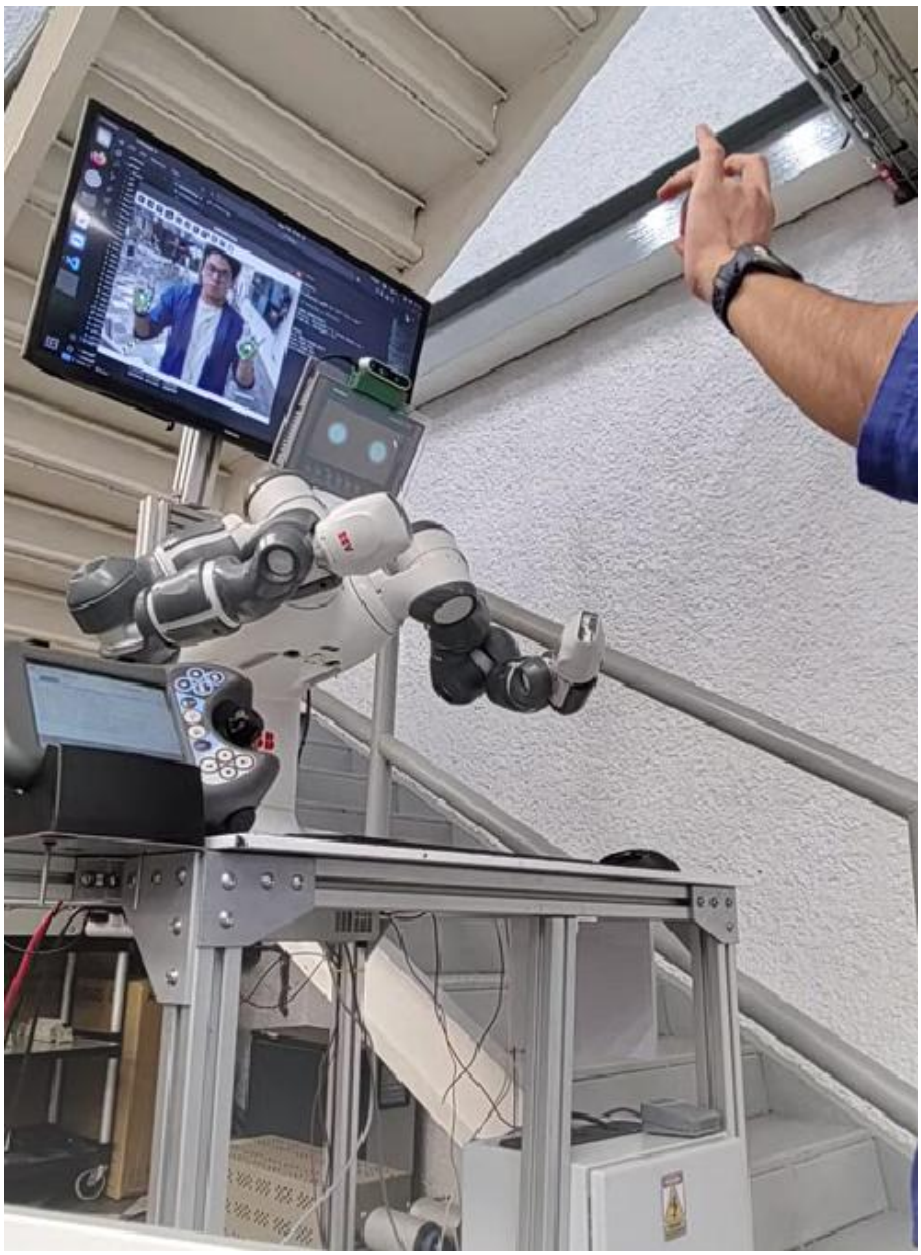
REPORTE DEL PROYECTO DE CONTROL DEL ROBOT ABB YuMi CON RECONOCIMIENTO DE MANOS



Autores:

Ing. *Carlos Martínez García*

Ing. estudiante *GROUX Kyllians*

RESUMEN :

1. Introducción

Este sistema permite controlar de forma **intuitiva** los brazos del robot ABB YuMi IRB 14000 usando el seguimiento en tiempo real de las manos del operador con un ligero retraso debido al tiempo de procesamiento, capturadas por una cámara **Intel RealSense** y procesadas con **Python**, **MediaPipe** y con una **Jetson Nano** como computadora para el procesamiento de las imágenes.

