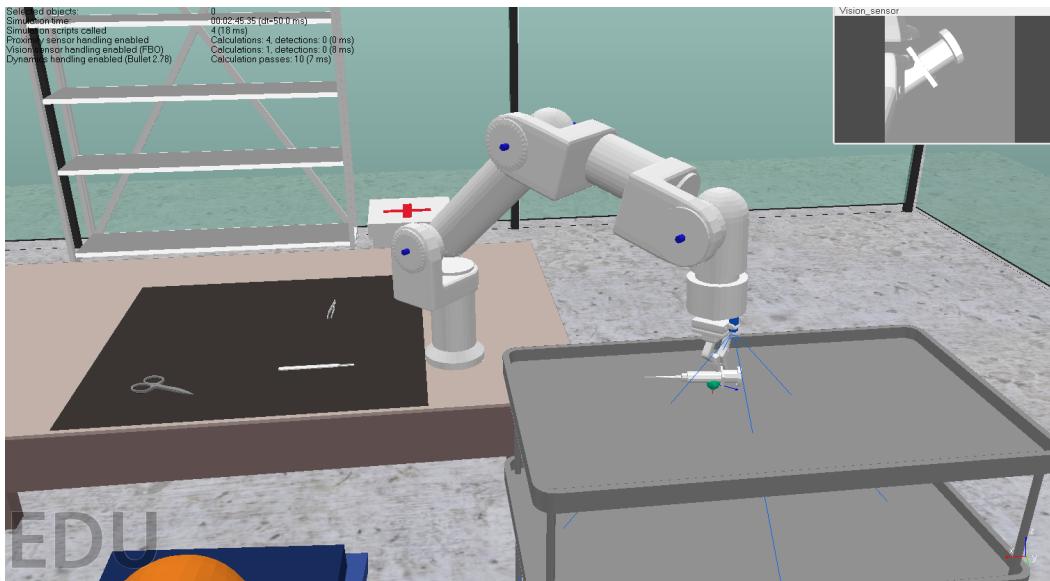
- Mesa con los 4 objetos interactuables con GAAR-I (tijeras, bisturí, jeringuilla y pinza ).
- Robot GAAR-I sobre la misma mesa anterior.
- Mesa quirúrgica pequeña de operaciones donde se depositarán los objetos que ordene el cirujano.
- Cirujano/doctor.
- Camilla con paciente (maniquí) encima.

La siguiente imagen muestra un plano general de toda la escena y en la parte superior derecha se muestra la visión de la cámara de GAAR-I.

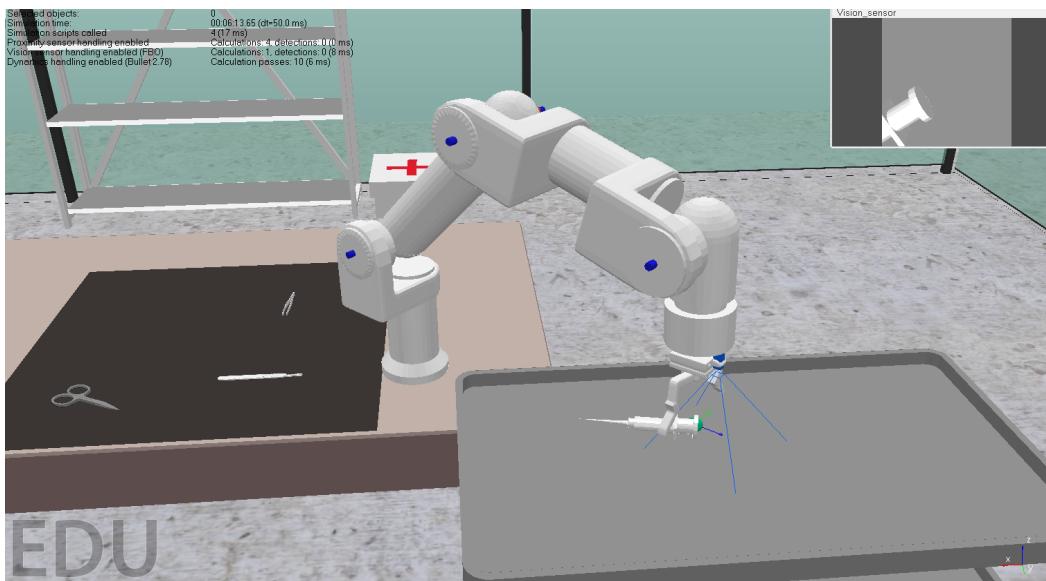


En esta segunda imagen, podemos observar un plano más cercano a GAAR-I y como se ha procesado la orden “**GAAR-I Jeringuilla**”, se ha colocado en posición de entrega, a la espera de la orden “**GAAR-I abre**” para abrir su pinza y de esta manera

