UR5-Scrub-Nurse

Seminario de servicio social & ASE

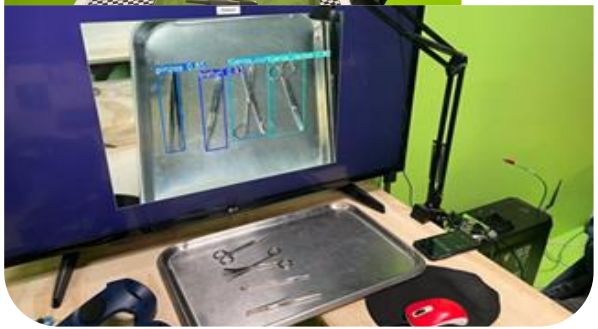
José Pablo Hernández Alonso Ing. Mecatrónica
Aldo Cova Martínez Ing. Sistemas Computacionales

01

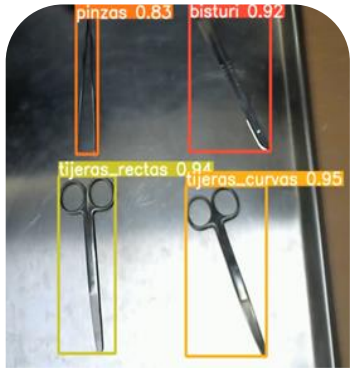
Antecedentes

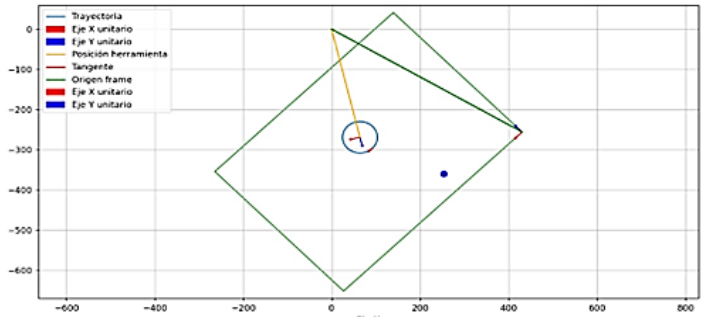
ASE 2:

- Implementación de YOLO en JetsonNano.
- Creación de conjunto de datos de instrumental.
- Interface de escritorio con ajuste dinámico.



Tecnología e instrumentación médica 1:





- Aproximación a movimiento de UR5 mediante Python.
- Clasificador de instrumental mediante IA (con problemas).
- Manejo de cámara de profundidad Intel RealSense para movimiento de robot y marco de referencia adaptativo.

Objetivo del proyecto

Desarrollar una tecnología basada en inteligencia artificial y visión por computadora que permita a un brazo robótico UR5 identificar de manera precisa y eficiente el instrumental quirúrgico, así como manipularlo con la destreza necesaria para asistir en procedimientos quirúrgicos, atacando el problema que implica el presente déficit de enfermeros instrumentistas.

Justificación

Falta de enfermeros

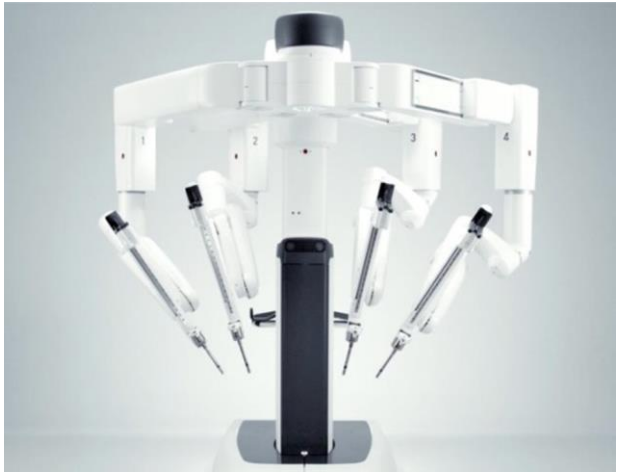
- **Desafío actual**: Crisis mundial de personal de enfermería. En *México se estima que hay 2.5 enfermeros* por cada 1000 personas, *cuando debería haber 9***.**
- **Impacto**: Esta falta de personal aumenta la carga de trabajo y el estrés en el equipo médico, lo que puede afectar la calidad de la atención y la seguridad del paciente. Es esencial encontrar soluciones que alivien esta presión.

La automatización

- **Automatización en la cirugía**: El UR-5e puede encargarse del manejo y la entrega de instrumentos quirúrgicos, asegurando que estén siempre a mano cuando los necesite el cirujano.
- **Fluidez en la intervención**: Esto no solo aumenta la eficacia del procedimiento, sino que también mejora la experiencia del paciente al reducir los tiempos quirúrgicos y potenciar la seguridad en la sala de operaciones.

Reducción de errores

- **Eficiencia en el quirófano**: La incorporación de robots como el UR5e puede optimizar los procesos quirúrgicos, aumentando la rapidez y efectividad de las intervenciones.
- **Asistencia continua**: Este robot proporciona soporte constante a los cirujanos, permitiéndoles concentrarse en su tarea principal sin distracciones, lo que mejora el rendimiento general del equipo.



Manejo de instrumental quirúrgico

- **Automatización en la cirugía**: El UR5e puede encargarse del manejo y la entrega de instrumentos quirúrgicos, asegurando que estén siempre a mano cuando los necesite el cirujano.
- **Fluidez en la intervención**: Esto no solo aumentará la eficacia del procedimiento, sino que también mejorará la experiencia del paciente al reducir los tiempos quirúrgicos y potenciar la seguridad en la sala de operaciones.

Grupos Beneficiados

01 Personal médico y de enfermería
02 Pacientes
03 Instituciones de salud
04 Sector tecnológico

Derechos humanos

Artículo 25 (1): "Toda persona tiene derecho a un nivel de vida adecuado que le asegure, así como a su familia, la salud y el bienestar, en especial la alimentación, el vestido, la vivienda, la asistencia médica y los servicios sociales necesarios".

Artículo 27 (1): "Toda persona tiene derecho a participar libremente en la vida cultural de la comunidad, a gozar de las artes y a participar en el progreso científico y en los beneficios que de él resulten".

Proyecto Broca

Presione para seguir el vínculo
https://dspace.umh.es/handle/11000/31616

Proyecto Gestonurse

Presione para seguir el vínculo
https://web.ics.purdue.edu/~jpwachs/gestonurse/

Enfoque de sector y problema

Enfoque de sector y problema

Contexto del sector: Cirugías ambulatorias y entrega de instrumentos

- **Normas y precisión en hospitales**: La entrega de instrumentos en quirófano debe seguir estrictas normas establecidas por el hospital para asegurar la correcta ejecución del procedimiento.

- **Orden preciso de los instrumentos**: Los cirujanos utilizan una secuencia específica de instrumentos (desde bisturí, pinzas, hasta suturas) dependiendo del tipo de intervención. Un pequeño error en el orden o en la entrega puede resultar en demoras o problemas durante la cirugía.

Problemática del proceso quirúrgico actual

- **Limitación de tiempo y recursos en hospitales públicos**: En hospitales públicos de segundo nivel, existe una gran demanda de cirugías ambulatorias, pero a menudo cuentan con capacidad limitada de personal quirúrgico. Esto puede causar retrasos y sobrecarga en los equipos médicos, afectando la eficiencia y seguridad en la cirugía.

- **Alta rotación de pacientes en hospitales privados**: En hospitales privados y clínicas especializadas en cirugía ambulatoria, se maneja una gran rotación de pacientes. Esto aumenta la presión sobre el equipo quirúrgico, que debe ser capaz de realizar las intervenciones de forma rápida y precisa, sin comprometer la calidad de la atención.

- **El tiempo de instrumentación:** es de aproximadamente 4 segundos dependiendo de varios factores como tipo de cirugía, Experiencia, Entrenamiento, Organización del equipo.

Costos y amortización

Costos de componentes de PC y robot

Producto	Descripción	Rango de Precio (USD)	Rango de Precio (MXN)
UR5 (Robotic Arm)	Robot colaborativo de Universal Robots y distribuidores autorizados	$35,000 - $50,000	-
PC Gama Alta (NVIDIA A4000 / RTX 4050 Super + Componentes)	Computadora de alta gama con componentes para tareas pesadas		-
Procesador (CPU) Intel Core i9 12ª Gen	CPU Intel Core i9 (12ª generación)	$400 - $600	$7,600 - $11,400
Tarjeta Gráfica (GPU)	NVIDIA RTX A4000 o RTX 4050 Super	A4000: $1,000 - $1,500	A4000: $19,000 - $28,500
		RTX 4050 Super: $400 - $600	RTX 4050 Super: $7,600 - $11,400
Placa Base (Motherboard)	Compatible con Intel Core i9	$200 - $400	$3,800 - $7,600
Memoria RAM (32 GB DDR4)	Memoria RAM DDR4 de 32 GB	$100 - $200	$1,900 - $3,800
Almacenamiento (SSD NVMe 1TB)	SSD NVMe de 1TB	$100 - $200	$1,900 - $3,800

Producto	Descripción	Rango de Precio (USD)	Rango de Precio (MXN)
Almacenamiento (SSD NVMe 1TB)	SSD NVMe de 1TB	$100 - $200	$1,900 - $3,800
Fuente de Poder (PSU 750W)	Fuente de poder 750W o más	$80 - $150	$1,500 - $2,900
Caja (Case ATX Mid Tower)	Caja ATX Mid Tower	$50 - $150	$950 - $2,900
Sistema Operativo (Windows 11)	Sistema operativo Windows 11 (opcional)	$100 - $150	$1,900 - $2,900
Intel RealSense D455 Depth Camera	Cámara Intel RealSense D455, 90 fps, USB 3.1, 1280x800 Video	$419.00	$8,400
Intel RealSense D455 Cámara Web	Cámara Intel RealSense D455, 90 fps, USB 3.1, 1280x800 Video	$11,671 (MXN)	$11,671
Intel RealSense D435i Cámara Web	Cámara Intel RealSense D435i, 2 megapíxeles, 30 fps, USB 3.1	$8,672.69 (MXN)	$8,672.69

Salario de un instrumentista

Supuestos iniciales:

- Promedio: $10,500 MXN/mes.
- Sueldo más alto: $37,156 MXN/mes.
- Sueldo más bajo: $14,875 MXN/mes.
- Costo por hora: $64.62 MXN.

Costo del proyecto

- Costo del robot (promedio): $725,000 MXN.
- Costo del equipo de apoyo (PC + Cámara): $45,000 MXN.

Inversión total inicial: $770,000 MXN.

Mantenimiento

- Mantenimiento del robot (10% del costo inicial): $72,500 MXN/anual.

Periodo	Número de pacientes egresados	Porcentaje ambulatorio
2023	2,275,770	69.7%

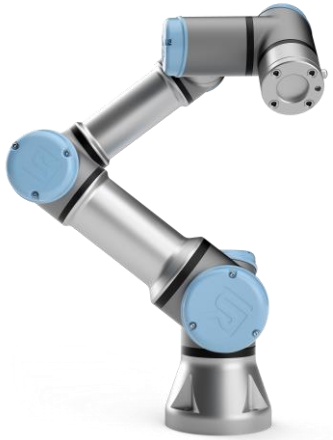
INEGI

Comparación de costos a lo largo del tiempo

Año	Costo del Instrumentista (Promedio)	Costo del Robot (Mantenimiento + Amortización)
1	$126,000 MXN	$770,000 MXN
2	$126,000 MXN	$72,500 MXN
3	$126,000 MXN	$72,500 MXN
4	$126,000 MXN	$72,500 MXN
5	$126,000 MXN	$72,500 MXN
6	$126,000 MXN	$72,500 MXN
7	$126,000 MXN	$72,500 MXN
8	$126,000 MXN	$72,500 MXN
9	$126,000 MXN	$72,500 MXN
10	$126,000 MXN	$72,500 MXN
Total	**$1,260,000 MXN**	**$1,105,000 MXN**

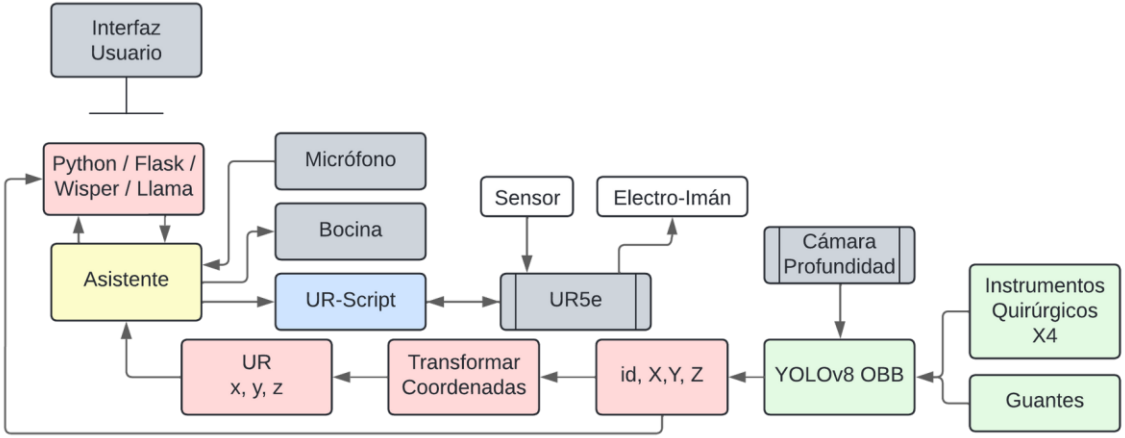
02

Información técnica del proyecto



Información general

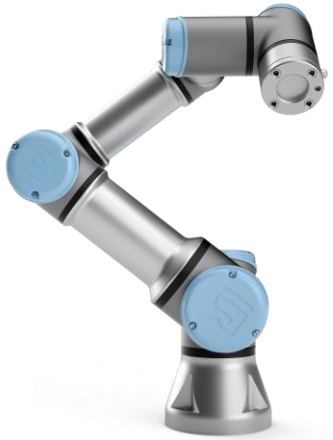
Diagrama del proyecto



El diagrama muestra las tecnologías utilizadas en el proyecto y como se relacionan entre ellas para hacer que el instrumental se reconozca mediante visión con inteligencia artificial y esa información utilizarla para que el robot UR5e tenga la capacidad de manipular los instrumentos mediante una interfaz de usuario (audio/visual) y mediante sensores y actuadores.

- El bloque verde se refiere a el sistema de inteligencia artificial capaz de localizar el instrumental quirúrgico y los guantes del cirujano para entregarlo.
- Los bloques gris y blanco se refiere a elemntos físicos que componen el sistema (la interfaz de usario requiere de una pantalla para ser visualizada).
- Los bloques rojos indican la información que se transmite al sistema y como esta se presenta y transforma.
- El bloque azul indica la forma de comunicación del UR5e con los programas elaborados en python
- El bloque amarillo es el asistente; este es un sistema en el que se desarrolla un algoritmo para indicar el instrumental que se desea (indicaciones aduitivas o por interfaz visual) y basado en la información que posee ejecuta comandos para manipular el instrumental.