El sistema envía al robot:

- **Coordenadas 3D** de cada mano para mover los TCPs de cada brazo.
- Señales "ABIERTO" y "CERRADO" para controlar la apertura y cierre de los **Smart Grippers**, usando el gesto de cerrar o abrir la mano.

Cuando no se detecta la mano durante **10 segundos**, el sistema devuelve automáticamente el brazo a una **posición HOME segura** y abre el gripper.

2. Componentes del sistema

Elemento	Descripción
Robot	ABB YuMi IRB 14000 con controlador IRC5.
Cámara	Intel RealSense D435i (o similar) para captura de color y profundidad.
Host visión	NVIDIA Jetson Nano con Ubuntu Linux.
Software visión	Python 3, MediaPipe, OpenCV, PyRealSense2.
Red	Conexión TCP/IP Jetson ↔ Controlador IRC5.
Programas RAPID	Módulos MainModule_Left.mod y MainModule_Right.mod.

3. Requisitos previos

Jetson Nano con:

- Python 3
- Dependencias: pip install opencv-python mediapipe pyrealsense2

Conexión de red Jetson ↔ IRC5 via cable Ethernet:

- Configura IP fija, por ejemplo:
    - Jetson: 192.168.125.2
    - IRC5: 192.168.125.1

Archivos RAPID cargados en cada brazo del YuMi:

- Brazo izquierdo escucha en puerto 30010
- Brazo derecho escucha en puerto 30011

FlexPendant calibrado (g_Calibrate para Smart Grippers).

4. Cómo usarlo

1) Inicia el controlador RAPID con MainModule_Left y MainModule_Right.
2) Verifica que los sockets acepten conexión (TPWrite "Esperando conexión...").
3) Corre la versión del script de Python más reciente guardado en la JetsonNano. Última versión funcional guardada en la Jetson (pero no respaldado en Teams): rs2YuMiv2.8.8.py
4) Oculte las manos hasta que el código de Python despliegue la imagen en tiempo real de la cámara en el monitor encima del robot.
5) Mueve tus manos frente a la cámara RealSense.
6) Observa cómo el YuMi sigue la posición de tus manos y abre/cierra el gripper.
7) Si sacas la mano por más de 15 s, los brazos deberán quedarse en la última posición registrada por el sistema de visión.

5. Estructura de archivos

- **rs2YuMiv2.8.x .py**
  Scripts Python principales, ejecutan la detección, transforman coordenadas y envían datos por socket. La última versión es la 2.8.8, en fecha de Febrero 2026.
- **posImitatorL.mod**
  Programa RAPID para el brazo izquierdo. Acepta comandos de posición y gripper.
- **posImitatorR.mod**
  Programa RAPID para el brazo izquierdo. Acepta comandos de posición y gripper.

6. Autores y versión

Gracias de descargar todo el proyecto, con las ultimas y previas versiones, en este enlace GitHub : **https://github.com/Kyll-GRX/ABB-YuMi-robot---SmartFactory-LabTecnologico-de-Monterrey**

Gracias de encontrar videos de presentación u demonstración del funcionamiento del robot con este enlace Google Drive, en el folder "*Documentation*" : https://poce-my.sharepoint.com/:f:/g/personal/kyllians_groux_edu_ece_fr/IgCfNc5_wvgQTpkpReF_z66bAdX41Q7YEMKHBMT2IjWQrGM?e=SE7wbk

*gracias de **pedirme el permiso** antes de usar mis videos! kyllians.groux@edu.ece.fr

AUTORES :