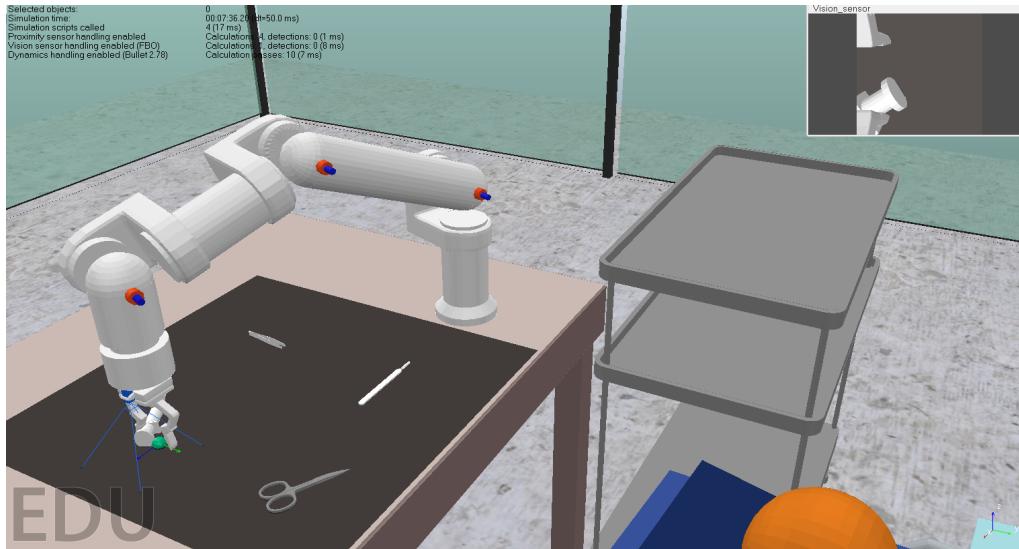
soltar el objeto en la mesa quirúrgica y volver a su posición inicial. Además también podemos observar como la visión de la cámara de GAAR-I muestra el objeto jeringuilla sujetado por la pinza.



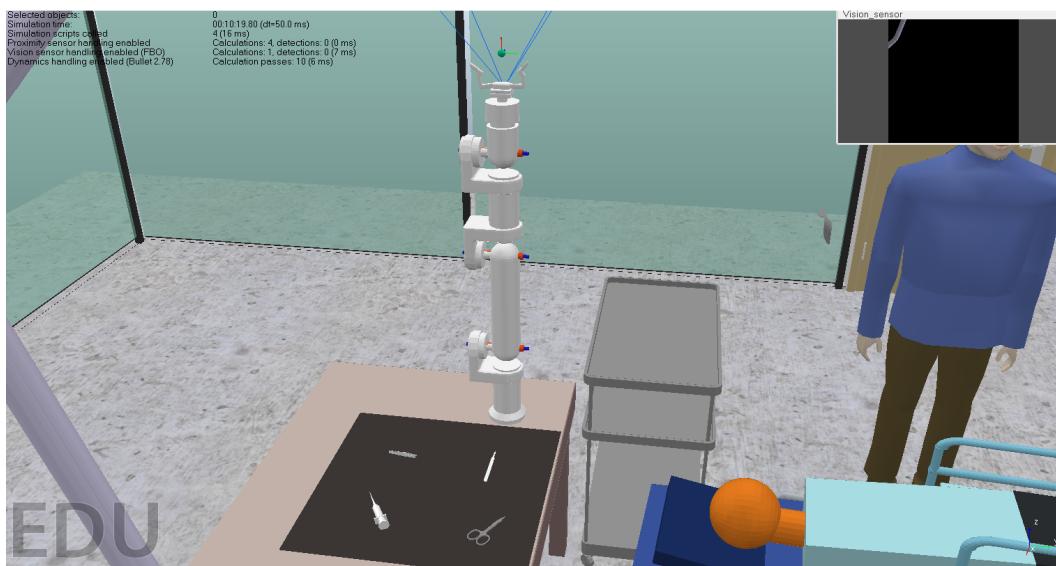
Seguimos con la orden “**GAAR-I ven**” para poderle devolver el objeto que nos ha entregado previamente. En este caso, GAAR-I se coloca en la posición de entrega/recogida con la pinza abierta a la espera de la siguiente orden.