- El sistema utiliza python y bibliotecas especializadas en visión artificial, comunicación con UR5e, interacción con cámara de profundidad Intel RealSense, reconocimiento de habla, texto a voz y visualización web para cumplir con su objetivo de presentar una solución tecnologica efieciente para la instrumentación quirúrgica.
- Durante el desarrollo de la documentación de este proyecto se aboradran los puntos necesarios para construir el sistema desde su origen.



Manual UR5e General

Presione para seguir el vínculo

https://www.universal-robots.com/download/manuals-e-seriesur20ur30/user/ur5e/57/user-manual-ur5e-e-series-sw-57-english-us-en-us/

Manual UR5e general

Se sugiere la revisión completa del manual UR5e antes de comenzar a manipular el robot.

El robot se compone de 3 partes fundamentales:

- **Brazo robótico**
  - ❖ Se refiere a el brazo en conjuto con sus articulaciónes y su herramineta de trabajo.
  - ❖ Para el proyecto se elaboro un sistema de cambio rápido para la herramienta de trabajo, los modelos se pueden obtener de la *documentación en github*.
- **Teach-pendant**
  - ❖ Desde la tableta se puede manipular el robot de forma segura. Cuenta con botón de encedidod y apagado para todo el sistema, asi como paro de emergencia.
  - ❖ En este se programa el robot para realización de tareas simples mediante Polyscope, además de que se puede especificar la forma de instalación y los parámetros de seguridad (límites de velocidad, aceleración y posición).
- **Caja de conexiones**
  - ❖ En esta se puede acceder a la memoria SD que contiene el sistema operativo, fusibles, conexión a brazo y teach-pendant, y conexiones de entradas y salidas (I/O). Se sugiere la revisón del manual para que estas conexiones se realicen de manera adecuada y manteniendo la seguridad e integridad del robot.

UniversalRobots/
URScript_Examples

Manual URe-Script

Presione para seguir el vínculo
https://www.universal-robots.com/download/manuals-e-seriesur20ur30/script/script-manual-e-series-sw-511/

Presione para seguir el vínculo
https://github.com/UniversalRobots/URScript_Examples

Manual UR Script

Los comandos de movimiento y control del robot se realizan mediante el lenguaje UR-Script.

- Estos comandos se envían en formato de string codificados mediante python y socket (ethernet) a la computadora del robot, el cual los recibe y ejecuta ordenes.
- Este método (socket) de comunicación que no puede interrumpir una acción programada.
- Para una comunicación en tiempo real y poder realizar interrupciones es necesario utilizar una comunicación mediante RTDE, pero los comandos de UR-Script permanecen.
- Los comandos que se utilizan principalmente son:

❖ Movel (pose, a=1.2, v=0.25, t=0, r=0)

❖ Movej (q, a=1.4, v=1.05, t=0, r =0)

❖ get_actual_joint_positions()

❖ get_actual_tcp_pose()

GitHub y control de versiones

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE

Git & GitHub

Git

Definición: Es un sistema de control de versiones distribuido.

Propósito:

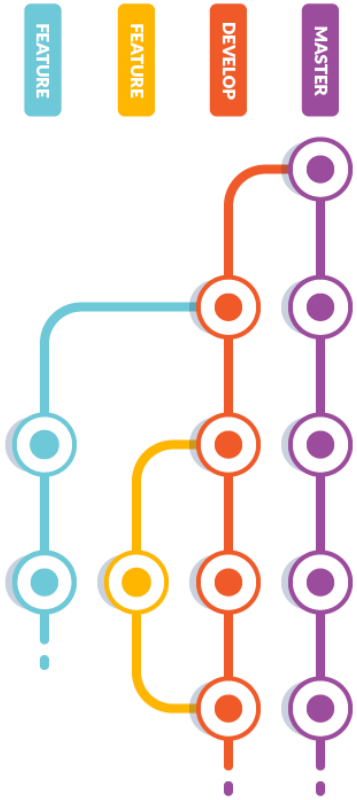
- Permite a los desarrolladores rastrear cambios en el código fuente.
- Facilita la colaboración mediante ramas y fusiones.
- Funciona de manera local, pero puede integrarse con servicios en la nube.

GitHub

Definición: Es una plataforma basada en la nube para alojar repositorios Git.

Propósito:

- Colaboración: Facilita el desarrollo colaborativo en equipo.
- Alojamiento: Permite almacenar repositorios en la nube para acceso global.
- Gestión de Código: Ideal para gestionar, revisar y colaborar en proyectos en tiempo real.
- Integración: Ofrece herramientas adicionales como issues, pull requests y documentación para gestionar proyectos.



GitHub & Estructura de la documentación.

- 3D_Models

Modelos de gripper y “quick-change-tool”

- Manual_UR

Manual de Robot UR5e y UR-Script

- V0.1

Tecnologías e instrumentación 1 (versión sin RTDE)

- V1.0

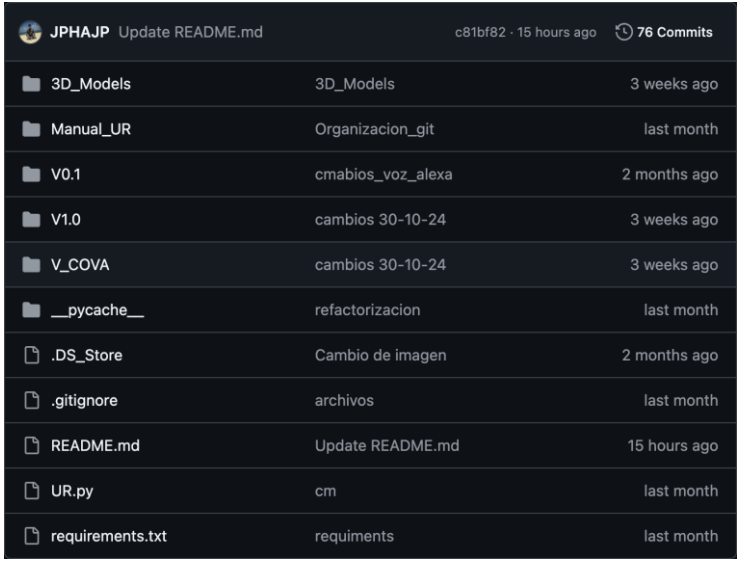
Tecnologías e instrumentación 1 (versión con RTDE y mano).

- V_COVA

Interfaz gráfica para control del robot.

- UR.py

Script principal del proyecto.



Python y entorno virtual

Python y entorno virtual

¿Qué es Python?

• **Definición**: Python es un lenguaje de programación de alto nivel, orientado a objetos, conocido por su simplicidad y legibilidad.

• **Características principales**:
- Diseñado para facilitar el desarrollo de código limpio y fácil de entender.
- Amplia comunidad y soporte, lo que lo hace accesible tanto para principiantes como para expertos.

• **Áreas de uso**:
- **Desarrollo web**: Creación de aplicaciones y sitios web dinámicos.
- **Ciencia de datos**: Análisis, visualización y manipulación de grandes volúmenes de datos.
- **Inteligencia artificial y aprendizaje automático**: Desarrollo de modelos y algoritmos avanzados.
- **Microcontroladores**: Programación de sistemas embebidos para hardware específico.

Python y entorno virtual

Ventajas de Usar Python

1) Simplicidad: Código limpio, fácil de entender y aprender.
2) Versatilidad: Aplicable en desarrollo web, ciencia de datos, IA, y más.
3) Portabilidad: Compatible con múltiples plataformas.
4) Comunidad: Gran soporte, bibliotecas y recursos disponibles.
5) Integración: Fácil conexión con otros lenguajes y herramientas.

Ventajas de los Entornos Virtuales

1) Aislamiento: Evita conflictos entre dependencias de proyectos.
2) Gestión sencilla: Control de paquetes y versiones por proyecto.
3) Estandarización: Uso de requirements.txt para replicar entornos.
4) Flexibilidad: Permite probar sin afectar otros proyectos.
5) Portabilidad: Facilita despliegue y colaboración.

Python y entorno virtual

Versión Python para el Proyecto: Python 3.10.

- Elegida por su compatibilidad con los recursos necesarios para una mejor optimización.

Creación de un Entorno Virtual

Crear el entorno virtual:

```
python3 -m venv nombre_entorno_virtual
```

Activar el entorno virtual (LINUX):

```
source nombre_entorno_virtual/bin/activate
```

Instalar las dependencias del proyecto:

```
pip install -r requirements.txt
```

Ejecutar un código de python

```
python3 code.py
```

Desactivar el entorno virtual (cuando sea necesario):

```
deactivate
```

Bibliotecas utilizadas

Frameworks web y API´s

1. Flask
    - Para creación aplicaciones web y APIs en Python.
2. Werkzeug
    - Una biblioteca WSGI utilizada por Flask para manejar solicitudes HTTP y la interacción entre el servidor web y la aplicación.
3. python-dotenv
    - Una biblioteca que permite cargar variables de entorno desde un archivo `.env` al entorno de la aplicación.

Bibliotecas utilizadas

Procesamiento y análisis de datos

1. Numpy
    - Biblioteca para cálculos numéricos y manejo de matrices multidimensionales.
2. Pandas
    - Herramienta para análisis y manipulación de datos estructurados en tablas.
3. Scipy
    - Extensión de NumPy con herramientas avanzadas de matemáticas, ciencia e ingeniería.

Bibliotecas utilizadas

Visualización

1. Matplotlib
    - Biblioteca para crear gráficos y visualizaciones 2D personalizables.
2. Seaborn
    - Extensión de Matplotlib para generar gráficos estadísticos atractivos y fáciles de interpretar.

Bibliotecas utilizadas

Procesamiento de audio y voz

1. SpeechRecognition
    - Para reconocer y convertir audio en texto mediante diferentes motores de reconocimiento de voz.
2. Sounddevice
    - Biblioteca para grabar y reproducir audio directamente desde dispositivos de entrada/salida.
3. GTTS
    - Genera audio mediante texto utilizando Google Text-to-Speech.
4. Pydub
    - Herramienta para manipular y convertir archivos de audio (cortar, unir, cambiar formato).

Bibliotecas utilizadas

Visión por computadora

1. opencv-python
    - Biblioteca para procesamiento de imágenes
2. Torch y torchvision
    - Framework para aprendizaje profundo
3. Pyrealsense2
    - Interfaz para trabajar con la camara Intel Realsence

Bibliotecas utilizadas

Aprendizaje automático e IA

1. openai-whisper
    - Modelo de transcripción de audio a texto de alta precisión desarrollado por OpenAI.
2. Ultralytics
    - Herramienta para implementar y entrenar modelos YOLO para detección de objetos.
3. Tiktoken
    - Biblioteca para el manejo eficiente de tokens en modelos de lenguaje como GPT
4. Ollama
    - Biblioteca para manejo de procesadores de lenguaje (llama3.1)

OpenAI

ultralytics

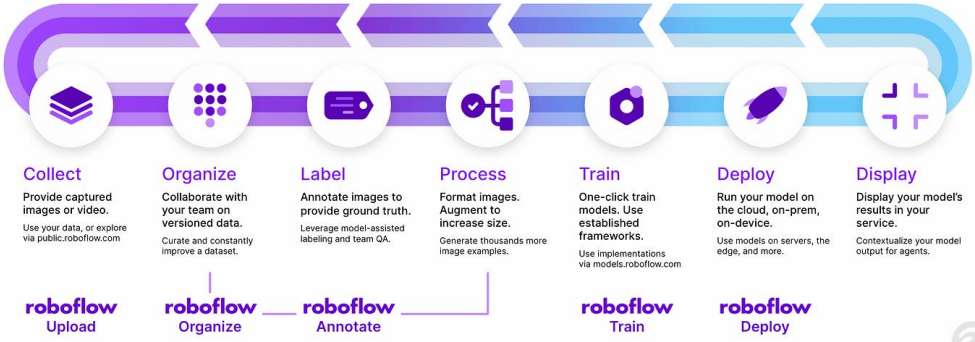
Construcción de conjunto de datos

Presione para seguir el vínculo
https://www.kaggle.com/datasets/jphajp/ur5e-srube-nurse-surgical-instruments/data

roboflow

Es una plataforma que facilita el manejo y procesamiento de datasets para proyectos de visión por computadora.

1. Gestión de conjuntos de datos
2. Herramientas de etiquetado
3. Preprocesamiento
4. Exportación



roboflow

¿Como funciona?

1. Crear una cuenta en Roboflow
2. Crear un nuevo proyecto
    - Tipo de proyecto
    - Formato de anotación
3. Subir imágenes al proyecto
4. Etiquetado de las imágenes
5. Preprocesamiento de datos
    - Resize
    - Augmentation
    - Normalization

roboflow

6. Dividir el conjunto de datos
    - **Entrenamiento (Train):** ~70% de las imágenes.
    - **Validación (Validation):** ~20% de las imágenes.
    - **Prueba (Test):** ~10% de las imágenes.
7. Exportar el dataset
    - **YOLOv8** (TXT y estructura de carpetas compatible).
8. Usar el conjunto de datos en tu proyecto (realizar entrenamiento)

Jupyter notebook para entrenamiento
https://www.kaggle.com/datasets/jphajp/ur5e-srube-nurse-surgical-instruments/data

Configuración de CUDA para manejo de entrenamiento en LINUX

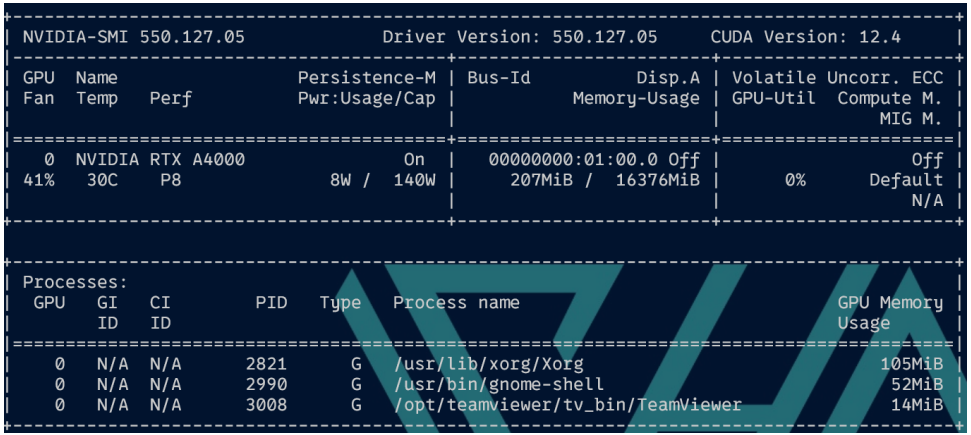
Nvidia CUDA

¿Qué es CUDA?

CUDA (Compute Unified Device Architecture) es una plataforma de computación paralela y una API desarrollada por **NVIDIA**. Permite a los desarrolladores utilizar el poder de las **GPUs (Unidades de Procesamiento Gráfico)** para ejecutar tareas de cómputo intensivo de manera más eficiente que en CPUs (Unidades de Procesamiento Central).

Comando de inspección de procesamiento gráfico:

```
nvidia-smi
```



Nvidia CUDA

Requisitos mínimos

- **Hardware:** Una GPU NVIDIA compatible con CUDA.
- **Software:**
  - Controladores actualizados de NVIDIA.
  - Kit de herramientas CUDA (CUDA Toolkit).
  - Librerías complementarias como cuDNN (para redes neuronales profundas).
- **Lenguaje de programación:** C, C++, Python o cualquier lenguaje compatible.
- **Sistema operativo:** Ubuntu(Linux) / Windows

Nvidia CUDA (Instalación)

1. Descargar CUDA Toolkit (https://developer.nvidia.com/cuda-11-8-0-download-archive)
    - Asegurate de descargar la version adecuada de python con Pythorch o Tensorflow
2. Actualiza el sistema
    - Sudo apt update && sudo apt upgrade
3. Descargar el instalador
    - wget https://developer.download.nvidia.com/compute/cuda/12.1.105/local_installers/cuda-repo-ubuntu2204-12-1-local_12.1.105-515.65.01-1_amd64.deb

Nvidia CUDA (Instalación)

1. Instalar el paquete descargado
   - sudo dpkg -i cuda-repo-ubuntu2204-12-1-local_12.1.105-515.65.01-1_amd64.deb
   - sudo apt-key add /var/cuda-repo-*/7fa2af80.pub
2. Actualizar e instalar CUDA
   - sudo apt update
   - sudo apt install -y cuda
3. **Agregar CUDA al PATH:** Abre el archivo `~/.bashrc`
   - nano ~/.bashrc
   - export PATH=/usr/local/cuda/bin:$PATH
   - export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
   - source ~/.bashrc

Entrenamiento

ULTRALYTICS

Son los desarrolladores oficiales de las versiones más avanzadas del modelo **YOLO (You Only Look Once)**, como **YOLOv8** (utilizado en el proyecto), que es ampliamente utilizado para tareas de detección de objetos, clasificación y segmentación para aplicaciones de procesamiento rápido de imágenes.

Esta herramienta ofrece la ventaja de poder trabajar casi en tiempo real (tiempos de análisis mínimos) en el procesamiento de imágenes para detección y clasificación.

Data Augmentation (Aumentación de Datos)

- Es una técnica utilizada para expandir el conjunto de datos mediante la generación de nuevas imágenes basadas en las existentes. Esto se logra aplicando transformaciones como:
  - Rotaciones.
  - Recortes.
  - Cambios de brillo o contraste.
  - Cajas negras para aprendizaje de objetos cortados (ocultos)
  - Escalado o inversión horizontal/vertical.
- Esto es mayormente utilizado cuando se tiene un conjunto de datos pequeño, o se quiere mejorar el aprendizaje para ciertas situaciones.

Épocas

¿Qué son?

- Una época es una iteración completa sobre todo el conjunto de datos de entrenamiento.
- Si tienes 100 imágenes en el conjunto de datos y el modelo entrena por 10 épocas, esas 100 imágenes serán procesadas 10 veces.

Ejemplo:

yolo train.model=yolov8n.pt
data=dataset.yaml
epochs=10

Batch Size

¿Qué es?

- Es el número de imágenes procesadas simultáneamente antes de que se actualicen los pesos del modelo.
- Por ejemplo, si tienes un batch size de 16, el modelo procesará 16 imágenes a la vez antes de ajustar sus parámetros.

Ejemplo:

```
yolo train.model=yolov8n.pt
data=dataset.yaml
epochs=50
batch=16
```

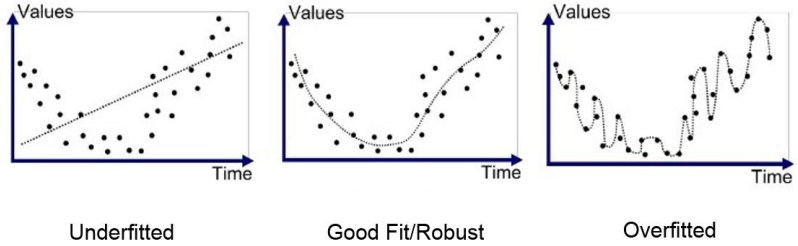
Early Stopping (Parada Temprana) & Sobreajuste (Overfitting)

Parada Temprana

- Es una técnica para detener el entrenamiento cuando el modelo deja de mejorar en un conjunto de validación.
- Esto previene el sobreajuste y ahorra tiempo computacional.

Sobreajuste

- Ocurre cuando el modelo aprende demasiado bien los detalles y el ruido del conjunto de datos de entrenamiento, pero pierde capacidad de generalizar a datos nuevos.
- Señales de sobreajuste:
- Alta precisión en el entrenamiento, pero baja en validación o prueba.

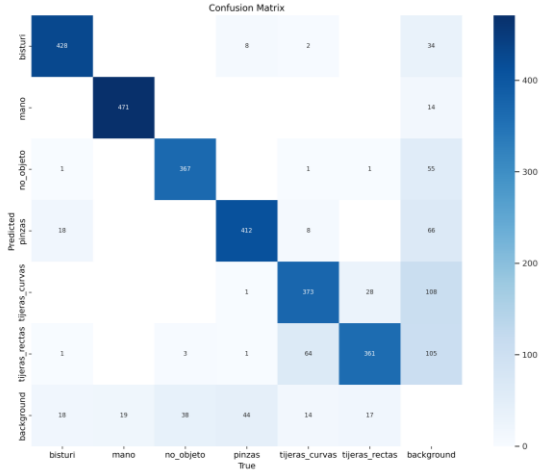


Resultados y Modelos Obtenidos

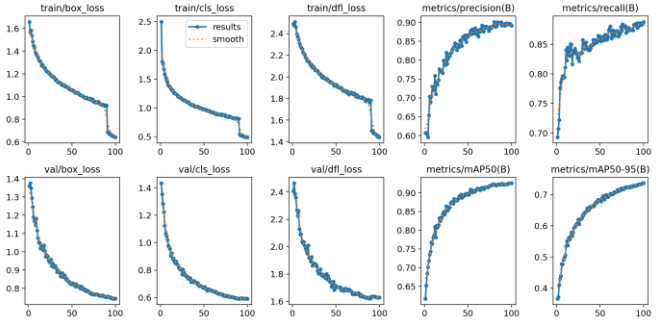
- **Evaluación del modelo:**
- Las métricas más comunes para YOLOv8 incluyen:
  - **mAP (Mean Average Precision):** Una medida de la precisión del modelo.
  - **Precisión y Recall:** Evalúan la capacidad del modelo para detectar objetos correctamente.
- **Dónde se almacenan los resultados:**
  - Después del entrenamiento, YOLOv8 guarda los resultados en la carpeta `runs/train`.
  - Los resultados incluyen: Gráficas de precisión, pérdida y mAP. & Pesos del modelo (`best.pt` y `last.pt`).
  - **Gráficas:**
    - *Matriz de confusión:* Es una herramienta que permite ver como se predijo el apartado de test (predicciones contra real).
    - *Gráficas de precisión:* Utilizadas para ver como fue el entrenamiento del modelo y evaluar si existe un sobreajuste.
    - *Curva F1 contra confianza:* La curva F1 muestra cómo cambian precisión y recall según el umbral de confianza, ayudando a encontrar el equilibrio óptimo.

Resultados y Modelos Obtenidos

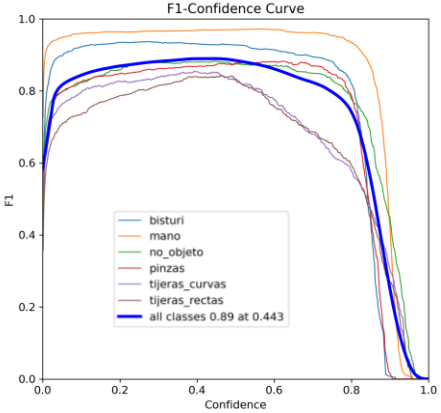
- **Gráficas de mejor modelo (último modelo):**



Matriz de confusión



Gráficas de precisión



Curva F1 - confianza

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/tree/main/V1.0/Train/runs_intsoob/obb/UR5_DATAmV7_100

Entrenamiento (Código)

```python
from ultralytics import YOLO

# Build a YOLOv9c model from scratch
#model = YOLO('yolov9c.yaml')
# Build a YOLOv9c model from pretrained weight
model = YOLO("yolov8m-obb.pt")
# Train the model on the COCO8 example dataset for 100 epochs
results = model.train(data='data.yaml', epochs=100, imgsz=640, name='UR5_DATAmV4_100', device='0', batch=15, workers=10, patience=8)
```

Jupyter notebook para entrenamiento
https://www.kaggle.com/datasets/jphajp/ur5e-srube-nurse-surgical-instruments/data

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V1.0/Train/train.py

Comunicación de UR mediante Python (RTDE)

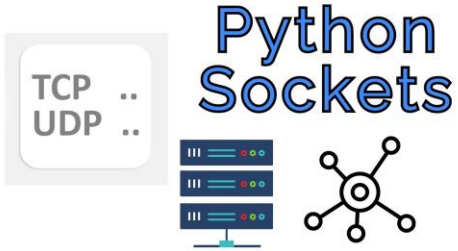
Presione para seguir el vínculo
https://sdurobotics.gitlab.io/ur_rtde/index.html

Comunicación ethernet

Para la comunicación con el robot se utiliza el protocolo de red ethernet. Para lograr esto en un primer momento se investigo el funcionamiento de los puertos del robot y se estableció una conexión socket al puerto de comunicación secuencial.

El código de client.py hace esta conexión para enviar y recibir cadenas de caracteres (strings) con los comandos codificados para realizar los movimientos del robot o leer su estado.

Posteriormente se realizo un cambio de los scripts de control y de comunicación para utilizar la librería de UR-RTDE que permite una comunicación en tiempo real con el robot utilizando puertos especializados para ello; esta comunicación se realiza de manera sencilla ya que la librería ofrece herramientas para un control simplificado en Python evitando la codificación de la información.



Socket y codificación de la información

El código de client.py hace la conexión entre la computadora para enviar y recibir cadenas de caracteres (strings) con los comandos codificados para realizar los movimientos del robot o leer su estado.

```python
#Communication between the robot and python
import socket
import numpy as np
import struct
from time import sleep

def getStatus(): ...
    return x, y, z, rx, ry, rz, angle[5]

def send_instruction_to_ur5(ip, port, instruction): ...
        print("Timeout, can't connect")

def freedrive(ur5_ip, ur5_port):
    while True:
        send_instruction_to_ur5(ur5_ip, ur5_port, 'freedrive_mode()\n')
        sleep(1)
```

```python
ur5_ip = "192.168.1.1"
ur5_port = 30002

#Home
Angles_list_0=[-51.9,-71.85,-112.7,-85.96,90,38]
#Convertir a radianes la lista de angulos
Angles_list_0=[np.radians(i) for i in Angles_list_0]

#Saludar variable
Angles_list_1=[-106.75,-100.55,-79.78,-3.93,152.10,38]
Angles_list_2=[-241.02,2.18,-59.69,-123.5,290.52,-315.06]
#Convertir a radianes la lista de angulos
Angles_list_1=[np.radians(i) for i in Angles_list_1]
Angles_list_2=[np.radians(i) for i in Angles_list_2]

ur5_move_command_1 = "movel(p[.117,-.364,.375,0,3.142,0], a = 1.2, v = 0.25, t = 0, r = 0)\n"
ur5_move_command_2 = "movel(p[.064,-.269,.325,2.471,1.940,0], a = 1.2, v = 0.25, t = 0, r = 0)\n" #Origen 2 para pruebas
ur5_move_command_3 = "movel(p[.064,-.269,.285,3.139,0.126,0], a = 1.2, v = 0.25, t = 0, r = 0)\n" #Origen 3 para pruebas
ur5_move_command_4 = "movel(p[-.060,-.270,.284,1.388,2.819,0], a = 1.2, v = 0.25, t = 0, r = 0)\n" #Origen 4 para pruebas
ur5_move_command_5 = "movel(p[-.174,-.286,.281,0.204,3.135,0], a = 1.2, v = 0.25, t = 0, r = 0)\n" #Origen 5 para pruebas
ur5_move_command_6 = "movel(p[.064,-.269,.400,0,3.142,0], a = 1.2, v = 0.25, t = 0, r = 0)\n" #Origen 6 para pruebas
```

Socket y codificación de la información

Resumen de la función ***getStatus***

Esta función se conecta a un robot UR5e mediante un socket TCP (IP: 192.168.1.1, puerto: 30002) para recibir datos sobre su estado. Se ejecuta un bucle que procesa tres paquetes de datos:

1. Procesamiento del paquete:
   - Extrae la longitud (packlen), tipo (packtype), y marca de tiempo (timestamp).
   - Si el paquete es tipo 16, analiza los mensajes internos:
     - Tipo 1: Obtiene los ángulos de las 6 articulaciones.
     - Tipo 4: Extrae las coordenadas cartesianas (x, y, z) y de orientación (rx, ry, rz) del extremo del brazo.
2. Depuración:
   - Imprime el tipo de mensaje y sus datos para monitoreo.
3. Resultado:
   - Devuelve las coordenadas cartesianas, las orientaciones y el ángulo de la sexta articulación.
   - Cierra la conexión tras procesar los datos.

Es útil para monitorear la posición y orientación del robot en tiempo real.

Socket y codificación de la información

Resumen de la función ***send_instruction_to_ur5***
Esta función envía instrucciones al robot UR5 utilizando un socket TCP.

1. Parámetros de entrada:
   - ip: Dirección IP del robot.
   - port: Puerto del robot.
   - instruction: Comando en formato texto para el robot.
2. Funcionamiento:
   - Crea un socket TCP.
   - Se conecta al robot utilizando la IP y el puerto proporcionados.
   - Envía la instrucción codificada en formato UTF-8.
   - Cierra el socket tras enviar el comando.
3. Gestión de errores:
   - Si la conexión es rechazada, muestra el mensaje: "Connection refused".
   - Si hay un tiempo de espera, muestra: "Timeout, can't connect".

Es útil para enviar comandos de control al UR5 de manera sencilla y confiable.

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V0.1/client.py

RTDE (Real-Time Data Exchange)

Explicación del Código con la ***Librería rtde***

Este código utiliza la librería rtde para simplificar y acelerar la comunicación con un robot UR5e mediante interfaces especializadas. A continuación, se detalla su funcionamiento:

1. Importación de módulos:
- rtde_control: Proporciona métodos para enviar comandos al robot.
- rtde_receive: Permite recibir datos en tiempo real del robot.

2. Creación de interfaces:
- receive = rtde_receive.RTDEReceiveInterface("192.168.1.1"):
  - Establece una conexión para recibir datos del robot desde la dirección IP especificada (192.168.1.1).
- control = rtde_control.RTDEControlInterface("192.168.1.1"):
  - Establece una conexión para controlar el robot desde la misma dirección IP.

3. Movimiento del robot:
- control.moveL([.309, -.277, .373, 0, 3.14, 0], 0.5, 0.5):
  - Comando para mover el robot de manera lineal (trayectoria recta) a una posición específica:
    - ❑ Coordenadas cartesianas: [x, y, z] en metros (.309, -.277, .373).
    - ❑ Orientación: [rx, ry, rz] en radianes (0, 3.14, 0).
- Velocidad y aceleración: Ambos configurados a 0.5 m/s y m/s^2, respectivamente.