Diciembre-Enero 2026: **el Ing. estudiante *Kyllians GROUX***

➔ Versiones v2.8.1 hasta v2.8.8

Datos de Contacto:

- E-mail: kyllians.groux@edu.ece.fr (por favor incluir asunto)
- Teléfono Celular: +33 670705670 (WhatsApp únicamente)
- Estudiante en la escuela ECE Paris, Electronics Engineering School

Enero-Mayo 2025: **el Ing. *Carlos Martínez García***

➔ Versiones v1.0 hasta v2.8.0

Datos de Contacto:

- E-mail: mgcar_los@hotmail.com (por favor incluir asunto)
- Teléfono Celular: +52 4641767027 (WhatsApp preferencialmente)

7. Arquitectura y Funcionamiento del Sistema (based on v2.8.8)

El sistema opera bajo un esquema de **teleoperación en tiempo real**, donde una unidad de procesamiento (NVIDIA Jetson) actúa como middleware entre un sensor de visión profunda y un controlador industrial ABB IRB 14000.

7.1 Flujo de Operación

El ciclo de control se divide en cuatro etapas fundamentales que ocurren en cada iteración del bucle principal:

1. **Percepción visual del usuario:** La cámara **Intel RealSense RGB-D** captura simultáneamente el flujo de video y el mapa de profundidad. Gracias à la librería **MediaPipe Hands**, se extraen los 21 *landmarks* (puntos de referencia) de cada mano, priorizando las coordenadas de la muñeca y las puntas de los dedos.
2. **Procesamiento y Normalización:** El script en Python recibe coordenadas normalizadas [0, 1] del espacio de la imagen (espacio de la cámara). Estas se transforman al espacio cartesiano del robot en milímetros mediante un **mapeo de rangos restringidos**.
3. **Transmisión de Datos (Middleware):** Las coordenadas y estados del gripper se empaquetan en un protocolo de comunicación robusto y se envían vía **Sockets TCP/IP**.
4. **Ejecución Cinemática:** El controlador ABB recibe el mensaje (la trama de datos), la desempaqueta (parsing) y ejecuta el movimiento mediante instrucciones MoveJ, asegurando que el TCP (*Tool Center Point*) alcance el objetivo.

7.2 Transformación de Coordenadas y Seguridad Industrial

7.2.1 Mapeo de Rangos y Escalamiento:

Como pasar de la escala en pixeles de la cámara a la escala en milímetros del robot ?

El sistema v2.8.8 implementa una **arquitectura de teleoperación basada en el mapeo de rangos restringidos**. A diferencia de versiones anteriores, el control no es relativo al movimiento, sino absoluto respecto a un volumen de trabajo predefinido.

Mediante la función map_range, las coordenadas normalizadas de la cámara se interpolan linealmente dentro de los límites físicos seguros del YuMi. Esto permite que el sistema actúe como un **filtro de seguridad activo**: aunque el usuario realice movimientos fuera de rango, el algoritmo los restringe (clamping) a las zonas operativas ROB_MIN/MAX, evitando colisiones bimanuales y exceder los límites articulares. Finalmente, la fijación del eje Z a una altura constante de 450mm garantiza la estabilidad operativa al ignorar el ruido sensorial de profundidad.

A diferencia de versiones anteriores con escalamiento lineal simple, la versión v2.8.8 implementa **Interpolación Lineal** para proyectar el área de visión en un volumen de trabajo seguro.

- **Filtro de seguridad activo (clamping):** El sistema actúa como un filtro. Si el usuario realiza un movimiento fuera del campo de visión, el algoritmo "recorta" el valor a los límites máximos permitidos (ROB_MIN/MAX) en el sistema de coordenadas del robot, evitando colisiones mecánicas o errores de configuración de ejes.