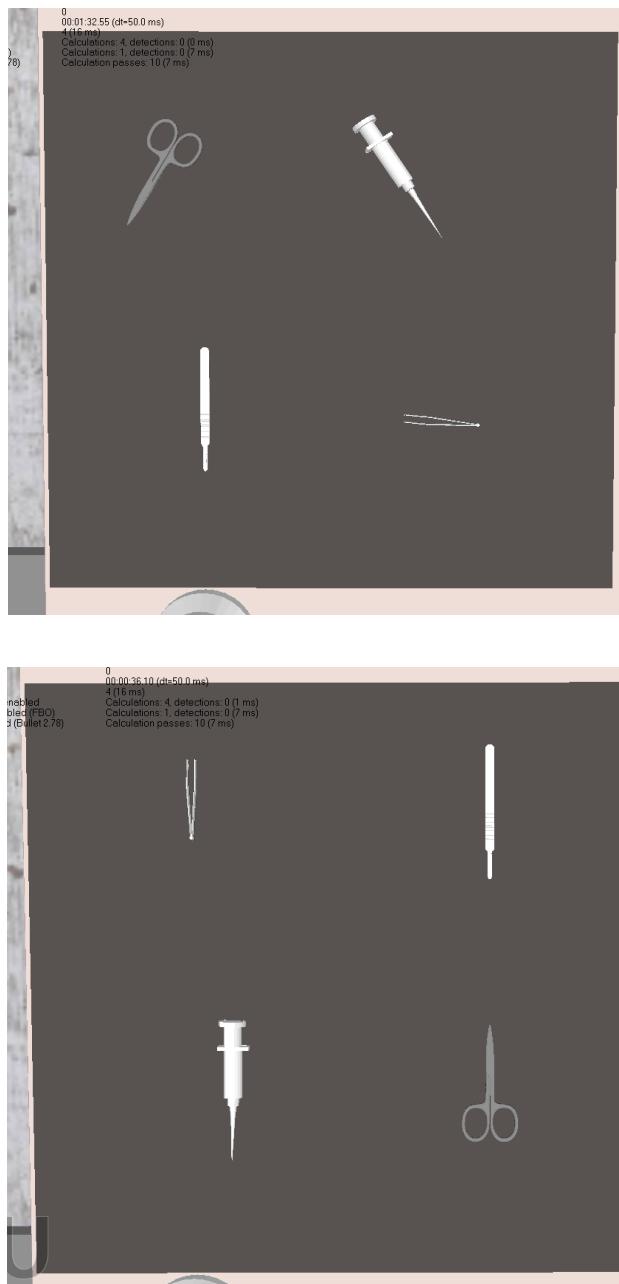
Con la orden “**GAAR-I devuelve jeringuilla**”, agarrara el objeto y lo dejara en la misma posición en la cual estaba originalmente. Cuando acabe este proceso se quedará en su posición inicial a la espera de la siguiente orden.



De esta manera se completa el proceso de entregar y devolver un objeto. Partiendo desde el punto de inicio se puede hacer el mismo proceso con los otros objetos e incluso el mismo objeto. En este caso, queremos apagar a GAAR-I y procesamos la orden “**apágate**” para que vaya su posición de apagado.



Disponemos de dos escenas diferentes para la simulación. La diferencia entre estas se encuentra en la colocación de los objetos de la zona de recogida, para poder demostrar correctamente el funcionamiento de la visión por computador. La primera imagen corresponde a la escena 1 y la segunda a la escena 2.