RTDE (Real-Time Data Exchange)

4. Obtención de datos del robot:

- print(receive.getActualTCPPose()):
  - Obtiene y muestra la posición actual del extremo del brazo del robot (TCP: Tool Center Point) en coordenadas cartesianas [x, y, z, rx, ry, rz].

```
import rtde_control
import rtde_receive


receive = rtde_receive.RTDEReceiveInterface("192.168.1.1")
control = rtde_control.RTDEControlInterface("192.168.1.1")

control.moveL([.309,-.277,.373,0,3.14,0],0.5,0.5)

print(receive.getActualTCPPose())
```

Esta librería permite movimientos síncronos y asíncronos para poder detenerlos o modificar el punto de destino cuando se le dio una orden.

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V1.0/Scripts/client.py

RTDE (Configuración)

Para poder utilizar el robot en esta configuración se deben de seguir los siguientes pasos:

- Asegurarse tener conexión al robot mediante cableado ethernet o equivalente.
- Establecer conexión IP estática (192.168.1.1) del robot (en configuración del robot; ver diapositiva de conexiones) y activar el control remoto.
- Asegurarse que en la pestaña de instalación se desactiven todos los módulos de comunicación (MODBUS, PROFINET, ETHERNET; no se activará la comunicación RTDE con estos módulos activados).
- En la computadora de control asegurarse de establecer una IP estática diferente a la del robot (192.168.1.2)
- Poner el robot en la pestaña de remoto e iniciarlo (el estado debe ser verde y decir “Stop”, si esta en running, detener el script con el botón stop).
- Inicializar el programa de prueba para verificar la conexión (El programa de prueba es saludar.py).

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V1.0/Scripts/saludar.py

RTDE (Ejemplo)

```python
import numpy as np
from time import sleep
import rtde_control
import rtde_receive

control = rtde_control.RTDEControlInterface("192.168.1.1")
receive = rtde_receive.RTDEReceiveInterface("192.168.1.1")

def saludar_continuo(ur5_ip, ur5_port):
    #Saludar variable
    #HOME
    Angles_list_0=[-51.9,-71.85,-112.7,-85.96,90,38]
    Angles_list_1=[-71.96,-90.60,-99.21,2.23,122.58,37.98]
    Angles_list_2=[-17.52,-134.93,-70.15,30.38,68.76,42.41]
    Angles_list_3=[-65.14,-147.04,-70.09,42.23,117.62,44.25]
    Angles_list_4=[-65.21,-65.16,-69.04,-62.73,110.99,37.60]
    #Convertir a radianes la lista de angulos
    Angles_list_0=[np.radians(i) for i in Angles_list_0]
    Angles_list_1=[np.radians(i) for i in Angles_list_1]
    Angles_list_2=[np.radians(i) for i in Angles_list_2]
    Angles_list_3=[np.radians(i) for i in Angles_list_3]
    Angles_list_4=[np.radians(i) for i in Angles_list_4]
    while True:
        for i in range(0, 5):
            variable_name = f"Angles_list_{i}"
            angles
            angles = eval(variable_name)  # Obtener la lista usando eval
            control.movej(angles, 1, 1)
            print(f"{variable_name} en radianes: {angles}")
            print(receive.getActualTCPPose())

#Home
Angles_list_0=[-51.9,-71.85,-112.7,-85.96,90,38]
#Convertir a radianes la lista de angulos
Angles_list_0=[np.radians(i) for i in Angles_list_0]

#Ejemplo movimiento lineal
ur5_move_command_1 = "movel(p[.117,-.364,.375,0,3.142,0], a = 1.2, v = 0.25, t = 0, r = 0)\n"
#Ejemplo movimiento por angulos
ur5_move_command_7 = f"movej([{Angles_list_0[0]}, {Angles_list_0[1]}, {Angles_list_0[2]}, {Angles_list_0[3]}, {Angles_list_0[4]}, {Angles_list_0[5]}], a=1, v=1)\n"

#Comando para activar el modo de control manual
ur5_move_command_10 = 'freedrive_mode()\n'

saludar_continuo()
```

Asignación de dirección IP del robot.

Lista de ángulos y conversión a radianes (ángulos dados como se ven en el Teach pendant).

Ejemplos de comandos movel y movej.

Presione para seguir el vínculo https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V1.0/Scripts/saludar.py

Conexiones físicas en UR5e

Presione para seguir el vínculo
https://jorgensen.pl/uploads/offer/Robotics/UR/OEM_Control_Box_Installation_Guide_en_20190815.pdf

Descripción de Inputs y Outputs del Sistema con el Robot UR5e

Hardware Conectado

1. *Electroimán:*

Propósito: Permite sujetar objetos ferromagnéticos desde el TCP.

Conexión:

- Alimentado por un circuito de potencia de 24VDC.
- Controlado por un relevador que actúa como intermediario entre el robot y el electroimán.
- Activación/desactivación gestionada por las salidas digitales del UR5e (I/O), asegurando aislamiento eléctrico y evitando sobrepasar la corriente.

2. *Final de Carrera:*

Propósito: Detecta contacto físico del TCP con una superficie u objeto.

Conexión:

- Montado en la punta del TCP junto con el electroimán.
- Se acciona mediante un resorte cuando se presiona.
- Conectado a las entradas digitales del UR5e (I/O) para enviar señales al controlador del robot.

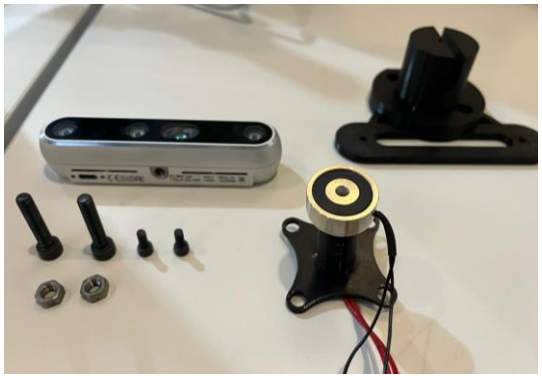
Descripción de Inputs y Outputs del Sistema con el Robot UR5e

3. Cámara Intel RealSense:

Propósito: Captura imágenes o nubes de puntos para aplicaciones de visión computacional.

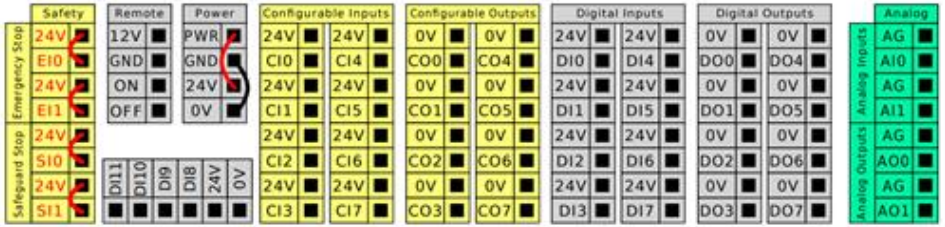
Conexión:

- Físicamente montada en el TCP del UR5e.
- Conectada directamente a la computadora de control mediante USB.
- Gestionada desde Python para análisis de visión independiente del controlador del robot.



Gripper desarmado

Inputs/Outputs



Descripción del diagrama de I/O del UR5e
El diagrama muestra las entradas y salidas disponibles en el controlador del robot UR5e. A continuación, se detalla cada sección y su uso, tanto desde el Teach Pendant como desde una conexión remota con ur_rtde:

Safety (Paro de emergencia):

- Pines SI0, SI1, EI0, EI1 están diseñados para integrar dispositivos de seguridad externos, como botones de parada de emergencia o cortinas de seguridad.
- Estas entradas aseguran que el sistema pueda detenerse de manera segura cuando sea necesario.

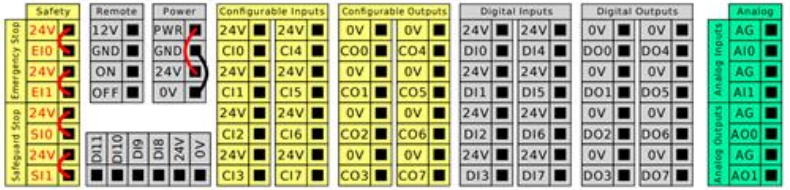
Remote (Arranque remoto):

- Pines de 12V, GND, ON y OFF.
- Permiten encender o apagar el controlador de manera remota mediante un circuito externo.

Power (Alimentación):

- Pines de 24V y GND.
- Proporcionan energía para dispositivos externos conectados a los pines de I/O.

Inputs/Outputs



Entradas configurables (CI0-CI7):

- Pines que reciben señales digitales de dispositivos externos, como sensores.
- Pueden configurarse desde el Teach Pendant para diferentes propósitos.

Salidas configurables (CO0-CO7):

- Pines que envían señales digitales a dispositivos externos, como relevadores o luces indicadoras.
- Configurables desde el Teach Pendant o por software.

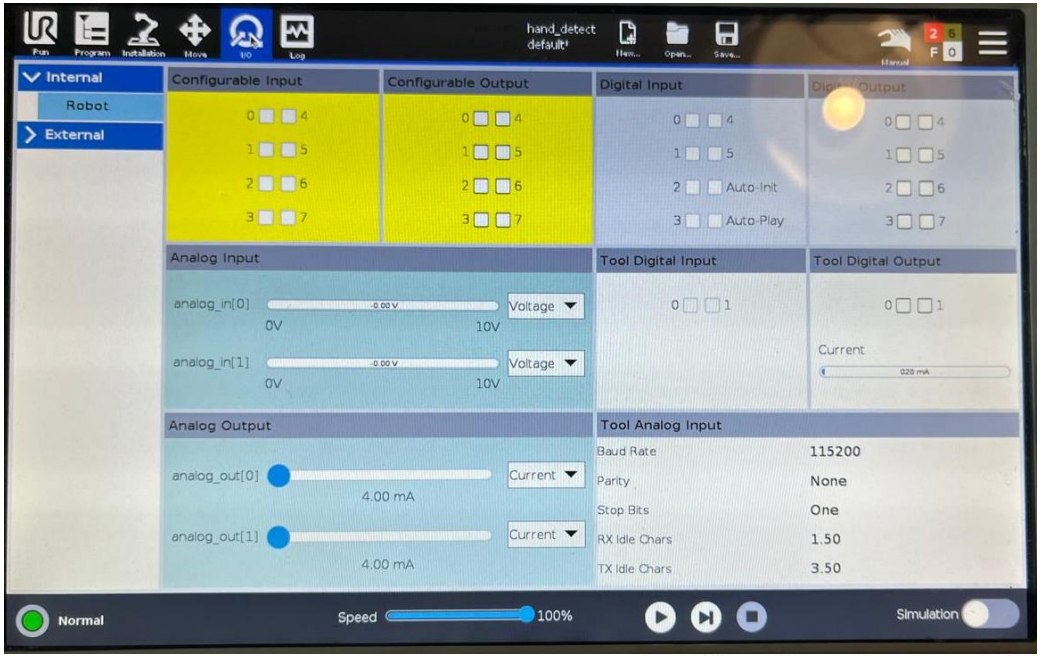
Digital Inputs/Outputs (Entradas y salidas digitales):

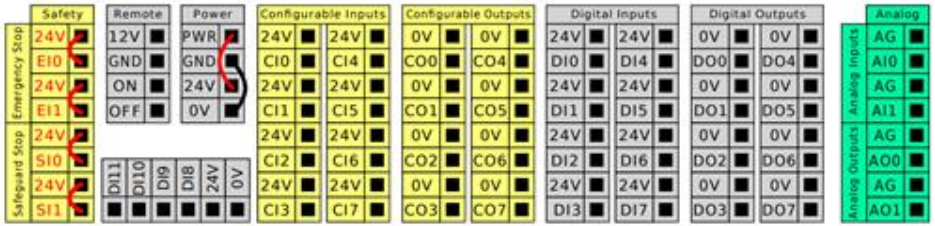
- Entradas digitales (DI0-DI17): Reciben señales de dispositivos externos (contactos, finales de carrera, etc.).
- Salidas digitales (DO0-DO7): Envían señales de control de 24V o 0V a actuadores externos, como un electroimán o un motor.

Analog Inputs/Outputs (Entradas y salidas analógicas):

- Entradas analógicas (AI0-AI1): Reciben señales analógicas, como datos de sensores (0-10V o 4-20mA).
- Salidas analógicas (AO0-AO1): Generan señales analógicas para controlar dispositivos externos (0-10V o 4-20mA).

Inputs/Outputs





Sistema de montado rápido

Se desarrolló un sistema que permite intercambiar herramientas en el robot de manera rápida y eficiente. Este diseño incluye:

- Base: Parte fija conectada al TCP del robot, con conexiones listas para facilitar el intercambio.
- Cople: Pieza intermedia que une la herramienta a la base, asegurando precisión y fácil alineación.
- Pestillo: Mecanismo de bloqueo que asegura la herramienta al sistema, garantizando estabilidad y facilidad de montaje/desmontaje.

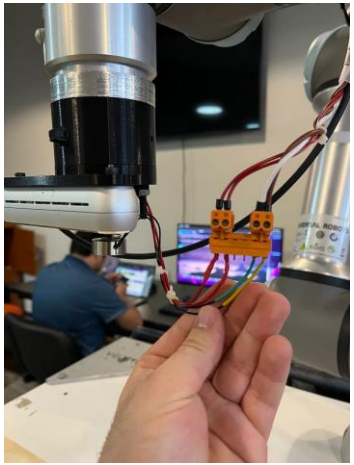
Ventajas

- Rapidez: Herramientas intercambiables en segundos.
- Conexiones integradas: Eléctricas y neumáticas listas para usar.
- Versatilidad: Compatible con múltiples herramientas.
- Seguridad: Bloqueo confiable para evitar desajustes.

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/tree/main/3D_Models

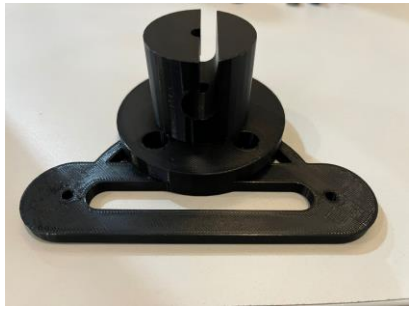
Sistema de montado rápido

Base en robot

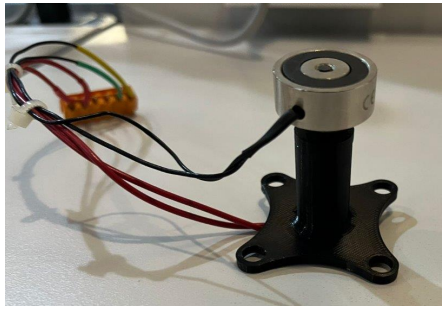


Conexiones rápidas

Sistema montado



Base en robot



Electroimán y final de carrera

Sistema armado

Cámara Intel RealSense

Presione para seguir el vínculo
https://www.intelrealsense.com/download/21345/?tmstv=1697035582

Intel RealSense SDK 2.0

Presione para seguir el vínculo
https://www.intelrealsense.com/developers/

Conexión con Python

Instalación de dependencias

1. Instala el SDK de Intel RealSense en tu sistema
   - sudo apt-get install librealsense2-dkms librealsense2-utils
2. Instala la biblioteca de Python
   - pip install pyrealsense2
3. Implementación

Revisa el siguiente ejemplo

Código con Python

```
####### EN ESTE CODIGO SE TOMARAN LAS FOTOS PARA EL DATASET #######
import cv2
import pyrealsense2 as rs
import numpy as np
import socket
from time import sleep

import rtde_control
import rtde_receive

control = rtde_control.RTDEControlInterface("192.168.1.1")
receive = rtde_receive.RTDEReceiveInterface("192.168.1.1")


# configurar el pipeline
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

# configurar UR5
ur5_ip = "192.168.1.1"
ur5_port = 30002

# Configurar angulos
#MEDIO (HOME)
Angles_list_0=[-51.9,-71.85,-112.7,-85.96,90,38]
#Convertir a radianes la lista de angulos
Angles_list_0=[np.radians(i) for i in Angles_list_0]
```

Código con Python

```python
#BAJO
Angles_list_1=[-51.85,-86.16,-136.82,-47.73,90,38]
#Convertir a radianes la lista de angulos
Angles_list_1=[np.radians(i) for i in Angles_list_1]

#ALTO
Angles_list_2=[-51.9,-82.72,-60.07,-127.72,90,38]
#Convertir a radianes la lista de angulos
Angles_list_2=[np.radians(i) for i in Angles_list_2]


ur5_move_command_0 = f"movej([{Angles_list_0[0]}, {Angles_list_0[1]}, {A
ur5_move_command_1 = f"movej([{Angles_list_1[0]}, {Angles_list_1[1]}, {A
ur5_move_command_2 = f"movej([{Angles_list_2[0]}, {Angles_list_2[1]}, {A
control.moveJ(Angles_list_1, 1, 1)

# inicio de la captura de la pantalla
profile = pipeline.start(config)

# crear un contador para las imagenes
count = 0
mov = 0
dir_name = "dataset"
class_name = "fist"
max_images = 100
```

Código con Python

```
try:
    dir_name = input("Ingrese el nombre de la carpeta: ")
    class_name = input("Ingrese el nombre de la clase: ")
    max_images = int(input("Ingrese el numero de imagenes: "))


# crear un while para capturar la imagen
    while max_images > count:
        frames = pipeline.wait_for_frames()
        color_frame = frames.get_color_frame()
        if not color_frame:
            continue
        frame = np.asanyarray(color_frame.get_data())
        # leer la imagen
        """ ret, frame = cap.read() """
        # agregar un filtro de escalas de grises
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        resized = cv2.resize(gray, (480, 480), interpolation=cv2.INTER_A
        # mostrar la imagen
        # cv2.imshow("Captura", frame)
        cv2.imshow("Captura", resized)
        # tomar una foto si se presiona la tecla "c" y guardarla en la c
        if cv2.waitKey(1) & 0xFF == ord('c'):
            count += 1
            file_name = f"dataset/{dir_name}/{class_name}_{count}.jpg"
            cv2.imwrite(file_name, resized)
            print(f"Imagen guardada en {file_name} - {count}/{max_images
        # si se presiona la tecla "q" se cierra la ventana
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

Código con Python

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        if cv2.waitKey(1) & 0xFF == ord('p'):
            mov = mov + 1
            print(mov)
            if mov == 1:
                print("Moviendo a la posición 1")
                control.moveJ(Angles_list_1, 1, 1)

            elif mov == 2:
                print("Moviendo a la posición 2")
                control.moveJ(Angles_list_2, 1, 1)

            else:
                print("Moviendo a la posición 0")
                control.moveJ(Angles_list_0, 1, 1)
                mov = 0

finally:
    pipeline.stop()
    cv2.destroyAllWindows()
# liberar la camara y cerrar la ventana

""" cap.release() """
cv2.destroyAllWindows()

################################## EN ESTE CODIGO SE TOMARAN LAS FOTO
```

Medición de profundidad

Esto nos ayuda para calcular la profundidad en donde se encuentran los objetos en milímetros

Implementación

Revisa el siguiente ejemplo

Transformación de coordenadas

Transformación de Coordenadas para el Robot UR5e

• **Objetivo:** Convertir coordenadas en píxeles de la cámara Intel RealSense a coordenadas cartesianas en el sistema del robot UR5e.
• **Importancia:** Permite al robot identificar con precisión la ubicación de un objeto en su espacio de trabajo físico.
• **Metodología:**
    - Calcular el tamaño físico del frame de la cámara.
    - Aplicar rotaciones y traslaciones para alinear los sistemas de coordenadas.
    - Mapear coordenadas en píxeles a posiciones reales.

Tamaño del Frame de la Cámara

- **Dimensiones del Frame:**
  - Fórmulas:
  - $H_{cam} = distance \cdot \tan(23°)$
  - $V_{cam} = distance \cdot \tan(83.96°)$
  - **Descripción:** Calcula el tamaño físico del frame de la cámara según la altura actual (distance).
- **Densidad de Píxel:**
  - Mapear los píxeles (640 x 480) al tamaño físico calculado asegura precisión dinámica.

Rotación y Traslación de Coordenadas

Rotación:

- Matriz de rotación en Z (R_z):

$$R_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Ajusta el sistema de la cámara al robot.

Traslación:

- Traslada el origen del sistema de la cámara al del robot.
- Se combina con R_z en una matriz homogénea para simplificar.

Conversión de Píxeles a Coordenadas Físicas

- Fórmula:

$$valor\_real = ((valor - \min\_pix) \cdot (\max\_real - \min\_real)/(\max\_pix - \min\_pix)) + \min\_real$$

- Función: Convierte las coordenadas de píxeles (x,y) en posiciones físicas según las dimensiones del frame calculadas en el paso 1.
- Propósito: Localizar objetos detectados en términos que el robot pueda interpretar.

Coordenadas Finales del Objeto

- Transformación Total:
- Aplica rotaciones y traslaciones calculadas en transformCoordinates.
- Integra el origen del frame ajustado con frameOriginCoordinates.
- Resultado: Coordenadas cartesianas físicas del objeto en el sistema del robot.
- Ejemplo:
  - Entrada: Coordenadas en píxeles (320,240).
  - Salida: Coordenadas reales (xrobot, yrobot) en metros.

Detalle de la Rotación (transformCoordinates)

- Fórmula de Rotación en R_z:
  - xrobot=H0f[0,0]·x1+H0f[0,1]·y1+xorigen
  - yrobot=H0f[0,0]·x1+H0f[0,1]·y1+yorigen
- Propósito: Ajusta el ángulo del sistema de coordenadas de la cámara para alinearlo con el del robot.

Detalle de la Traslación (frameOriginCoordinates)

•**Fórmulas:**

- Posición del origen:

  $ofx = cfx - (Hcam/2) \cdot \cos(\text{ángulo tangente}) - (Vcam/2) \cdot \sin(\text{ángulo tangente})$
- $ofy = cfy - (Hcam/2) \cdot \cos(\text{ángulo tangente}) - (Vcam/2) \cdot \sin(\text{ángulo tangente})$

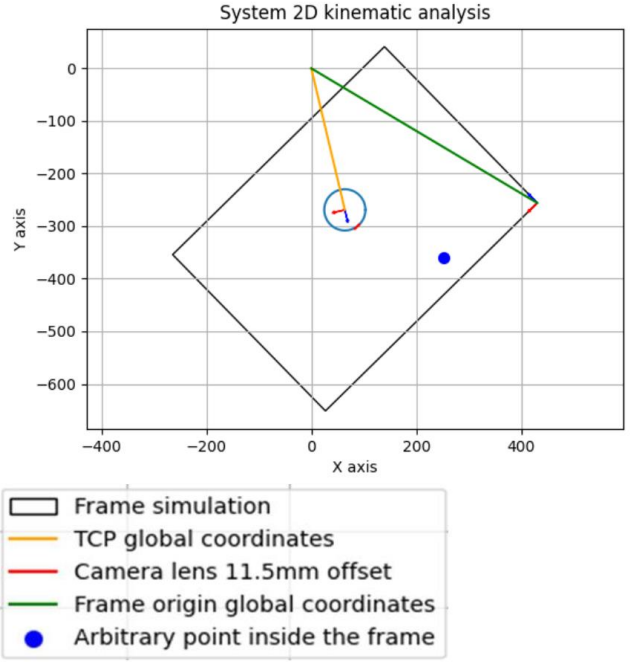
•**Propósito:** Calcula la ubicación del origen del marco de referencia de la cámara en coordenadas del robot.

Relación entre transformCoordinates y frameOriginCoordinates

- frameOriginCoordinates: Calcula el origen del marco de la cámara en el sistema del robot.
- transformCoordinates: Usa el origen calculado para transformar puntos individuales detectados en el marco de la cámara.

Ejemplo de Transformación

- Entrada: Objeto detectado en coordenadas de píxeles (320,240).
- Proceso:
  - Calcular Hcam y Vcam según la profundidad.
  - Ajustar origen del frame con frameOriginCoordinates.
  - Transformar coordenadas del objeto con transformCoordinates.
- Salida: Coordenadas físicas para el robot (xrobot,yrobot).



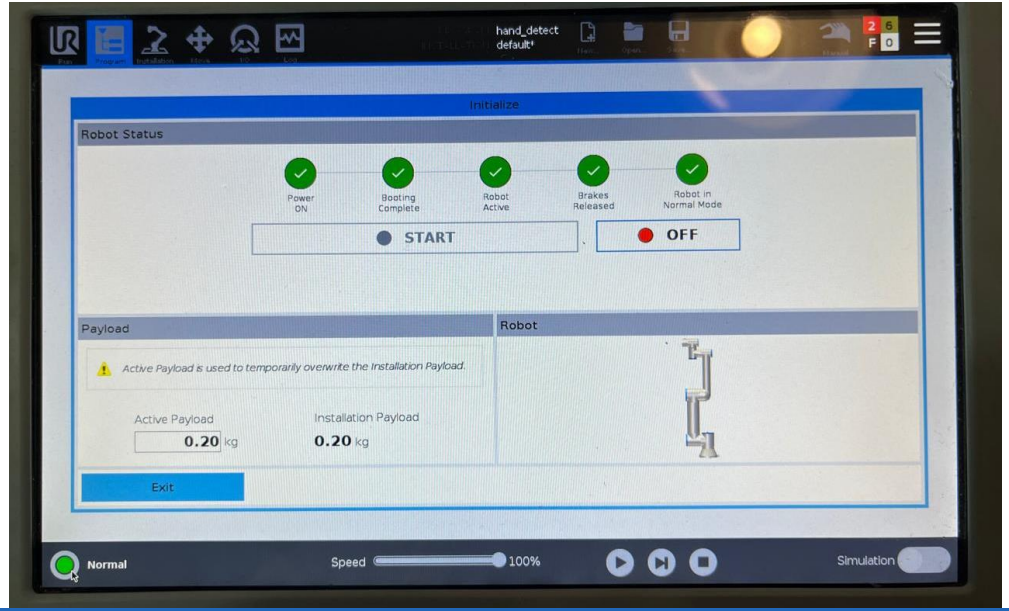
Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/UR.py

Manejo del robot (teach-pendat)

Inicio de robot

Para iniciar el robot se debe presionar en la parte inferior derecha, se mostrara el estado del robot y la opción para encenderlo o activarlo; apagarlo. Se debe presionar iniciar dos veces (una para iniciar programa y otra para iniciar robot y quitar frenos).

En caso de paro de emergencia, después de desactivar el paro, se debe de regresar a esta pantalla para reiniciar el robot.



Movimiento del robot (manual)

La interfaz gráfica del Teach Pendant de Universal Robots permite realizar movimientos manuales del robot UR5e de manera intuitiva. A continuación, se describe cómo se controla el robot para movimientos no programados:

Opciones de Movimiento

1. Control de la Posición del TCP (Tool Center Point):

- Las flechas en la sección "TCP Position" permiten mover el extremo del brazo robótico (TCP) en las direcciones X, Y, Z.
- Los botones controlan los desplazamientos lineales en las coordenadas cartesianas del TCP.

2. Control de la Orientación del TCP:

- Las flechas curvas en "TCP Orientation" permiten ajustar la orientación del TCP en los ejes de rotación Rx, Ry, Rz.
- Estos controles giran la herramienta montada en la punta para ajustar su orientación angular.

Movimiento del robot (manual)

3. "Free Drive":

- El botón "Free Drive" activa un modo en el que el brazo robótico puede moverse manualmente con la mano del operador.
- Este modo desactiva temporalmente los motores, permitiendo un posicionamiento libre y suave.

4. Control de Juntas (Joint Position):

- La columna "Joint Position" permite ajustar manualmente el ángulo de cada una de las articulaciones del robot (Base, Shoulder, Elbow, Wrist 1, Wrist 2, Wrist 3).
- Se puede usar para posicionar el robot manipulando cada articulación individualmente.

5. Indicadores y ajustes

- Velocidad: El control deslizante en la parte inferior ajusta la velocidad de los movimientos manuales (porcentaje del máximo).
- Modo Normal / Simulación: El robot puede moverse físicamente o en un entorno de simulación, dependiendo de la configuración seleccionada.

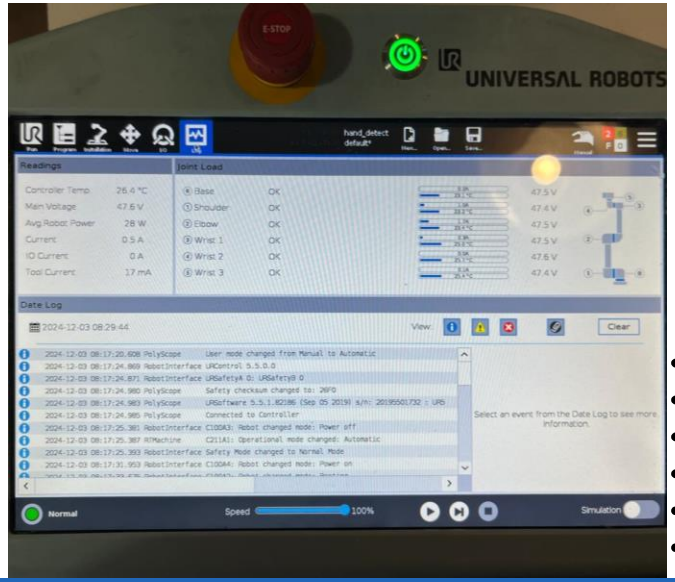
Registro de comandos

El Teach Pendant del UR5e muestra un registro detallado de las operaciones realizadas por el robot, tanto manuales como remotas. En la pantalla observamos:

1. Date Log (Parte inferior):
- Muestra un historial de eventos y comandos:
  - Comandos enviados desde software remoto (como ur_rtde).
  - Cambios de modo (Manual/Automático).
  - Ejecución de programas o movimientos.
- Incluye mensajes de información, advertencias o errores.
- Permite filtrar por tipo de evento para un análisis rápido.

2. Lecturas (Readings):
- Información del sistema en tiempo real:
  - Temperatura del controlador, voltaje y corriente consumida.
  - Estado energético del robot.

3. Carga por Articulación (Joint Load):
- Estado de las articulaciones (Base, Shoulder, Elbow, etc.).
- Porcentaje de carga aplicado en cada una.



Configuración de instalación del robot

Instalación

En el **Teach Pendant** del robot UR5e, se puede acceder al menú de **"Instalación"** para configurar parámetros esenciales del robot. Estas configuraciones incluyen límites de movimiento, posiciones seguras, interfaces de comunicación, entre otros. Para realizar cambios, es necesario que el robot esté en **modo manual** y se ingrese la contraseña **"IBERO"**.

Aspectos clave para el proyecto:

1. Desactivación de Interfaces de Comunicación:

1. Todas las interfaces de comunicación como **MODBUS**, **PROFINET** y **ETHERNET/IP** deben desactivarse.
2. Esto es imprescindible para que el robot pueda operar en modo **RTDE** y ser controlado desde un script externo.

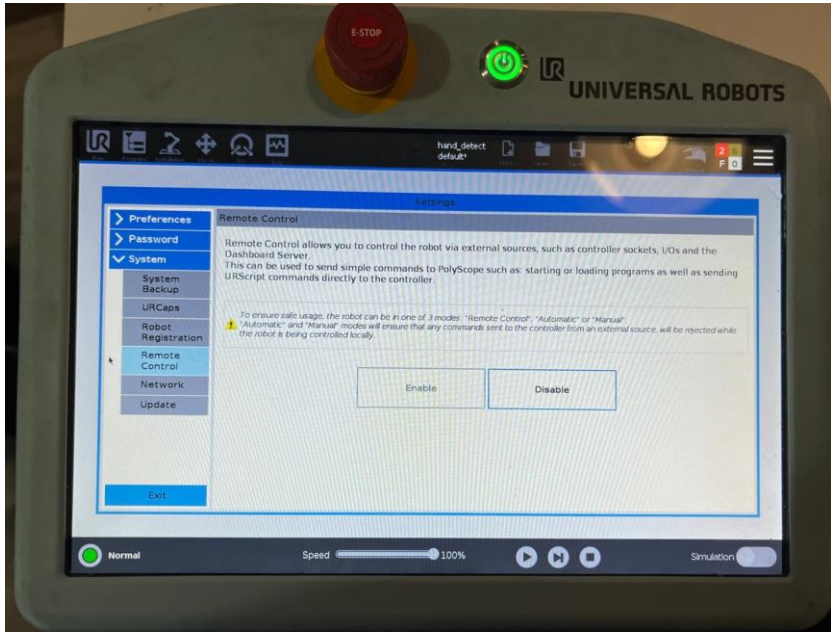
2. Configuración de la IP:

1. Aunque se desactive la interfaz de **ETHERNET**, el robot debe tener asignada una dirección **IP estática**.
2. Esta configuración no se encuentra en el menú de instalación, sino en el apartado de **Configuraciones Generales** del robot.

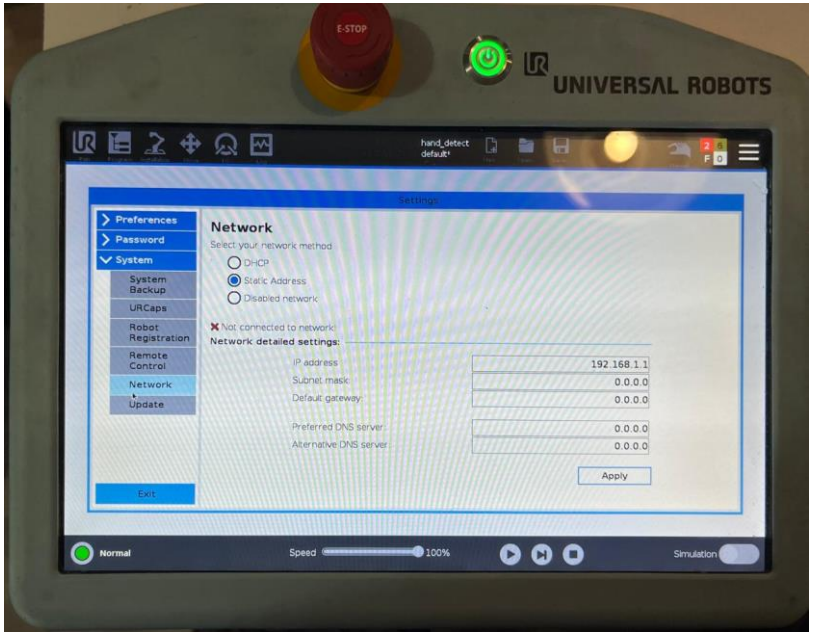
Nota: Para la activación automática del robot y para el control con RTDE se debe de regresar a modo remoto (modos disponibles: remoto, automático y manual).

Instalación

Ajustes generales del robot



Activación de control remoto



Asignación de dirección IP

Nota: La computadora de control debe de tener otra dirección IP, se recomienda que sea estática; para el proyecto se asigno que la conexión alámbrica tuviera la dirección 192.168.1.2

Instalación

Tool Center Point (TCP)

•**Propósito**: Configurar la posición y orientación del TCP, que representa el punto de interacción de la herramienta montada en el robot.

•**Funciones principales**:

- Ajustar la posición del TCP en coordenadas X, Y, Z.
- Configurar la orientación en rotaciones Rx, Ry, Rz.
- Definir el peso y el centro de gravedad de la herramienta montada.

Robot Mounting and Angle

•**Propósito**: Configurar el montaje y la orientación inicial del robot.

•**Funciones principales**:

- Ajustar el ángulo y la posición del robot según su montaje físico (por ejemplo, en pared, techo o suelo).
- Ver una representación gráfica del robot para validar los cambios.

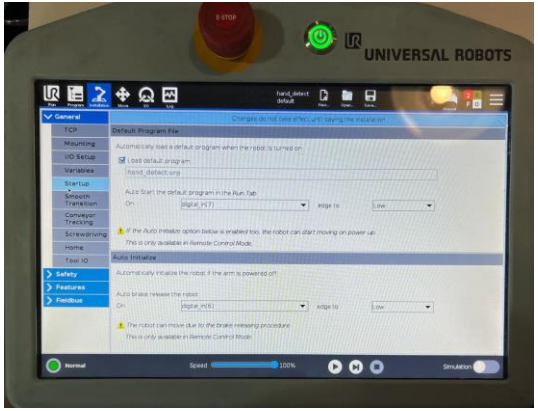
Instalación

Configuración de Inicio Automático

•**Propósito**: Configurar el comportamiento del robot al iniciar o reiniciar el sistema.

•**Funciones principales**:

- Permitir que el robot se mueva automáticamente a una posición inicial al encenderse.
- Configurar la posición inicial del robot (Home).
- Activar/desactivar la función de inicio automático para evitar movimientos no deseados.

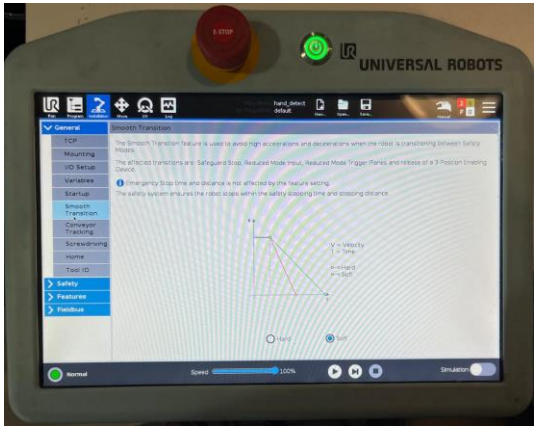


Transición de Modos de Velocidad

•**Propósito**: Configurar la transición entre modos de velocidad para garantizar movimientos suaves al cambiar entre velocidad normal y reducida.

•**Funciones principales**:

- Definir la distancia de transición entre modos.
- Configurar velocidades y tiempos de reacción para evitar movimientos bruscos.
- Visualizar gráficamente cómo se realiza la transición para validarla antes de aplicarla.



Instalación

Configuración de Home (Posición Inicial)

•**Propósito**: Definir y gestionar la posición inicial o de referencia del robot.

•**Funciones principales**:

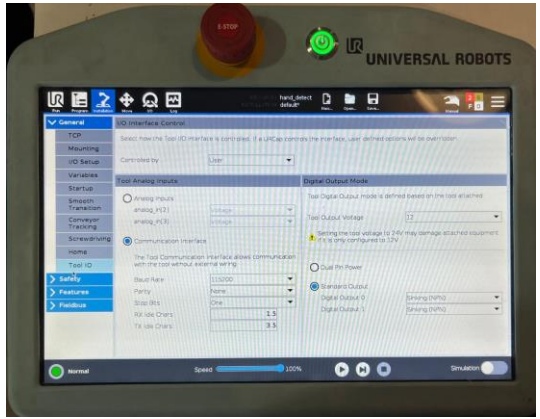
- **Editar posición**: Permite ajustar las coordenadas articulares para definir una nueva posición inicial.
- **Eliminar**: Borra la configuración actual de Home.
- **Mover a Home**: Envía el robot a la posición configurada como Home.

Configuración de Interfaces I/O

•**Propósito**: Configurar las entradas y salidas analógicas y digitales del robot.

•**Funciones principales**:

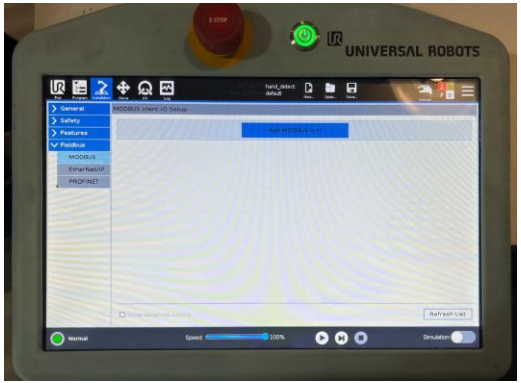
- Ajustar el rango de las entradas analógicas (0-10V o 4-20mA).
- Configurar el modo de salida digital (por ejemplo, activación por pulsos o constantes).
- Asignar funciones específicas a las señales de I/O para interactuar con periféricos externos.

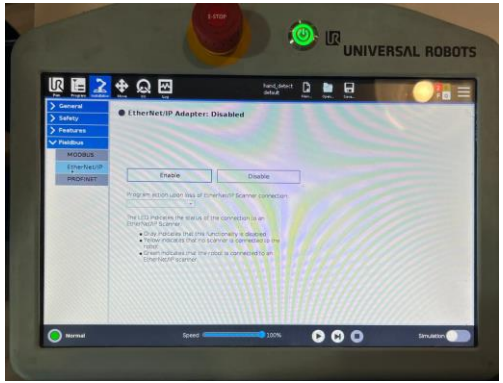


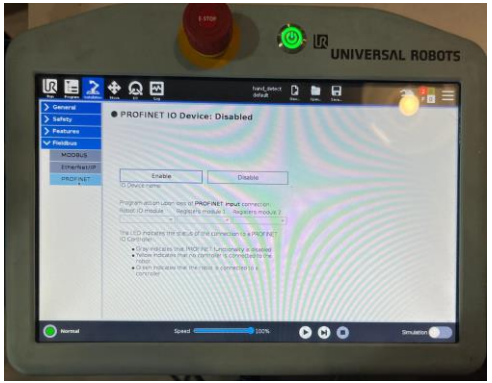
Instalación

Ajustes de comunicación de robot (apartado de instalación)

En este apartado todos los métodos de comunicación deben de estar desactivados como se muestra en las imágenes; esto es necesario para poder usar la comunicación mediante RTDE.





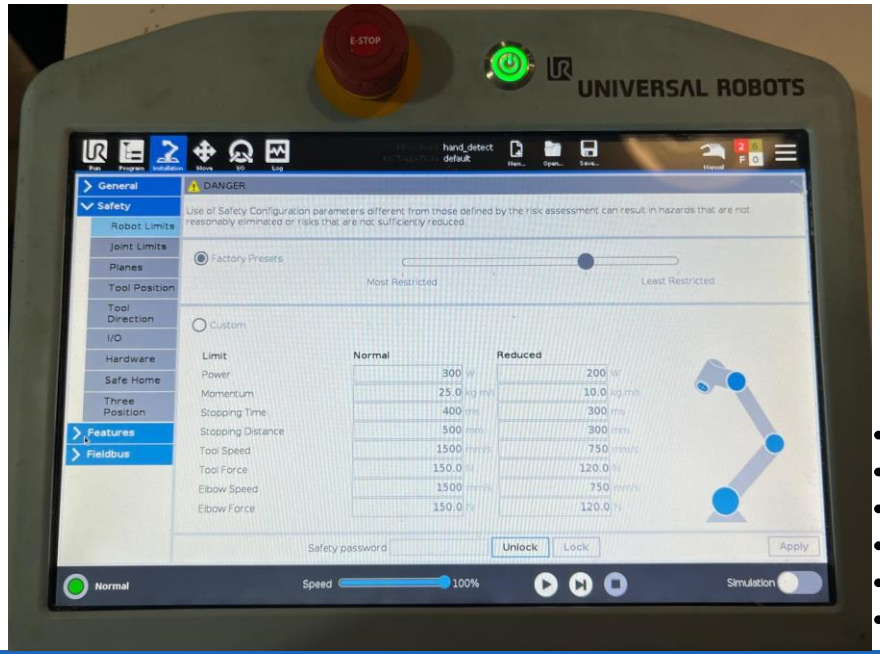


Parámetros de seguridad UR5e para proyecto

Límites del robot

Esta pantalla permite configurar los límites de seguridad del robot, como potencia, velocidad, fuerza y distancias de frenado. Incluye:

- Modos de configuración:
  - "Factory Presets": Configuraciones predeterminadas de fábrica.
  - "Custom": Configuraciones personalizadas por el usuario.
- Parámetros ajustables: Potencia, momentum, tiempo y distancia de frenado, velocidad de la herramienta y fuerza en las articulaciones.
- Opciones de restricción: Permite elegir entre configuraciones más restrictivas o menos restrictivas.
- Controles adicionales: Botones para desbloquear o bloquear los parámetros mediante contraseña de seguridad.



Límites de articulaciones

Esta pantalla permite configurar los límites de movimiento y velocidad de las articulaciones del robot para garantizar la operación segura. Se divide en dos secciones principales:

1. Position Range (Rango de Movimiento):

- Muestra el rango permitido de cada articulación (Base, Shoulder, Elbow, Wrist 1, 2, 3).
- Configurable en dos modos:
  - **Normal Mode**: Límites estándar para operación regular.
  - **Reduced Mode**: Límites más estrictos para entornos más controlados.

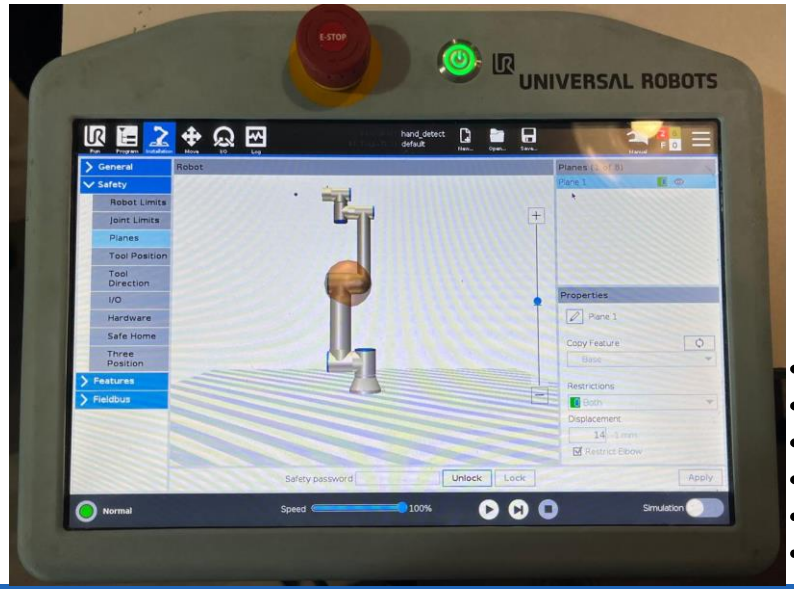
2. Maximum Speed (Velocidad Máxima):

- Establece la velocidad máxima de cada articulación en grados/segundo.
- Configurable para los modos:
  - **Maximum**: Valor límite máximo permitido.
  - **Normal Mode**: Velocidad regular.
  - **Reduced Mode**: Velocidad reducida para mayor seguridad.

Planos

Esta pantalla permite definir y configurar planos de seguridad virtuales que restringen el movimiento del robot en el espacio de trabajo para evitar colisiones o ingresar a zonas prohibidas. Los elementos clave son:

1. **Visualización del Robot**:
   - Representación 3D del robot y del plano de seguridad configurado.
   - Permite observar la interacción del robot con el plano en tiempo real.
2. **Planes**:
   - Lista de planos definidos (ejemplo: "Plane 1").
   - Cada plano puede configurarse individualmente.
3. **Propiedades del Plano**:
   - **Copiar característica (Copy Feature)**: Permite duplicar propiedades de otros planos.
   - **Restricciones**:
     - **Active**: Activa o desactiva el plano.
     - **Displacement**: Define la distancia o ubicación del plano respecto a un punto de referencia.
     - **Restrict Elbow**: Restringe el movimiento del codo si cruza el plano.



Posición de herramienta

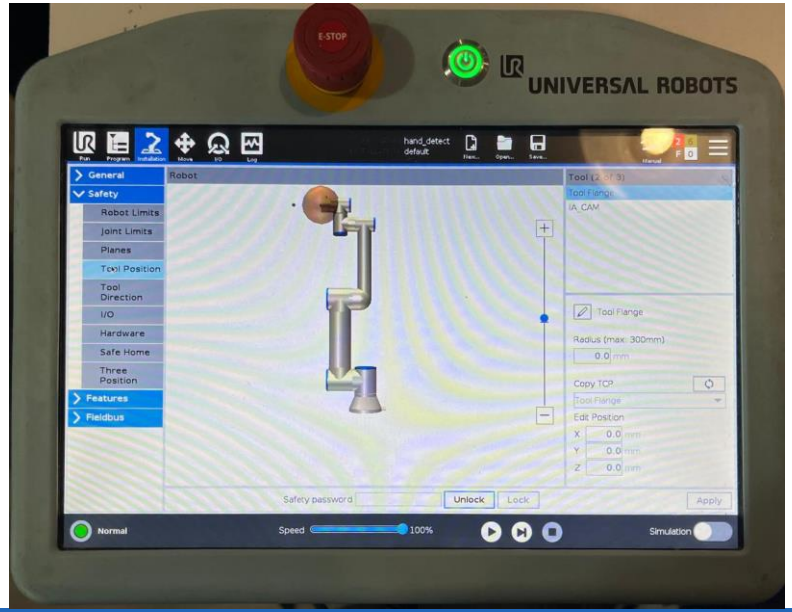
Esta pantalla permite definir la posición y el rango de alcance de la herramienta instalada en el TCP.
Incluye:

•**Visualización 3D**: Representa el robot con la herramienta configurada.

•**Parámetros ajustables**:

- **Radio máximo**: Límite de alcance de la herramienta (hasta 300 mm).
- **Posición (X, Y, Z)**: Ajusta la ubicación exacta de la herramienta.
- **Copiar propiedades**: Replica configuraciones de otras herramientas.

Esta configuración asegura precisión en los movimientos y evita colisiones al considerar el tamaño de la herramienta.



Dirección de herramienta

Esta pantalla permite definir y ajustar la orientación y restricciones de movimiento de la herramienta instalada en el TCP. Incluye:

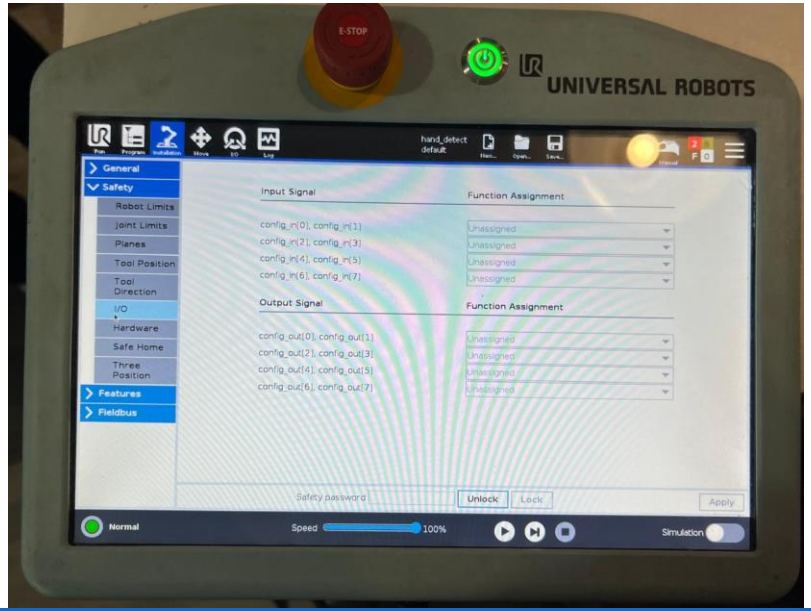
- **Propiedades de límites**:
  - **Restricciones**: Configura el rango angular permitido (por ejemplo, desviación máxima de 160°).
- **Propiedades de la herramienta**:
  - Ajustes de dirección (Pitch, Pan) para definir su orientación.

Es útil para garantizar precisión en movimientos específicos y evitar colisiones por orientaciones no deseadas.

I/O

Esta pantalla permite asignar funciones específicas a las señales de entrada y salida configurables del robot. Incluye:

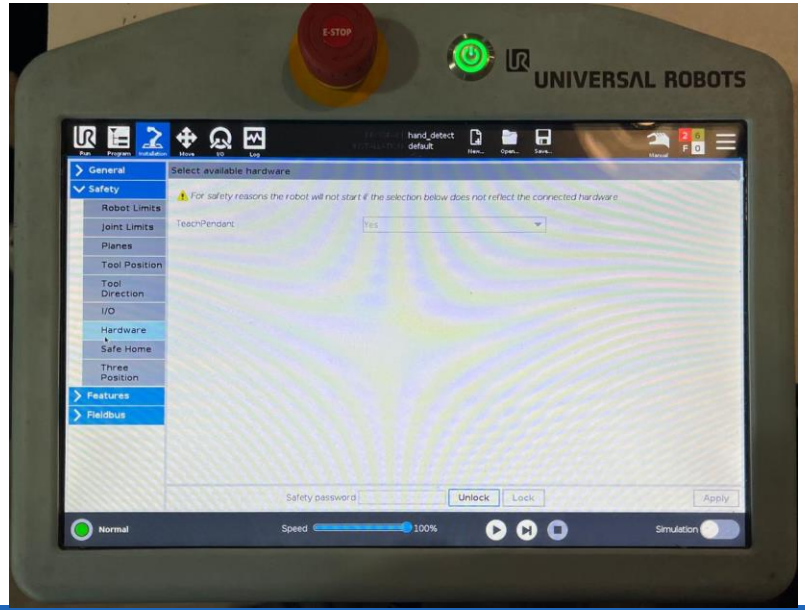
- **Señales de entrada (Input Signal)**:
  - Configuración para vincular entradas a sensores o dispositivos externos.
- **Señales de salida (Output Signal)**:
  - Asignación para controlar actuadores externos como relevadores o luces.



Hardware

Esta pantalla permite configurar el hardware disponible para operar el robot, asegurando que el sistema solo funcione con el hardware conectado. Incluye:

- **Selección de dispositivos**:
  - Ejemplo: Activar o desactivar el uso del Teach Pendant.



Casa segura

Esta pantalla permite definir y configurar una posición segura para el robot. Incluye:

- **Definición de Safe Home**:
  - Botón **"Sync From Home"**: Configura la posición actual del robot como posición segura.
  - Botón **"Delete"**: Elimina la configuración actual de Safe Home.
- **Posiciones de las articulaciones**:
  - Muestra los ángulos actuales de cada articulación.

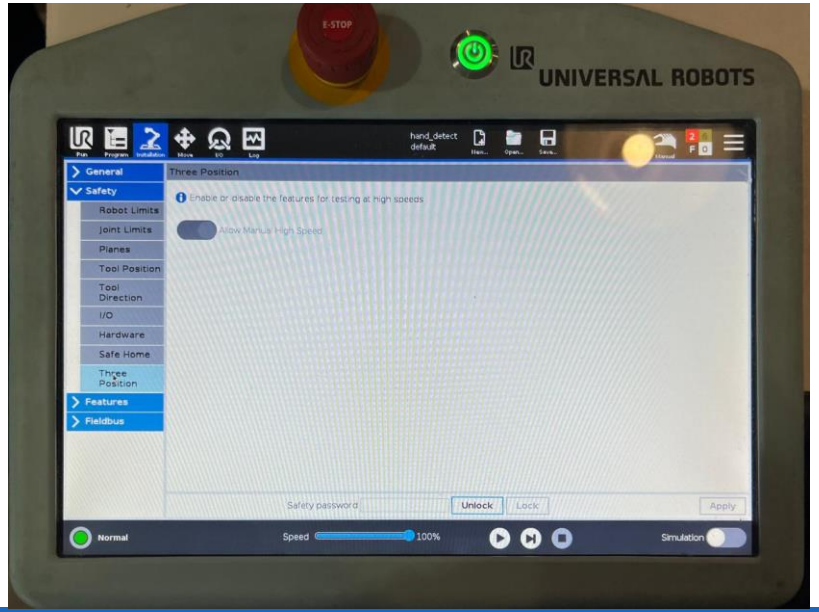
Tres posiciones

Esta pantalla permite habilitar o deshabilitar la opción de operar el robot a alta velocidad de manera manual, lo que es útil para realizar pruebas específicas.

- **Opción principal**:
  - **"Allow Manual High Speed"**: Permite realizar movimientos manuales a alta velocidad cuando está activada.

¿Qué es "Three Position"?

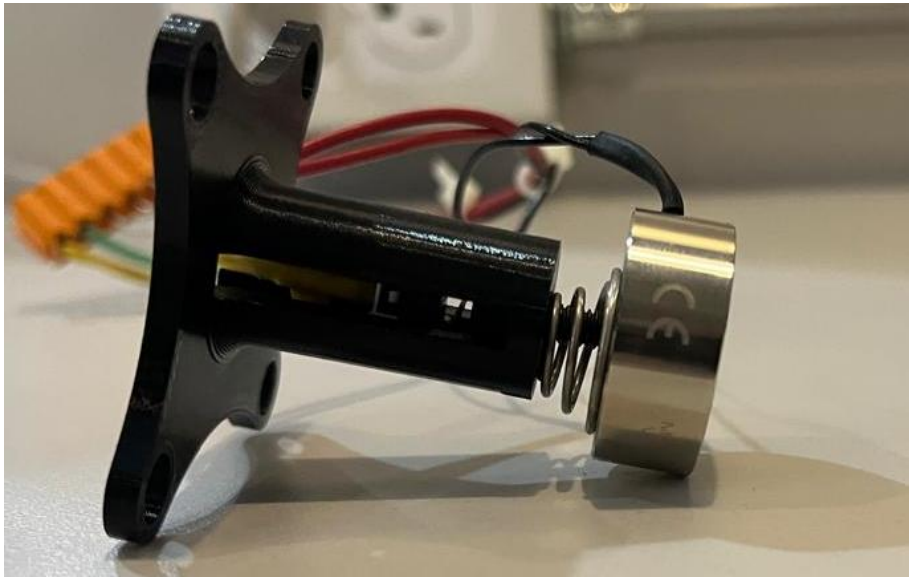
Es una característica de seguridad que controla el uso de altas velocidades manuales durante pruebas, limitando los riesgos en entornos donde se requiere precisión o ajustes manuales rápidos. Esto debe activarse únicamente en condiciones seguras y bajo supervisión adecuada.



Control de interruptor final de carrera

Ubicación, conexión, lectura, script, puntos a mejorar

Ubicación y Funcionamiento
El interruptor de final de carrera está montado dentro de la punta del **TCP (Tool Center Point)** del robot. Su propósito es detectar cuando el electroimán, también ubicado en el TCP, presiona un objeto o superficie. El sensor se activa mecánicamente por la presión generada al hacer contacto, proporcionando una señal digital al sistema de control del robot.



Código para el Control del Sensor
Se utiliza un hilo especial en Python para monitorear el estado del sensor en tiempo real. Este hilo permite una respuesta asincrónica e inmediata cuando el sensor detecta contacto, lo que asegura que el robot pueda detenerse y regresar a su posición segura.

Ubicación, conexión, lectura, script, puntos a mejorar

Función *monitor_io_and_interrupt*

Propósito:

- Monitorea continuamente el estado del interruptor de final de carrera conectado al pin digital 0.
- Detiene el robot de forma segura y lo devuelve a su posición segura (Safe Home) al activarse el sensor.

Parámetros:

- control: Controla movimientos y paradas del robot.
- receive: Lee el estado de los pines digitales.
- io: Maneja las salidas digitales, como el electroimán.
- stop_event: Permite detener el hilo cuando sea necesario.

Funcionamiento:

- Monitoreo: Verifica periódicamente el estado del sensor.
- Secuencia al activarse:
- Detiene el robot con stopL y stopJ.
- Llama a gohome(control) para moverlo al Safe Home.
- Apaga el electroimán tras 5 segundos.
- Espera: Duerme 0.1 segundos entre verificaciones.
- Finalización: El hilo termina cuando stop_event es activado.

```python
def monitor_io_and_interrupt(control, receive, io, stop_event):
    print("Iniciando el hilo de monitoreo del sensor digital.")
    while not stop_event.is_set():
        sensor_state = receive.getDigitalInState(0)  # Leer el estado del pin 0
        if sensor_state:  # Si el sensor se activa
            print("Sensor activado, deteniendo el robot.")
            control.stopL(1.0)
            control.stopJ(1.0)
            gohome(control)
            time.sleep(5)
            io.setStandardDigitalOut(0, False)  # Apagar el electroimán
        time.sleep(0.1)  # Esperar un poco antes de verificar de nuevo
    print("Hilo de monitoreo detenido.")
```

Ubicación, conexión, lectura, script, puntos a mejorar

Puntos a mejorar

Diseño del Eje de Presión:

- Actualmente, el botón del final de carrera es presionado por un tornillo, lo cual no es ideal.
- Problema: La cuerda del tornillo puede atorarse, afectando la precisión y confiabilidad del sensor.
- Mejora Propuesta: Implementar un eje diseñado específicamente para ejercer presión uniforme sobre el botón, evitando atascos y garantizando un funcionamiento más fluido.

Optimización del Cableado:

- Actualmente, el cableado del final de carrera se realiza por fuera del robot.
- Problema: Esto puede interferir con el movimiento del robot, aumentar el desgaste del cable y afectar la estética.
- Mejora Propuesta: Investigar la posibilidad de conectar el sensor directamente al conector ubicado en la punta del UR5e. Esto eliminaría el cableado externo y haría la implementación más robusta y profesional.

Control de electroimán

Ubicación, conexión, script, puntos a mejorar

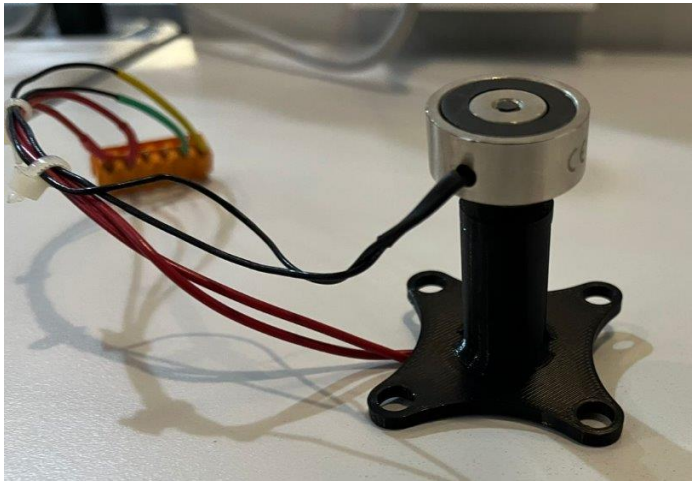
Ubicación y Conexión

1. Ubicación:

1. El electroimán está montado en la punta del TCP (Tool Center Point) del robot, junto con el final de carrera.
2. Su posición permite recoger y manipular objetos metálicos de forma eficiente.

2. Conexión:

1. Alimentado mediante un circuito de potencia de 24VDC.
2. Controlado por un relevador activado por una salida digital estándar del robot, garantizando aislamiento eléctrico y protegiendo el robot contra sobrecarga de corriente.



Ubicación, conexión, script, puntos a mejorar

Script para Control del Electroimán

Encender:

- Se activa al detectar y seleccionar la localización de un objeto válido:
- python
  - io.setStandardDigitalOut(0, True)

Apagar:

- Se desactiva al entregar el objeto en una ubicación predefinida (por ejemplo, sobre la mano del operador):
- python
  - io.setStandardDigitalOut(0, False)

Ubicación, conexión, script, puntos a mejorar

Puntos a Mejorar

Seguridad del Circuito:

- Problema: Actualmente se utiliza un relevador básico. Esto podría no ser suficiente para proteger el sistema en caso de fluctuaciones en la corriente, fallos eléctricos o sobrecalentamiento.
- Mejora Propuesta: Implementar un sistema de protección adicional, como un fusible o un circuito de monitoreo de corriente, para prevenir daños.

Ubicación del Cableado:

- Problema: Los cables del electroimán están expuestos y se dirigen al controlador a través del exterior del robot, lo que puede interferir con los movimientos y aumentar el desgaste.
- Mejora Propuesta: Investigar el uso del conector en la punta del UR5e para integrar el cableado internamente, eliminando cables externos y mejorando la estética y la durabilidad.

Optimización del Encendido/Apagado:

- Problema: El electroimán se apaga inmediatamente después de la entrega del objeto, lo que podría no ser ideal para objetos pesados o situaciones de entrega imprecisa; además existe la posibilidad de un sobrecalentamiento si se queda encendido.
- Mejora Propuesta: Añadir un pequeño retraso programado antes de apagar el electroimán, asegurando una entrega más estable, y buscar asegurar su apagado.

Seguimiento de mano

Script

Este script utiliza un modelo YOLOv8 con **fine-tuning** para localizar la mano en tiempo real mediante visión computacional. Las coordenadas detectadas se envían al robot UR5e para que posicione su TCP (Tool Center Point) directamente sobre la mano.

Características principales

1. Detección precisa:
- El modelo YOLOv8 identifica la mano y calcula sus coordenadas a partir de una imagen capturada por una cámara RealSense.

2. Seguridad en el movimiento:
- Se implementaron límites para evitar que el robot intente alcanzar coordenadas fuera de su rango operativo:
  - **Muy lejanas**: Evita estirar en exceso al robot.
  - **Muy cercanas**: Previene colisiones consigo mismo o movimientos erráticos.

3. Altura predefinida:
- Se asignó una altura fija al TCP porque el script de transformación de coordenadas y densidad de píxeles tiene limitaciones cuando la mano está colocada directamente en el centro de la cámara.

Script

Explicación del Script (Función por Función)

Inicialización y Utilidades

- initialize_pipeline(): Configura y arranca el pipeline de la cámara RealSense para capturar frames de color y profundidad.
- get_color_frame_and_distance(pipeline): Captura frames desde la cámara y calcula la distancia desde el centro de la imagen al objeto.
- convert_to_grayscale(color_image): Convierte una imagen BGR a escala de grises para procesamiento simplificado.
- run_yolo(model, image, conf_threshold=0.6): Ejecuta el modelo YOLO en la imagen y devuelve las detecciones con un umbral de confianza.
- calculate_object_points(results): Extrae las coordenadas del centro de los objetos detectados, su clase y confianza.
- draw_center_points(image, points): Dibuja puntos en la imagen en los centros de los objetos detectados.
- display_image(window_name, image): Muestra una imagen en una ventana.

Script

Coordenadas y Transformaciones

- robot_and_camara(distance, object_points): Calcula las coordenadas de los objetos detectados respecto a la posición del robot.
- mapValue(value, from_low, from_high, to_low, to_high): Realiza una transformación lineal para mapear valores de un rango a otro.
- transformCoordinates(x1, y1, ofx, ofy, theta): Transforma las coordenadas del objeto al marco del robot usando matrices homogéneas.
- frameOriginCoordinates(xtool, ytool, H_cam, V_cam, wrist3): Calcula el origen del frame de referencia de la cámara basado en la posición y orientación del TCP.

Control del Robot

- move_robot(xtransfn, ytransfn, ofzn, control, receive): Mueve el robot a las coordenadas transformadas si están dentro de su alcance.
- gohome(): Lleva el robot a la posición "Safe Home" predefinida.
- monitor_io_and_interrupt(): Monitorea el estado del sensor digital y detiene el robot si se activa.
- safe_move_to_home(): Verifica la conexión del robot y lo mueve a "Safe Home" si es seguro hacerlo.
- seguir_mano(object_points): Mueve el robot hacia los puntos detectados de la clase "Mano".

Script

Ejecución Principal

- main():
  - Configura el pipeline de la cámara y el modelo YOLO.
  - Lanza un hilo para monitorear el sensor digital.
  - Captura frames, procesa detecciones, calcula coordenadas y mueve el robot según los resultados.

Presione para seguir el vínculo
https://github.com/JPHAJP/UR5_SRUB_NURSE/blob/main/V1.0/Scripts/jp_hand.py

03

Resumen y herramientas útiles

Resumen técnico

1. Se enciende el UR5 y se conecta por ethernet.
2. Se ejecuta el programa principal.
3. Con la ayuda de la IA empieza a detectar los instrumentos quirúrgicos y la voz.
4. Se ejecuta una acción de que instrumento se detecto y se toma (con voz, SILVIA, o teclado).
5. Se entrega el instrumento quirúrgico con la ayuda de un electro imán (mediante comando de entrega, con voz o teclado).

Herramientas mas útiles

1. Python
   - Flask: Con la ayuda de esta herramienta pidamos hacer lo que es desarrollo de la interfaz gráfica y correr una web app
   - Tensorflow: Con esta herramienta nos permite crear lo que es la red neuronal
   - Ultralytics: Nos ayuda a correr el modelo ya entrenado y detectar los objetos en tiempo real
2. CUDA
   - Con esto podemos hacer los entrenamientos de las redes neuronales con la GPU
3. OPEN AI
   - Whisper: Nos permite crea una herramienta por comandos de voz
4. Google
   - GTTs: Con ello podemos hacer la transcripción de voz a texto

DATASET

Jupyter notebook para entrenamiento https://www.kaggle.com/datasets/jphajp/ur5e-srube-nurse-surgical-instruments/data

GITHUB

Presione para seguir el vínculo https://github.com/JPHAJP/UR5_SRUB_NURSE

UR_RTDE

Presione para seguir el vínculo https://sdurobotics.gitlab.io/ur_rtde/index.html

04

Referencias

Referencias de antecedentes

- Bai, M., Guo, R., Zhao, Q., & Li, Y. (2021). Artificial intelligence-based CT images in analysis of postoperative recovery of patients undergoing laparoscopic cholecystectomy under balanced anesthesia. Scientific Programming, 2021, 1–7. https://doi.org/10.1155/2021/1125573
- Briganti, G., & Le Moine, O. (2020). Artificial intelligence in medicine: Today and tomorrow. Frontiers in Medicine. https://doi.org/10.3389/fmed.2020.00027
- Cochran, A. (2016). Introduction to the operating room. McGraw Hill Professional.
- Gollapudi, S. (2019). Learn computer vision using OpenCV with deep learning CNNs and RNNs. Springer. https://doi.org/10.1007/978-1-4842-4261-2
- Guillén, I. S., & Brau, A. J. G. (2010). Manual práctico de instrumentación quirúrgica en Enfermería. Elsevier España. https://dialnet.unirioja.es/servlet/libro?codigo=707765
- Hou, Y., et al. (2021). Adaptive kernel selection network with attention constraint for surgical instrument classification. Neural Computing and Applications, 34(2), 1577–1591. https://doi.org/10.1007/s00521-021-06368-x
- Kühn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer. https://doi.org/10.1007/978-1-4614-6849-3
- Li, Y. (2022). Research and application of deep learning in image recognition. 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA). https://doi.org/10.1109/icpeca53709.2022.9718847
- Liu, R., Rong, Y., & Peng, Z. (2020). A review of medical artificial intelligence. Global Health Journal, 4(2), 42–45. https://doi.org/10.1016/j.glohj.2020.04.002

Referencias de antecedentes

- Loaiza Quintana, A. F., Manzano Herrera, D. A., & Múnera Salazar, L. E. (2012). Sistema de visión artificial para conteo de objetos en movimiento. El Hombre y la Máquina.
- Martins, D. (2018). Sorting Surgical Tools from a Cluttered Tray – Object Detection and Occlusion Reasoning. Universidad de Coimbra.
- Muralidhar, D., et al. (2021). Collaborative robot as scrub nurse. Current Directions in Biomedical Engineering, 7(1), 162–165. https://doi.org/10.1515/cdbme-2021-1035
- Nakano, A., Author_Id, N., & Nagamune, K. (2022). A development of robotic scrub nurse system - detection for surgical instruments using faster Region-Based Convolutional Neural Network. Journal of Advanced Computational Intelligence and Intelligent Informatics, 26(1), 74–82. https://doi.org/10.20965/jaciii.2022.p0074
- Nemitz, R. (2019). Instrumental quirúrgico. Editorial El Manual Moderno.
- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2 (3rd ed.). Packt Publishing.
- Sarvamangala, D., & Kulkarni, R. (2020). Convolutional neural networks in medical image understanding: A survey. Springer. https://doi.org/10.1007/s12065-020-00540-3

Referencias técnicas

- González, G. M., Reynoso, S. C., & Alonso, J. P. H. (2024). Development of an Object Recognition-Based Assistance System for Surgical Instrumentation Using an UR5e Robotic Arm. 2024 IEEE International Conference on Engineering Veracruz (ICEV), Boca del Río, Veracruz, Mexico, pp. 1–6. https://doi.org/10.1109/ICEV63254.2024.10765997
- GitHub Repository. (n.d.). UR5_SRUB_NURSE project. https://github.com/JPHAJP/UR5_SRUB_NURSE/tree/main
- Intel Corporation. (2020). Intel RealSense D400 Series Datasheet. https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf
- Kebria, P. M., et al. (2016). Kinematic and Dynamic Modelling of UR5 Manipulator. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). https://doi.org/10.1109/SMC.2016.7844896
- Meta AI. (n.d.). Meta Llama 3.1. https://ai.meta.com/blog/meta-llama-3-1/
- OpenAI Whisper. (n.d.). https://github.com/openai/whisper
- SDU Robotics. (n.d.). UR_RTDE Documentation. https://sdurobotics.gitlab.io/ur_rtde/index.html
- Ultralytics. (n.d.). https://www.ultralytics.com/es
- Universal Robots. (2019). OEM Control Box Installation Guide. https://jorgensen.pl/uploads/offer/Robotics/UR/OEM_Control_Box_Installation_Guide_en_20190815.pdf
- Universal Robots. (2024). UR5e RGB Fact Sheet. https://www.universal-robots.com/media/1807465/ur5e-rgb-fact-sheet-landscape-a4.pdf

05

Agradecimientos

Proyecto de ASE 2 - Inteligencia Artificial - JETSON-NANO

Miembros

- Altamirano Canepa Omar (Ing. Mecatrónica)
- Avendaño Bravo Juan Carlos (Ing. Mecatrónica)
- Castro Reynoso Salvador (Ing. Mecatrónica)
- Hernández Alonso José Pablo (Ing. Mecatrónica)
- Michel González Guillermo (Ing. Biomédica)

Asesores

- Mtro. Huber Girón Nieto
- Ing. Rafael Pérez Aguirre
- Ing. Oliver Ochoa García
- Dra. Nora del Rocío Morúa Álvarez

Participación del proyecto

- ExpoIbero (2° lugar)

Tecnología e instrumentación médica 1:

Miembros

- Castro Reynoso Salvador (Ing. Mecatrónica)
- Michel González Guillermo (Ing. Biomédica)

Asesores

- Mtro. Huber Girón Nieto
- Ing. Rafael Pérez Aguirre
- Ing. Oliver Ochoa García
- Mtra. Laura Duran Fernández

Colaboradores

- José Pablo Hernández Alonso (Ing. Mecatrónica)

Participación del proyecto

- ExpoIbero

Tecnología e instrumentación médica 2:

Miembros

- José Pablo Hernández Alonso (Ing. Mecatrónica)
- Aldo Cova Martínez (Ing. Sistemas Computacionales)

Asesores

- Mtro. Huber Girón Nieto
- Ing. Rafael Pérez Aguirre
- Ing. Oliver Ochoa García
- Mtra. Laura Duran Fernández

Colaboradores

- Castro Reynoso Salvador (Ing. Mecatrónica)
- Michel González Guillermo (Ing. Biomédica)
- Javier Atlixqueño Bárcenas (Ing. Biomédica)
- Alexia Hernández Arciniega (Ing. Biomédica)
- Juan Carlos Jiménez Molina (Ing. Biomédica)
- Noe Aguirre Gallegos (Ing. Mecatrónica)

Participación del proyecto

- ExpoIbero
- CONITACS
- IEEE (Robotics)