- **Formula de mapeo:**

$$Pos_{rob} = ROB_{min} + \frac{(Pos_{cam} - CAM_{min}) \cdot (ROB_{max} - ROB_{min})}{CAM_{max} - CAM_{min}}$$

7.2.2 Reducción de Grados de Libertad (2D vs 3D):

Aunque el sistema utiliza un sensor RGB-D, existe una diferencia técnica fundamental entre cómo se capturan los ejes:

1. **Naturaleza del sensor:** Mientras que los ejes **X** e **Y** se obtienen directamente de la matriz de pixeles de la imagen (datos en 2D de alta precisión), el eje **Z (profundidad)** es una **estimación calculada** basada en la disparidad de la visión estéreo o el tiempo de vuelo de la luz infrarroja.

2. **El problema del ruido (jitter):** Debido a que la mano humana no es una superficie plana y presenta constantes cambios de oclusión y textura, el cálculo de la profundidad tiende a generar **ruido sensorial extremo**.
   En un entorno de control, esto se traduce en valores que fluctúan rápidamente (jitter), incluso si el usuario mantiene la mano fixa.

3. **Riesgo de inestabilidad cinemática:** Si enviáramos estos valores iunestables al YuMi, el robot intentaría corregir su posición milímetros hacia adelante y atrás constantemente a alta velocidad. Esto provocaría:
   - **Movimientos brutales:** Cambios de posición repentinos que podrían dañar los servomotores.
   - **Riesgo de Colisión:** Un salto falso en el cálculo de profundidad podría clavar el gripper contra la mesa o el entorno.

Decisión Técnica de Seguridad: Para garantizar una operación **planimétrica estable**, se optó por ignorar la lectura dinámica de profundidad y **fijar el eje Z** del robot a una altura constante de **450 mm**. Esta simplificación técnica elimina el riesgo de colisión por ruido sensorial, permitiendo que el operador se concentre en la precisión del movimiento en el plano de trabajo (Y-Z del robot) sin preocuparse por desplazamientos accidentales en el eje de profundidad.

7.3 Control del Efector Final (Gripper)

El estado del gripper se determina gracias al cálculo de la **distancia euclidiana entre el pulgar y el dedo medio**.

- **Lógica Binaria:** Si la distancia es menor a un umbral definido (GRIPPER_THRESH), se envía la señal 0 (CERRADO); de lo contrario, se envía 1 (ABIERTO).

- **Integración de Datos:** En la v2.8.8, el estado del gripper se integra al final de la cadena de coordenadas (ej. 450,-150,350,1/), permitiendo que el robot reciba la posición y el estado de la herramienta en un solo paquete de datos, reduciendo el tráfico de red.

7.4 Gestión de Errores y Robustez de Comunicación

7.4.1 Protocolo de Comunicación TCP/IP

Para mandar las coordenadas, se usa el protocolo TCP/IP entre la Jetson y el controlador ABB (de IP: 192.168.125.1). El protocolo TCP/IP es una forma estándar de comunicar dos computadoras por red Ethernet, con comunicación fiable. Es un conjunto de los protocolos:

IP: direcciona los dispositivos (IP: 192.168.125.1 etc...)

TCP: abre una conexión confiable entre dos equipos y envía datos en orden.

- **Protocolo Robusto:** El uso de Sockets TCP garantiza que los paquetes lleguen en orden y sin pérdidas, algo vital para la integridad del robot.
- **Prevención de TimeOuts:** Si la mano del operador sale del campo de visión por más de **15 segundos**, el script activa un protocolo de **"Keep Alive"**. Envía periódicamente la señal NO-DATA o la última posición registrada para evitar que el controlador ABB cierre el puerto por inactividad.

7.4.2 Reasignación de Ejes (Mapping Espacial)

MediaPipe te da las coordenadas (X, Y, Z) de la imagen. Pero el robot interpreta coordenadas en su propio sistema... Para el diseño del *ABB YuMi*, el plano de trabajo donde se mueven los brazos corresponde a los ejes Y–Z del robot.