## Modules results

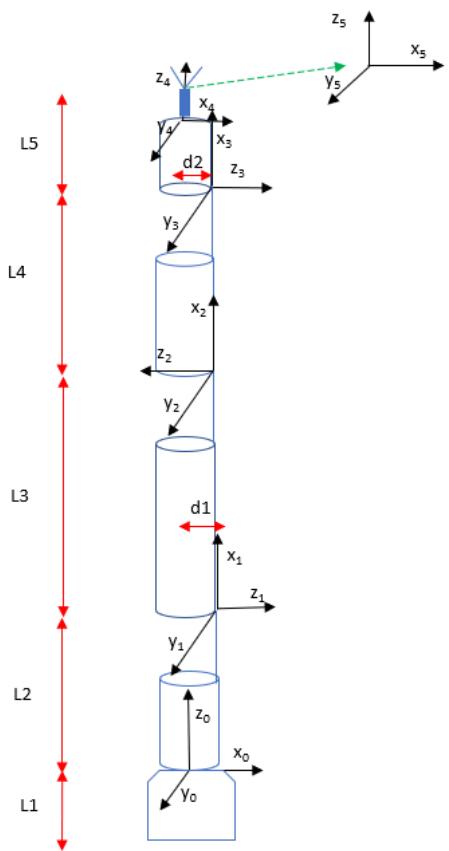
### Visión

El módulo de visión ha sido implementado mediante la red neuronal YOLOv2. Para que esta proceda a la detección de objetos primeramente se ha realizado un trabajo de recopilación de datos, más concretamente se han obtenido aproximadamente unas 2000 imágenes de los objetos a reconocer con diferentes ángulos, perspectivas y escalas y sobre diferentes paisajes. Después de capturar las imágenes se ha realizado un proceso de etiquetado, en el cual mediante una herramienta especializada para ello se ha especificado qué objeto es cada uno, creando así las 4 clases de objetos que se detectan.

Una vez ya obtenido el dataset se ha procedido a entrenar la red para conseguir la clasificación de los objetos que aparecen en la escena. Una vez ya entrenada, como entrada toma la nueva imagen como entrada y como salida devuelve los bounding box donde se encuentra cada objeto.

Una vez se tiene el bouding box recortamos la imagen y la binarizamos para luego, aplicando morfología matemática, encontrar el objetos con una forma sólida, minimizando agujeros dentro del objeto y eliminando pequeños píxeles ruidosos de la imagen. Con esto podemos aplicar una función para encontrar el centro del objeto y su orientación, es decir encontramos la posición x e y de destino y la orientación que debe tomar el último eje para poder coger el objeto.

## Cinemática inversa



Parámetros Denavit-hartenberg:

Articulación	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	$L_1 + L_2$	0	$90^\circ$
2	$\theta_2$	$d_1$	$L_3$	0
3	$\theta_3$	$d_2$	$L_4$	0
4	$\theta_4$	0	0	$-90^\circ$
5	$\theta_5$	$L_5$	0	0

Para realizar la cinemática inversa del robot, al tratarse de un brazo de cinco ejes no se ha podido realizar mediante la matriz de parámetros de Denavit-Hartenberg. Por ello la solución se ha realizado mediante métodos geométricos y desacoplamiento cinemático mediante los ángulos de Euler.

GAAR-I tiene el objetivo de determinar la posición en la que se encuentra el objeto deseado, para ello tiene acoplada una cámara que le permite captar los instrumentos que hay. La cámara está enfocada en el área de trabajo del brazo, por ello lo que se captura es una pequeña parte de la escena. Mediante la visión por computador se determina cuál es el instrumento deseado y determina cuáles son las coordenadas de ese objeto respecto al área de trabajo. Una vez determinadas, estas coordenadas se tienen que pasar a las coordenadas reales globales de la escena, para ello se hace un escalado y una traslación en base a unos puntos de referencia predefinidos.

## Reconocimiento de voz

Tal y como se ha mencionado en el apartado de *Arquitectura del Software*, uno de los módulos más importantes es el reconocimiento de voz. En este apartado, se explicará más en detalle su funcionamiento y se mostrará el resultado obtenido.

La librería de python *SpeechRecognition* nos permite mediante la API de Google, transformar un input de audio a texto. De esta manera podemos pasarle los comandos a procesar por los demás módulos de GAAR-I. Una de las características de este módulo es que está constantemente “escuchando” o esperando una orden por parte del usuario, debido a esto, se le ha añadido una opción de supresión de sonido ambiente para que el input pueda ser más nítido. Además, se ha escogido como único idioma interprete el español. Este es el listado total de órdenes que GAAR-I acepta:

- GAAR-I tijeras (entrega el objeto tijeras)
- GAAR-I bisturí (entrega el objeto bisturí)
- GAAR-I jeringuilla (entrega el objeto jeringuilla)
- GAAR-I pinza (entrega el objeto pinza)
- GAAR-I abre (abre la pinza)
- GAAR-I ven (va a la posición de entrega/recogida)
- GAAR-I devuelve tijeras (devuelve tijeras a su posición inicial)
- GAAR-I devuelve bisturí (devuelve bisturí a su posición inicial)
- GAAR-I devuelve jeringuilla (devuelve jeringuilla a su posición inicial)
- GAAR-I devuelve pinza (devuelve pinza a su posición inicial)
- apágate / adiós (apaga el robot)

El nombre GAAR-I lo detecta como “Gary” o “Cari”, de manera interna se trata con estos términos.

Constantemente está sacando información por consola para saber el estado en el cual nos encontramos. En caso que se introduzca una orden válida, mostrará ésta por consola y en caso que la orden no sea válida, pedirá que se vuelva a repetir. Este es un ejemplo de un ciclo entero de entregar y recoger un objeto con errores forzados de por medio:



```
Simulator incializado
GAARI says: Inicializando...
GAARI says: Ya estoy en reposo
Escuchando...
GAAR-I le escucha
Ha dicho: Hola Gary
GAAR-I le escucha
GAARI says: Disculpa, no se le escucha
GAARI says: Vuelva a probar
GAAR-I le escucha
Ha dicho: Gary jeringuilla
{'label': 'jeringuilla', 'codigo': 22} 0
GAARI says: Gaari coje el objeto
GAAR-I le escucha
Ha dicho: abre
GAAR-I le escucha
Ha dicho: Cari abre
{'label': 'abre', 'codigo': 2} 0
GAARI says: Gaari está abriendo
GAAR-I le escucha
Ha dicho: Gary ven
{'label': 'ven', 'codigo': 1} 0
GAARI says: Gaari viene
GAAR-I le escucha
Ha dicho: Gary devuelve jeringuilla
{'label': 'devuelve', 'codigo': 4} 22
GAARI says: Gaari va a devolver
GAAR-I le escucha
Ha dicho: Gary gira
{'label': None, 'codigo': None} 0
GAARI says: No te he entendido, repite porfavor.
GAAR-I le escucha
Ha dicho: apágate
Apagando...
```

## Foreseen risks and contingency plan

Risk #	Description	Probability (High/Medium/Low)	Impact (High/Medium/Low)	Contingency plan
1	No conseguir el reconocimiento de voz	Baja	Alto	No se puede realizar el proyecto si ocurre esto
2	No conseguir el reconocimiento visual de objeto	Baja	Alto	No se puede realizar el proyecto si ocurre esto
3	No conseguir la cinemática inversa del robot	Baja	Alto	No se puede realizar el proyecto si ocurre esto
4	Baja de un miembro del equipo	Baja	Medio	Repartir el trabajo entre el resto de integrantes del grupo.
5	Coste superior al previsto, por falta de componente.	Medio	Medio	Añadir el componente si no es muy caro, buscar una alternativa o sacrificar prestaciones según el componente en cuestión.

6	No conseguir acabar el trabajo del sprint a tiempo	Medio	Alto	Aumentar las tareas a realizar para el próximo sprint y dedicarle mayor número de horas.
7	Simulador no adecuado	Baja	Alto	Cambiar de simulador o hacer una simulación más simple, dependiendo del estado de progreso del proyecto
8	Mal diseño de los brazos del robot (no llega a las posiciones requeridas)	Baja	Alto	Rediseño (aumentamos o disminuimos la longitud de los brazos)
9	Diseño de la pinza ineficiente para coger los objetos	Medio	Medio	Rediseño de la pinza
10	Componentes eléctricos no adecuados	Baja	Alto	Cambiar de componentes
11	Componentes averiados	Baja	Alto	Habría que volverlo a comprar si es imprescindible o adaptar el robot para suplir su ausencia.

# References

---

Enlace al proyecto principal en el que nos hemos inspirado:

- <https://www.instructables.com/Voice-Controlled-Robot-Arm/>

Enlaces a otros proyectos:

- <https://bit.ly/3dQqsD7>
- <https://bit.ly/3pVwgo0>
- [https://marvelcinematicuniverse.fandom.com/es/wiki/Dum-E\\_y\\_U](https://marvelcinematicuniverse.fandom.com/es/wiki/Dum-E_y_U)