Debido a la orientación física de la cámara frente al YuMi, existe una discrepancia en los marcos de referencia. El sistema realiza una reasignación automática:

Cámara (X, Y, Z) ⇔ Robot (Y, Z, X)

Esto permite que el usuario, al mover la mano de derecha a izquierda (eje X en cámara), vea al robot moverse en su eje Y, logrando un efecto espejo natural e intuitivo.

7.5 Definiciones Técnicas Clave

- **TCP (Tool Center Point):** Es el punto focal de la herramienta del robot. Todas las coordenadas enviadas por el script Python le indican al controlador ABB exactamente dónde posicionar este punto en el espacio tridimensional.
- **Offset:** Es la compensación numérica aplicada para alinear el "cero" de la cámara con la posición **HOME** de seguridad del robot.
- **RAPID Logic:** El script dentro del controlador ABB se encarga de convertir las cadenas de texto (Strings) recibidas por el socket en valores numéricos procesables, aplicando filtros finales de rango mediante la función LimitTCP antes de mover los motores.

8. Explicaciones sobre el lenguaje de programación RAPID:

Función: ***SocketPeek(stringVar)***
Checa si hay bytes esperando en la parte de recepción de datos y cuentos hay...
Regresa:

0 ? no datos disponibles
>0 ? números de bytes listos para estar leído
<0 ? error de programa

Función: ***MoverJ(positionVar[], Velocity, Zone, toolName)***
ejecuta un movimiento NO rectilíneo del COMPONENTE «*toolName*»
desde su posición ACTUAL hasta la posición de coordenadas en «*positionVAR*[]»
con una VELOCIDAD específica, en una ZONA/ÁREA específica.

Proceso de Conversión de las Coordenadas:
El sistema externo (cámara + Jetson) envía coordenadas en forma de texto, ya que es sencillo y robusto a través de la conexión TCP/IP.
Así, el robot recibe una cadena y la convierte en coordenadas enteras utilizando las siguientes funciones:

→ **Función**: ***StrPart(variableName, start, length)***
Extrae una subcadena de una cadena... por lo que puede extraer una parte de la cadena «*variableName*», desde *START* hasta *START+LENGHT* incluido.

→ **Función**: ***StrToVal(strVar, numVar)***
Convierte una cadena numérica en un número (devuelve *TRUE* si la conversión se realiza correctamente, *FALSE* si falla).

Más detalles sobre el eje externo: son datos para robots con:

- guías lineales
- posicionadores
- ejes rotativos adicionales

9E+09 significa: ignorar este eje / no se utiliza ningún eje externo

Qué es *ZONE* en la función *MoveJ()* ?

ZONE = precisión con la que el robot debe atravesar el punto de destino.
fine → el robot debe detenerse exactamente en el punto de destino
(sin redondeos, más lento, más seguro)

- z1 → permite un redondeo de 1 mm
- z10 → 10 mm
- z50 → 50 mm
- z100 → 100 mm

Una zona más alta = movimiento más suave y rápido, pero menos preciso

CoordenadasVar ***[[X,Y,Z], [Q1,Q2,Q3,Q4], [CF1,CF2,CF3,CF4], [E1,E2,E3,E4,E5,E6]]***

[X,Y,Z] <-- posición del TCP (punto central de la herramienta – Tool Central Point) en coordenadas cartesianas

[Q1,Q2,Q3,Q4] <-- orientación (o cuaternión) que representa la rotación de la herramienta en 3D

[CF1,CF2,CF3,CF4] <-- datos de configuración, contiene información sobre la postura del brazo del robot

[E1,E2,E3,E4,E5,E6] <-- ejes externos (9E+09 = ignorar este eje)

Más detalles sobre la parte de datos de configuración: contiene información sobre la postura del brazo robótico:

- CF1: Configuración del brazo (codo arriba/abajo, hombro izquierdo/derecho)
- CF2: Configuración de la muñeca
- CF3: Número de giros (revoluciones de los ejes)
- CF4: configuración cinemática externa

Los robots ABB necesitan esto para elegir una de las varias posturas posibles del robot que alcanzan el mismo punto.

NEW PAGE HERE: