Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO:

Control del brazo robot UR3 de Universal Robot mediante imitación de movimiento

AUTOR: MÉRIDA RUBIO, ORIOL

APELLIDOS: Mérida Rubio **NOMBRE: Oriol**

TITULACIÓN: Grado en Ingeniería Electrónica Industrial y Automática

PLAN: 2009

DIRECTOR: De Sousa Perez, Oscar

DEPARTAMENTO: ESAII - Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial

CALIFICACIÓN DEL TFG

TRIBUNAL

PRESIDENT	SECRETARI	VOCAL
Ramon Guzman Sola	Pau Marti Colom	Agustin Fortuny Sanroma

FECHA DE PRESENTACIÓN: octubre 2023

Este Proyecto tiene en cuenta aspectos medioambientales: ☐ Sí ☒ No

RESUM

En aquest projecte es proposa realitzar el control dels moviments d'un braç robòtic a partir dels moviments d'un braç humà. En aquest cas es tracta de, un braç robot d'Universal Robot, més concretament, el model UR3e.

Per a captar i detectar els moviments del braç humà, sé usaran dispositius IMU (sensors de mesurament inercial) per a captar els moviments del dispositiu i transformar aquesta informació per a enviar-li les ordres de moviment al braç robot.

En aquest treball es desenvoluparà un sistema compost, prioritàriament, per sensors inercials on, primer de tot, es descriurà i seleccionar totes les eines i dispositius necessaris per a realitzar aquest projecte (Visual Studio Code, URSim, UR3e, BNO055, ATmega4809, NINA-W102).

D'altra banda, s'exposaran les diferents metodologies que s'han fet servir per a intentar captar els moviments del braç humà, a més de la traducció d'aquesta informació captada, perquè l'UR3e sigui capaç d'interpretar aquesta informació en moviment.

Amb aquest punt de partida, s'apliquen eines específiques del grau en Enginyeria Electrònica Industrial i Automàtica, per a així poder elaborar una proposta vàlida conceptual que satisfaci els objectius fixats en el projecte.

Paraules clau (màxim 10):

UR3 e	IMU	BNO055	Universal Robot
Braç humà	Robot	Comunicació	Pitch
Roll	Heading		

RESUMEN

En este proyecto se propone realizar el control de los movimientos de un brazo robótico a partir de los movimientos de un brazo humano. En este caso se trata de, un brazo robot de Universal Robot, más concretamente, el modelo UR3e.

Para captar y detectar los movimientos del brazo humano, se usarán dispositivos IMU (sensores de medición inercial) para captar los movimientos del dispositivo y transformar dicha información para enviarle las órdenes de movimiento al brazo robot.

En este trabajo se va a desarrollar un sistema compuesto, prioritariamente, por sensores inerciales donde, primero de todo, se va a describir y seleccionar todas las herramientas y dispositivos necesarios para realizar dicho proyecto (Visual Studio Code, URSim, UR3 e, BNO055, ATmega4809, NINA-W102).

Por otro lado, se expondrán las diferentes metodologías que se han usado para intentar captar los movimientos del brazo humano, además de la traducción de esa información captada, para que el UR3 e sea capaz de interpretar esa información en movimiento.

Con este punto de partida, se aplican herramientas específicas del grado en Ingeniería Electrónica Industrial y Automática, para así poder elaborar una propuesta válida conceptual que satisfaga los objetivos fijados en el proyecto.

Palabras clave (máximo 10):

UR3 e	IMU	BNO055	Universal Robot
Brazo Humano	Robot	Comunicación	Pitch
Roll	Heading		

ABSTRACT

This project aims to control the movements of a robotic arm from the movements of a human arm. In this case, it is a robot arm of Universal Robot, more specifically, the UR3e model.

To capture and detect movements of the human arm, will use IMU devices (inertial measuring sensors) to capture movements of the device and transform this information to send the movement commands to the robot arm.

In this work, a system composed, as a priority, of inertial sensors will be developed where, first of all, all the tools and devices needed to perform this project (Visual Studio Code, URSim, UR3 e, BNO055, ATmega4809, NINA-W102) will be described and selected.

On the other hand, the different methodologies that have been used to attempt to capture the movements of the human arm, in addition to the translation of this captured information, will be exposed so that the UR3 e is able to interpret this moving information.

With this starting point, specific bachelor's degree tools are applied in Industrial and Automatic Electronics Engineering, so that we can draw up a valid conceptual proposal that meets the objectives set out in the project.

Keywords (10 maximum):

UR3 e	IMU	BNO055	Universal Robot
Human arm	Robot	Communication	Pitch
Roll	Heading		

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Este proyecto está centrado en la realización de un sistema capaz de leer las posiciones reales de un brazo humano en todo momento y que, acto seguido, un brazo robot (UR3 e) sea capaz de reproducir dichas posiciones. El objetivo principal de esta investigación es conseguir que el brazo robot sea capaz de imitar el brazo humano para poder llegar realizar tareas a distancia sin la necesidad de estar en la misma sala que el robot. Más concretamente:

- Estudiar antecedentes y referentes que encontramos hoy en día
- Creación de una comunicación inalámbrica entre el dispositivo del brazo humano y el robot.
- Diseño de un dispositivo para la lectura de movimiento del brazo humano y transmisión de datos al brazo robot.
- Estudio de la sincronización entre los datos que genera el sensor respecto los interpretados por el robot.

Este proyecto consiste en la comunicación inalámbrica entre el dispositivo del brazo humano y el robot, pasando por la investigación de los IMU (sensores inerciales) así como de los robots, hasta el diseño del dispositivo capaz de leer la posición del brazo humano para su construcción final.

Desde bien pequeño me ha fascinado el mundo de la robótica. A medida que he ido creciendo, académicamente hablando, me ha despertado más interés este campo, es por eso que decidí encaminar mis estudios hacia este mundo y así poder aportar mi conocimiento y motivación a este sector todavía en expansión.
Los robots fueron creados para generar tareas que los humanos no podemos hacer, ya sea por la repetición, por el peso a mover, por la precisión que alcanzan o por el entorno en el que se tiene que trabajar. Porque, ya sea en el núcleo de una central nuclear, en un quirófano donde necesites precisión de micras, en el fondo del mar o incluso en el espacio, nuestros compañeros robots si pueden estar sin desfallecer. Por estos motivos este proyecto me ha provocado tanta motivación y empeño para llevarlo a cabo.

Metodología

En este proyecto hará falta aplicar una metodología relacionada con el estudio y programación de microcontroladores, en específico, aplicada a los robots colaborativos, que permita entender y actuar en base a las especificaciones del dispositivo. Se utilizará una serie de softwares de programación de Arduino, en este caso, Visual Studio Code (extensión PlatformIO) y Arduino IDE. Para la simulación del robot se utilizará VirtualBox y URSim. Por otro lado, se trabajará con los dos métodos de programación que ofrece el robot, PolyScope y URScript. A parte, se hizo uso de la maquinaria y material disponible en Serveis Tècnics de Laboratori de la universidad como soporte en la investigación y desarrollo del proyecto.

Estructura

Este proyecto está dividido por capítulos, con los que se pretende desarrollar el proyecto. En primer lugar, se encuentra el marco teórico, donde se detallan los diferentes robots que hay, así como su clasificación según distintos criterios. Además, de define que es y cómo funcionan los IMU (sensores inerciales), así como las distintas comunicaciones industriales para su implementación en la comunicación entre los dispositivos tratados en el proyecto.

A continuación, se especifican las características del robot elegido a estudio, así como de la empresa creadora de este.

Además, se proponen distintos dispositivos existentes en el mercado para hacer su posterior elección según sus características y compatibilidad, teniendo en cuenta el objetivo inicialmente establecido.

Seguidamente, se describen las herramientas de programación utilizadas, así como, los dos grandes planteamientos estudiados para el desarrollo óptimo del trabajo.

Por otro lado, se encuentra el presupuesto necesario para llevar a cabo el proyecto, desde la compra de los materiales necesarios hasta las horas que se han invertido tanto en el montaje del mecanismo como en el proceso de ingeniería.

Por último, están los anexos, donde se puede encontrar los diferentes códigos, pasos que se han seguido, todos los *datasheets*, documentación de las librerías usadas y los comandos del robot.

1. ASPECTOS GENERALES

1.1. Definición de robot

Si buscas la definición de la robótica encontrarás varias definiciones diferentes, pero una de las primeras que te aparecerán será: "Técnica que se utiliza en el diseño y la construcción de robots y aparatos que realizan operaciones o trabajos, generalmente en instalaciones industriales y en sustitución de la mano de obra humana."

Pero en mi opinión es un campo que está en constante evolución, ya que es una disciplina que combina diferentes campos, como la informática, la ingeniería, la mecánica, la electrónica y cada vez más la inteligencia artificial. El objetivo primordial es desarrollar máquinas capaces de interactuar con el entorno y realizar tareas de manera autónoma (normalmente repetitivas) o colaborativas con las personas.

La robótica cada vez más está teniendo un impacto mayor en nuestras vidas, desde la industria como la automoción, farmacéutica y la agricultura, hasta la exploración espacial y la robótica de servicios. Donde se están explorando e implementando nuevas tecnologías y aplicaciones como la robótica colaborativa, los sistemas autónomos, la conectividad en la nube y la implementación de inteligencia artificial para poder hacer las cosas mejor, más rápidos, pudiendo sobre todo hacernos la vida más fácil.

1.2. Clasificación de los robots

1.2.1. Tipos de robots por cronología

En este caso, se distinguen hasta cinco tipos de robots, según las etapas por las que ha ido pasando la robótica hasta el momento actual [1]. A continuación, se muestra una tabla donde se observa de forma esquemática esta clasificación:

TIPOS DE ROBOTS POR CRONOLOGÍA	
1ª GENERACIÓN	Robots manipuladores
2ª GENERACIÓN	Robots de aprendizaje
3ª GENERACIÓN	Robots reprogramables
4ª GENERACIÓN	Robots móviles
5ª GENERACIÓN	Robots con inteligencia artificial

Tabla 1. Clasificación de los robots por cronología. Fuente: elaboración propia.

Seguidamente, se procede a explicar con más detalle este tipo de clasificación:

- **Primera Generación: robots manipuladores**

Aquí pertenecen los primeros robots, los desarrollados en los inicios de la segunda mitad del siglo XX. En muchas bibliografías son catalogados como "robots manipuladores". Su sistema es bastante sencillo, realiza tareas repetitivas previamente programadas que se ejecutan de forma manual o secuencial fija. Estas tareas suelen ser de recoger y colocar elementos en cierto lugar, con grandes limitaciones en sus movimientos. Como no tienen conocimiento alguno del entorno, se clasifican como sistemas de lazo abierto, pues no poseen esa retroalimentación para verificar los resultados o errores. Estos tipos de robots aún existen y son utilizados en muchas industrias.

Ilustración 1. Robot manipulador. Fuente: blog automatismosmundo.

- **Segunda Generación: robots de aprendizaje**

Surgen en los años 80. Conocidos como los primeros robots de aprendizaje, siendo capaces de memorizar secuencias de movimientos previamente realizadas por un operador humano. Se caracterizan por tener la capacidad de desplazarse (posiciones fijas) y poder percibir parte de su entorno, mediante sensores especializados. Debido a esto, se convirtieron en sistemas de lazo cerrado, con sistema de retroalimentación básico, pudiendo verificar parte de sus resultados. En la actualidad son utilizados también, fundamentalmente en la industria de la automoción, realizando tareas de pintado, traslado y ensamblaje de piezas en las cadenas de montajes.

Ilustración 2. Robot de aprendizaje. Fuente: blog automatismosmundo.

- **Tercera generación: robots reprogramables**

Fueron denominados como la generación de robots de control sensorizados. La característica fundamental que los hizo superiores fue la posibilidad de que pueden ser reprogramados a través de ordenadores, permitiéndoles ver (visión artificial) y tocar empleando sensores y lenguajes de programación. Esta generación tiene mayor conocimiento del entorno

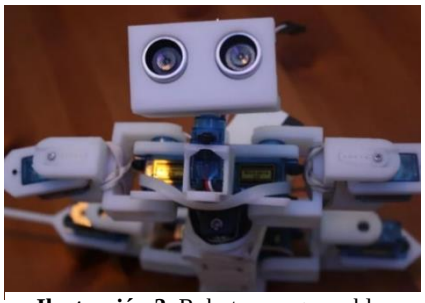


Ilustración 3. Robot reprogramable. Fuente: blog automatismosmundo.

permitiéndole modificar su estrategia de trabajo o desplazamiento, por lo que sistema de control es de lazo cerrado. Algunos estudiosos plantean que aquí comenzó la era de los robots inteligentes.

- **Cuarta Generación: robots móviles**

La generación de los robots inteligentes, similares a los de la cuarta generación, pero con la posibilidad de tomar decisiones en tiempo real y libertad de movimiento o para completar sus tareas. El procesamiento de la información recolectada por sensores, cada vez más precisos, era muy superior a generaciones anteriores. Su desarrollo se determinó, en su mayoría, por la utilización de modelos de análisis con lógicas difusas y redes neuronales, que fueron claves en esa época, finales del siglo XX e inicios del XXI, donde el crecimiento de la informática y electrónica fue exponencial. Se utilizaron en muchísimas actividades de todos los sectores posibles, desde aplicaciones industriales hasta de servicios y entretenimiento. En la actualidad, muchos comienzan a verse con aspectos humanoides.

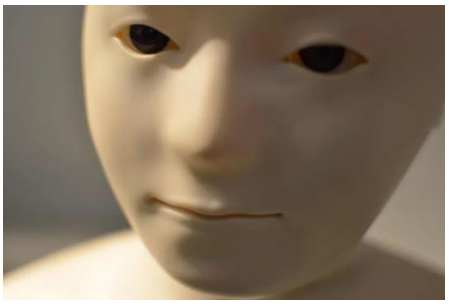


Ilustración 4. Robot móvil. Fuente: blog automatismosmundo.

- **Quinta Generación: robots con inteligencia artificial**

Hasta la fecha, es la última de las generaciones de la robótica. La característica fundamental está en el uso de inteligencia artificial avanzada con aprendizaje automático, sin intervención humana, siendo capaces de imitar y desarrollar modelos de actuación, razonamiento y de conducta, en los entornos más dinámicos y desconocidos. Aquí se rompe con la falsa limitación de que los robots solo pueden ser utilizados en tareas repetitivas. Las aplicaciones de este tipo de robots no tienen límites, su gran capacidad de adaptación les permite ser multifacéticos. Es una generación que aún está en desarrollo y que nos corresponde a las nuevas generaciones continuar con su perfeccionamiento.

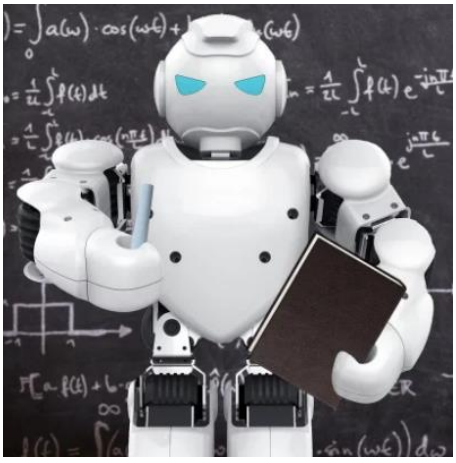


Ilustración 5. Robot con inteligencia artificial. Fuente: blog automatismosmundo.

1.2.2. Tipos de robots según movilidad

Otra clasificación muy interesante de los robots tiene mucho que ver con su movilidad [1]. A continuación, se muestra una tabla donde se observa de forma esquemática la clasificación según su desempeño, su capacidad de movimiento y su toma de decisiones:

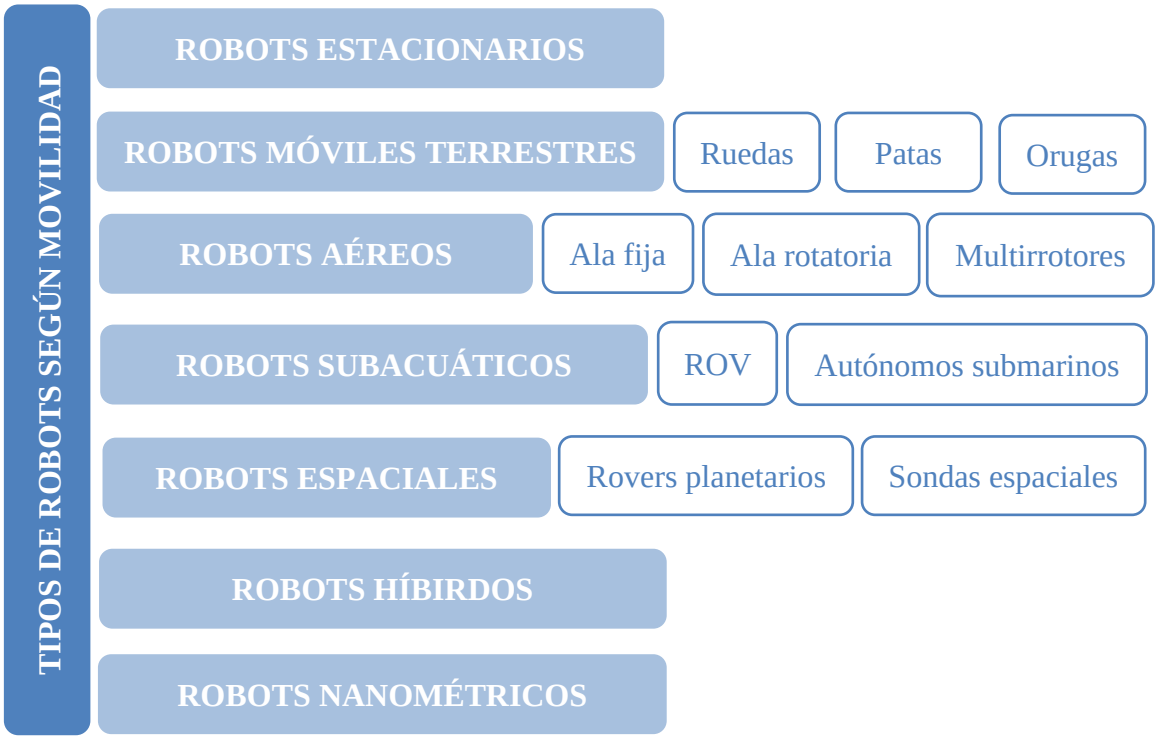


Tabla 2. Clasificación de los robots según movilidad. Fuente: elaboración propia.

Seguidamente, se procede a explicar con más detalle este tipo de clasificación:

- **Robots Estacionarios:** Estáticos y fijos en un lugar determinado. Realizan tareas específicas en su ubicación.
- **Robots Móviles Terrestres:**
  - *Con Ruedas*: Utilizan ruedas para desplazarse en superficies planas.
  - *Con Patas*: Pueden ser bipedales (2 patas) o cuadrúpedos (4 patas).
  - *Con Orugas*: Utilizan orugas, ideales para terrenos irregulares.
- **Robots Aéreos:**
  - *Drones de Ala Fija*: Diseñados como aviones, vuelan de manera más eficiente en largas distancias.
  - *Drones de Ala Rotatoria*: Utilizan rotores para el vuelo vertical y la maniobrabilidad.
  - *Drones de Multirrotores*: Tienen más de cuatro rotores para mayor estabilidad y agilidad.

- **Robots Subacuáticos:**
  - *Vehículos Operados Remotamente (ROV)*: Controlados por operadores humanos para exploración submarina.
  - *Autónomos Submarinos*: Capaces de operar de forma autónoma en entornos acuáticos.
- **Robots Espaciales:**
  - *Rovers Planetarios*: Diseñados para explorar la superficie de planetas y lunas.
  - *Sondas Espaciales*: Se utilizan para explorar cuerpos celestes más allá de la Tierra.
- **Robots Nanométricos:** A nivel micro y nanométrico, estos robots pueden moverse en escalas extremadamente pequeñas.

Ilustración 8. Robot estacionario industrial. Fuente: descubrearduino.com

Ilustración 7. Robot aéreo. Fuente: nexciencia

Ilustración 6. Robot móvil con ruedas. Fuente: multirobotica.

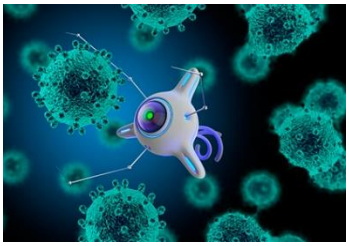


Ilustración 11. Nanorobot. Fuente: eexcellence.

Ilustración 10. Pez robot para vigilar la salud de los océanos. Fuente: The New York Times

Ilustración 9. Robot espacial. Fuente: pocket-lint.

1.2.3. Tipos de robots por función o sector

La robótica abarca muchos tipos de robots, que se podrían clasificar en dos grandes grupos según su aplicación: robots industriales y robots de servicio [1]. A continuación, se muestra una tabla donde se observa de forma esquemática esta clasificación:

TIPOS DE ROBOTS POR FUNCIÓN O SECTOR			
ROBOTS INDUSTRIALES		**ROBOTS DE SERVICIO**	
Cartesianos	Cilíndricos	Humanoides	Zoomórficos
SCARA	Delta	Vehículos autónomos	Consumo
Polares	Articulados	Drones y submarinos	Médicos
Colaborativos		Exoesqueletos	Educativos
		Militares	

Tabla 3. Clasificación de los robots por función o sector. Fuente: elaboración propia.

Estos serían los grandes grupos que engloban lo que son los robots más relevantes en el campo de la robótica. Cada uno de ellos tiene sus especificaciones y aplicaciones según nuestras necesidades. El continuo desarrollo de las nuevas tecnologías está avanzando tanto que está impulsando el desarrollo y la diversificación de robots en el futuro.

Seguidamente, se procede a explicar con más detalle estos dos grandes tipos de robots, así como cada uno de sus subtipos.

- **Robots de servicio:** Estos robots se utiliza principalmente en entornos no industriales, y por lo general ofrecen servicios a las personas. Algunos de los tipos más comunes de los robots de servicio son los siguientes:
  - *Humanoides:* Estos robots tiene la peculiaridad de tener una apariencia humana y tienden a imitar el comportamiento humano. Se utiliza para investigación, asistencia personal y entretenimiento.
  - *Vehículos autónomos:* Estos robots, al igual que los coches autónomos, son dispositivos capaces de circular por sí mismos en las carreteras sin intervención de humanos. Esto es gracias a los sistemas de lectura (sensores) y toman decisiones instantáneas.
  - *Drones y Submarinos:* Estos vehículos no tripulados se utilizan para investigación y exploración, aunque también hay personas que los utiliza por simple hobby. Los drones son dispositivos aéreos, mientras que los submarinos son dispositivos acuáticos.

- *Exoesqueletos:* Estos dispositivos se adaptan al cuerpo de la persona y reducen el esfuerzo requerido para realizar movimientos o tareas más físicas. Se utilizan tanto en rehabilitación para personas.
- *Robots de Consumo:* Estos robots están diseñados para el entretenimiento, como juguetes para niños, o para tareas sencillas del hogar, como limpiar o barrer. Por lo general, no son programables.
- *Robots educativos:* Estos robots sí que son programables, se utilizan en colegios e instituciones educativas para fomentar la curiosidad, para aprender y desarrollar habilidades.
- *Robots médicos:* Estos robots se utilizan en el campo de la medicina, como robots quirúrgicos o superordenadores para simulación, gracias a su alta potencia computacional y resolución de algoritmos.
- *Robot Zoomórficos:* Estos dispositivos toman la forma de animales y también intentan imitar su comportamiento. Pueden tener diversas aplicaciones, desde apoyo emocional hasta el uso terapéutico.
- *Robots militares:* Estos dispositivos están diseñados específicamente para el uso en aplicaciones y operaciones militares. Su objetivo es proporcionar apoyo a las fuerzas armadas en todo tipo de escenarios, mejorando así la eficiencia operacional y evitando al máximo las bajas humanas posibles. Entro des este tipo de robots hay varios tipos como los terrestres, aéreos, acuáticos, aéreos de combate y robots logísticos.

- **Robots industriales:** Estos robots nacieron de la necesidad de aumentar la producción industrial y reducir los costes de producción. Normalmente, se utilizan para realizar tareas repetitivas, tareas con elevado peso o tareas que pueden suponer un riesgo para los trabajadores. A continuación, se nombran algunos de los tipos más comunes de robots industriales:
  - *Robots cartesianos*: son robots lineales constituidos por tres ejes y que los permite mover por el sistema de coordenadas cartesianas como su nombre indica (X, Y, Z). Gracias a la tipología es un dispositivo flexible en cuanto a velocidad, precisión y mesure del robot. Es un sistema común en las máquinas de control numérico por ordenador (CNC)
  - *Robots SCARA*: proveniente del acrónimo en ángulos *Selective Compliance Assembly Robot Arm* son robots con el espacio de trabajo

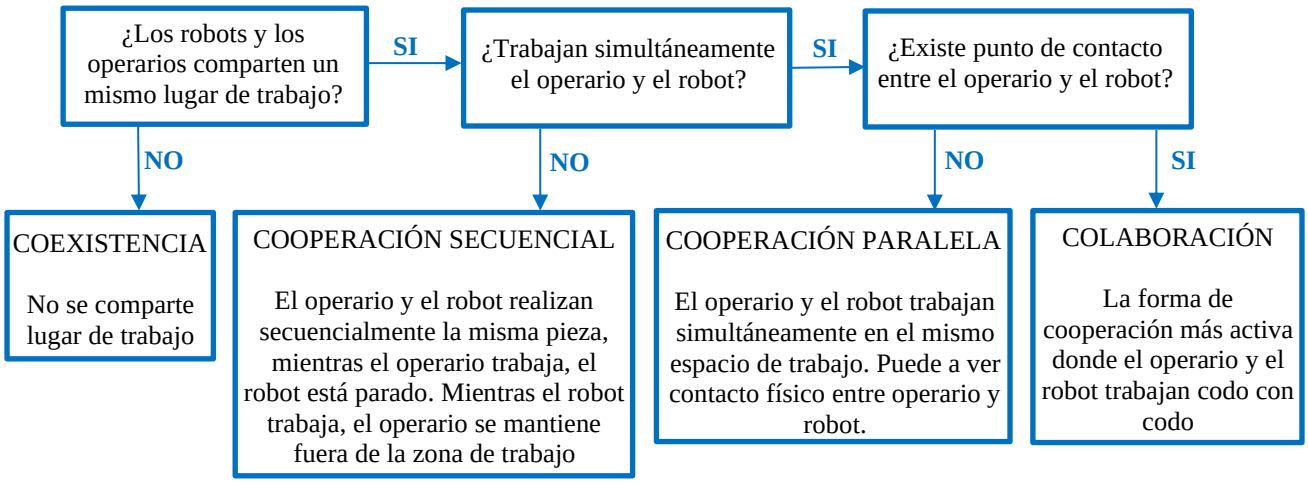
igual que los cartesianos, pero todas las articulaciones son de revolución y una articulación prismática y destacan por la rapidez de los movimientos y la precisión. Son muy utilizados en aplicaciones de paletización y desplazamientos de objetos.

- *Robots cilíndricos*: muy parecidos a los SCARA, pero constan solo de un eje de revolución y dos esos prismáticos. Suelen ser de dimensiones compactas y utilizados en aplicaciones sencillas de ensamblaje de piezas.
- *Robots delta*: también conocidos como robots paralelos, son robots formados por tres brazos conectados en un punto común. La posición de la base del elemento terminal viene determinada por los movimientos simultáneos de los tres brazos. Debido a las limitaciones, el área de trabajo es menor, pero permite moverse a altas velocidades y precisión. Utilizado en industrias alimentarias por la selección y colocación de los productos.
- *Robots polares*: son robots con 3 movimientos, dos articulaciones de rotación en la base y un brazo con una articulación lineal. Cómo indica el nombre, trabajan a partir de coordenadas polares. Fueron uno de los primeros modelos de robots utilizados por la manipulación de materiales, a la industria de la fundición y trabajo de metales, etc.
- *Robots articulados*: son robots de tres o más esos que permiten efectuar movimientos más complejos que se aproximan a los movimientos de un brazo humano. Son muy utilizados en la industria automovilística por las soldaduras en arco, manipulación de materiales, por ensamblaje, etc.
- *Robots colaborativos*: también conocidos con el nombre de *Cobot*, proveniente del inglés *Collaboratie Robots*, son robots articulados que permiten llevar a cabo tareas con interacción segura con humanos en un espacio de trabajo compartido.

Dado que este trabajo consta de la puesta en marcha de un robot colaborativo, en el siguiente apartado se profundiza sobre estos tipos de robots

A. Robots colaborativos

Los robots colaborativos son aquellos que permiten una colaboración directa entre humano-robot para realizar tareas comunes [2]. Según el tipo de interacción entre estas ambas partes, se puede determinar el tipo de colaboración.



Tal como se muestra en la figura anterior se definen cuatro tipos de operaciones colaborativas HRC (*Human-Robot Collaboration*). En la coexistencia y la cooperación secuencial no hay una colaboración simultánea, por lo tanto, no es necesario un robot colaborativo dado que las tareas las puede hacer un robot convencional/tradicional. En cambio, en la cooperación paralela y la colaboración, el humano y el robot trabajan en el mismo tiempo en un mismo puesto de trabajo. En este segundo caso sí que es necesario este tipo de robot.

Según el tipo de sistemas que incorporan, se pueden distinguir cuatro modos de operación en la robótica colaborativa [14].

- **Modo de parada de seguridad (*Safety Stop*):** En este modo, el robot se detiene inmediatamente en respuesta a cualquier entrada de seguridad, como la detección de un obstáculo o una interferencia inesperada. Este modo garantiza la seguridad del trabajador, pero puede interrumpir la productividad.
- **Modo de velocidad reducida (*Reduced Speed*):** El robot opera a una velocidad reducida cuando un trabajador se acerca demasiado. Esto permite que el robot continúe operando de manera segura, pero a una velocidad que minimiza el riesgo de lesiones en caso de contacto con un ser humano.
- **Modo de seguimiento (*Hand Guiding*):** En este modo, un operador humano puede guiar físicamente el movimiento del robot utilizando una interfaz de control manual. Esto es útil para tareas que requieren precisión y adaptabilidad, como el ensamblaje.
- **Modo de programación por demostración (*Teach Pendant or Tablet Programming*):** Los cobots a menudo se pueden programar mediante una programación por demostración, donde un operador humano mueve el robot a

través de la tarea deseada mientras graba los movimientos. El robot luego reproduce estos movimientos de manera autónoma.

- **Modo de colaboración directa (*Collaborative Operation*):** En este modo, el robot trabaja en estrecha colaboración con un ser humano sin detenerse ni reducir la velocidad. Los sensores y sistemas de seguridad del robot garantizan que pueda detectar y evitar obstáculos o personas en su camino, lo que permite una colaboración cercana y eficiente.
- **Modo mixto (*Mixed Mode*):** En este modo, el robot colaborativo puede alternar entre trabajar de forma autónoma y colaborar con un ser humano según sea necesario. Esto es útil en situaciones donde ciertas partes de una tarea son adecuadas para la automatización, mientras que otras requieren la intervención humana.
- **Modo de seguridad avanzada (*Advance Safety Mode*):** Algunos cobots tienen modos de seguridad avanzados que les permiten realizar tareas más complejas en entornos compartidos, gracias a sensores de alta precisión y algoritmos de detección avanzados. Esto puede incluir la capacidad de ajustar la velocidad y la fuerza en tiempo real según las condiciones del entorno.
- **Modo de programación *offline*:** En este modo, se puede programar el robot colaborativo utilizando software de programación en una computadora, lo que permite la simulación y la planificación de tareas sin que el robot esté en funcionamiento.

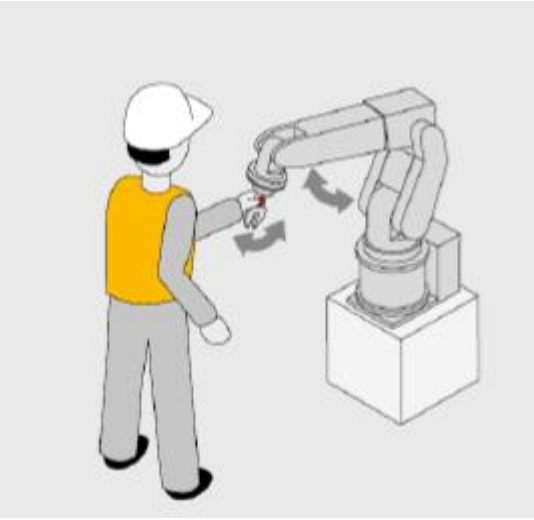


Ilustración 13. Representación guiado manual. Fuente: The Safety Compendium

Ilustración 12. Representación de parada monitorizada. Fuente: The Safety Compendium

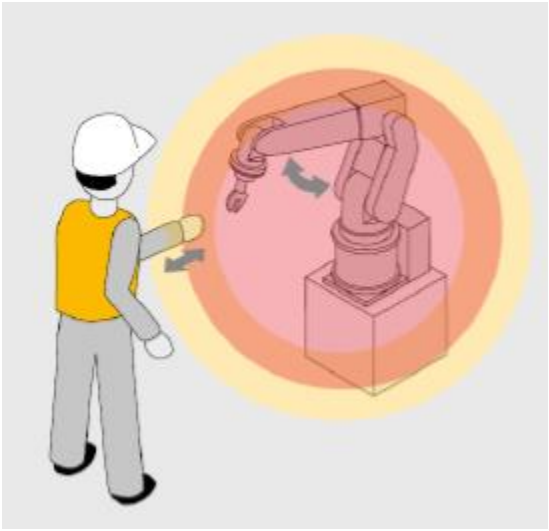


Ilustración 15. Representación monitorización de velocidad y separación. Fuente: The Safety Compendium

Ilustración 14. Representación limitada de potencia y fuerza. Fuente: The Safety Compendiumc

Los robots colaborativos podrían tener algunas similitudes con los robots tradicionales. Sin embargo, son más las diferencias y ventajas que trae la implementación de cobots en las empresas [3].

ROBOTS INDUSTRIALES	ROBOTS COLABORATIVOS
Son ideales para grandes compañías con grandes procesos de producción.	Diseñados para maximizar la producción de cualquier tipo de empresa.
Requiere un proceso extensivo de programación que puede llevar días o semanas de configuración.	Es fácil de programar, incluso por usuarios que no hayan tenido experiencia con ingeniería.
Requieren ser instaladas en un entorno seguro y con diferentes medidas de protección para el trabajador.	Están diseñados para trabajar en conjunto con los humanos de forma segura en diferentes procesos.
Realizan siempre la misma acción ya que están configuradas usualmente para realizar un solo proceso.	Los cobots tienen la capacidad de ser programados para realizar diferentes tareas y en diferentes lugares.
Las empresas deben invertir un presupuesto muy alto para obtener este tipo de maquinaria. Pasarán algunos años antes de poder amortizar la inversión.	Los robots colaborativos son más económicos y amigables en su proceso de instalación y programación. Esto permite que exista un retorno de la inversión más rápido.

Tabla 4. Cobots vs robots industriales. Fuente: elaboración propia.

1.3. Estructura de los robots industriales

Un manipulador robótico consta de una secuencia de elementos estructurales rígidos, denominados enlaces o eslabones, conectados entre sí mediante juntas o articulaciones, que permiten el movimiento relativo de cada dos eslabones consecutivos [12].

Una articulación puede ser:

- Lineal (deslizante, traslacional o prismática), si un eslabón desliza sobre un eje solidario al eslabón anterior.
- Rotacional, en caso de que un eslabón gire en torno a un eje solidario al eslabón anterior.

Ilustración 16. Elementos estructurales de un robot industrial. Fuente: Robots industriales

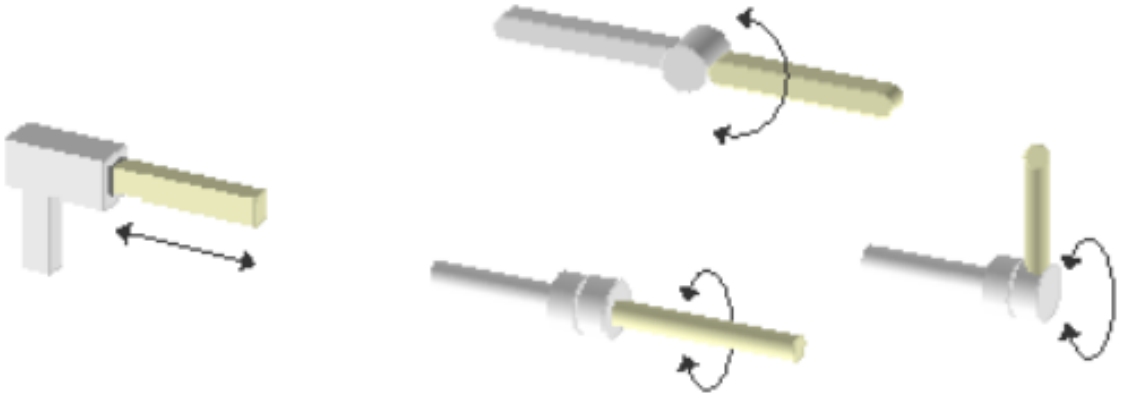


Ilustración 17. Distintos tipos de articulaciones de un robot: lineal (izquierda) y rotacionales (derecha). Fuente: Robots industriales.

El conjunto de eslabones y articulaciones se denomina cadena cinemática. Se dice que una cadena cinemática es abierta si cada eslabón se conecta mediante articulaciones exclusivamente al anterior y al siguiente, exceptuando el primero, que se suele fijar a un soporte, y el último, cuyo extremo final queda libre. A éste se puede conectar un elemento terminal o actuador final: una herramienta especial que permite al robot de uso general realizar una aplicación particular, que debe diseñarse específicamente para dicha aplicación: una herramienta de sujeción, de soldadura, de pintura, etc. El punto más significativo del elemento terminal se denomina punto terminal (PT). En el caso de una pinza, el punto terminal vendría a ser el centro de sujeción de la misma.

Ilustración 18. Punto terminal de un manipulador. Fuente: Robots industriales.

Los elementos terminales pueden dividirse en dos categorías:

- **Pinzas** (*gripper*): se utilizan para tomar un objeto, normalmente la pieza de trabajo, y sujetarlo durante el ciclo de trabajo del robot. Hay una diversidad de métodos de sujeción que pueden utilizarse, además de los métodos mecánicos obvios de agarre de la pieza entre dos o más dedos. Estos métodos suplementarios incluyen el empleo de casquillos de sujeción, imanes, ganchos, y cucharas.
- **Herramientas**: se utilizan como actuador final en aplicaciones en donde se exija al robot realizar alguna operación sobre la pieza de trabajo. Estas aplicaciones incluyen la soldadura por puntos, la soldadura por arco, la pintura por pulverización y las operaciones de taladro. En cada caso, la herramienta particular está unida a la muñeca del robot para realizar la operación.

A los manipuladores robóticos se les suele denominar también brazos de robot por la analogía que se puede establecer, en muchos casos, con las extremidades superiores del cuerpo humano.

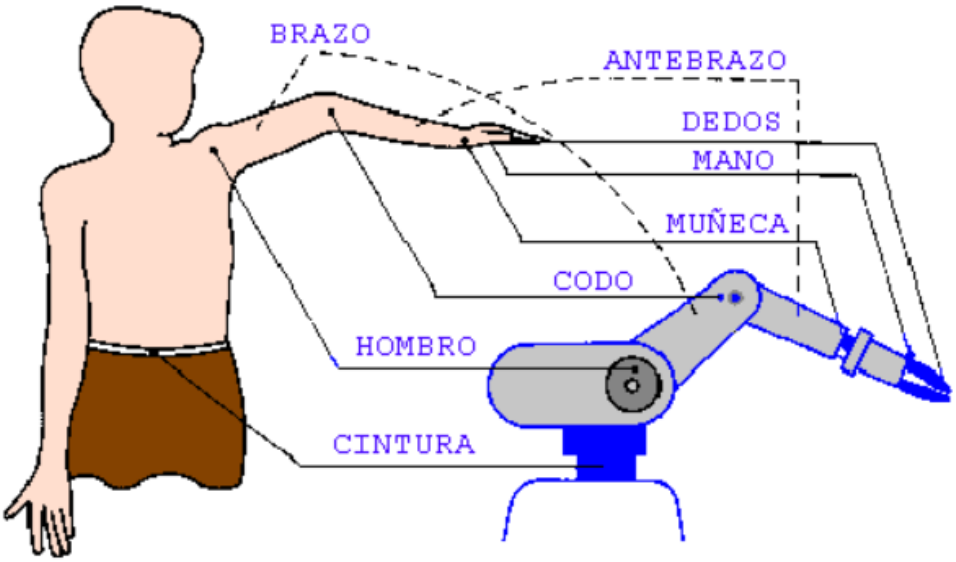


Ilustración 19. Semejanza de un brazo manipulador con la anatomía humana. Fuente: Robots industriales.

Se denomina grado de libertad (GdL) a cada una de las coordenadas independientes que son necesarias para describir el estado del sistema mecánico del robot (posición y orientación en el espacio de sus elementos). Normalmente, en cadenas cinemáticas abiertas, cada par eslabón-articulación tiene un solo grado de libertad, ya sea de rotación o de traslación. Pero una articulación podría tener dos o más GdL que operan sobre ejes que se cortan entre sí.

Ilustración 20. Distintos grados de libertad de un brazo de robot Fuente: Robots industriales.

Para describir y controlar el estado de un brazo de robot es preciso determinar:

- La posición del punto terminal (o de cualquier otro punto) respecto de un sistema de coordenadas externo y fijo, denominado el sistema mundo.
- El movimiento del brazo cuando los elementos actuadores aplican sus fuerzas y momentos.

El análisis desde el punto de vista mecánico de un robot se puede efectuar atendiendo exclusivamente a sus movimientos (estudio cinemático) o atendiendo además a las fuerzas y momentos que actúan sobre sus partes (estudio dinámico) debidas a los elementos actuadores y a la carga transportada por el elemento terminal.

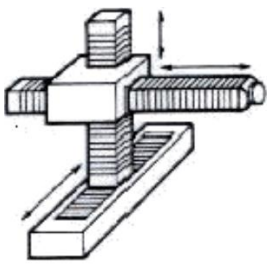
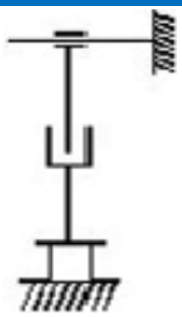
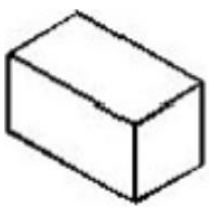
1.4. Configuraciones morfológicas y parámetros característicos de los robots industriales

Según la geometría de su estructura mecánica, un manipulador puede ser [13]:

- **Cartesiano**: su posicionamiento en el espacio se lleva a cabo mediante articulaciones lineales.
- **Cilíndrico:** una articulación rotacional sobre una base y articulaciones lineales para el movimiento en altura y en radio.
- **Polar**: cuenta con dos articulaciones rotacionales y una lineal.
- **Esférico** (o de brazo articulado), con tres articulaciones rotacionales.
- **Mixto**: posee varios tipos de articulaciones, combinaciones de las anteriores. Es destacable la configuración SCARA (*Selective Compliance Assembly Robot Arm*)
- **Paralelo**: posee brazos con articulaciones prismáticas o rotacionales concurrentes.

Los principales parámetros que caracterizan a los robots industriales son:

- **Número de grados de libertad (GdL)**: se entiende por grados de libertad de una cadena cinemática (conjunto mecánico formado por barras rígidas unidas por articulaciones con un extremo fijo y un móvil) el conjunto de variables para definir la configuración de la cadena. Dedo de otro modo, corresponde al número de actuadores para mover la cadena. En el caso de robots con cadena cinemática abierta (ex: robots articulados, UR3e) los grados de libertad corresponden al número de articulaciones. En cambio, los robots con cadena cinemática cerrada (ex: robot dual-arm, Mitsubhishi SCARA RP) tienen un número inferior de grados de libertad.
- **Espacio de accesibilidad o espacio (volumen) de trabajo**: es el conjunto de puntos del espacio accesibles al punto terminal, que depende de la configuración geométrica del manipulador. Un punto del espacio se dice totalmente accesible si el PT puede situarse en él en todas las orientaciones que permita la constitución del manipulador y se dice parcialmente accesible si es accesible por el PT, pero no en todas las orientaciones posibles. En la figura inferior se aprecia el volumen de trabajo de robots de distintas configuraciones.
- **Capacidad de posicionamiento del punto terminal**. Se concreta en tres magnitudes fundamentales: resolución espacial, precisión y repetibilidad, que miden el grado de exactitud en la realización de los movimientos de un manipulador al realizar una tarea programada.
- **Capacidad de carga**. Es el peso que puede transportar el elemento terminal del manipulador. Es una de las características que más se tienen en cuenta en la selección de un robot dependiendo de la tarea a la que se destine.
- **Velocidad**. Es la máxima velocidad que alcanzan el PT y las articulaciones.

CONFIGURACIÓN GEOMÉTRICA	ESTRUCTURA CINEMÁTICA	ESPACIO DE TRABAJO	EJEMPLO
Cartesianos 			

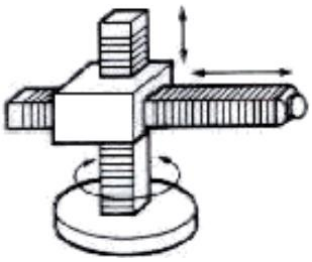
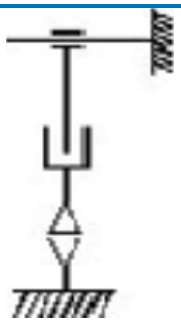
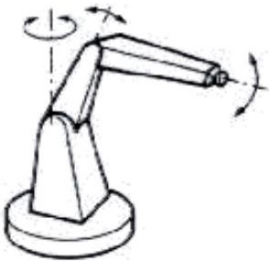
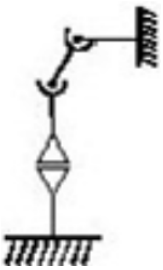
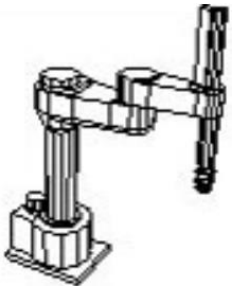
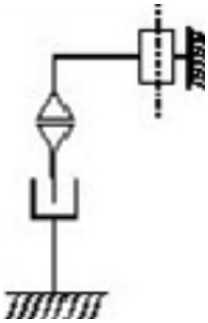
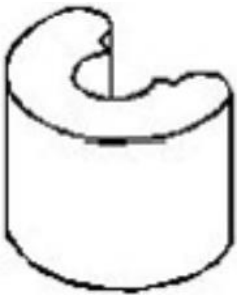
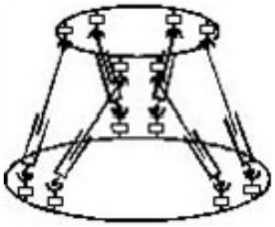
Cilíndrico 			
Polar 			
Esférico 			
SCARA 			
Paralelo 			

Tabla 5. Configuraciones geométricas, estructura cinemática, espacio de accesibilidad y ejemplos de robots industriales. Fuente: elaboración propia.

1.5. Seguridad para robots colaborativos

Los fabricantes de una amplia variedad de industrias emplean robots colaborativos ("cobots") para aprovechar los beneficios de las funciones de seguridad integradas que permiten que estos robots compartan las tareas con los seres humanos y se adapten de manera flexible a los nuevos requisitos de las nuevas tareas repetitivas. Aunque la seguridad es una característica clave de su diseño, los "cobots" aún requieren evaluaciones de riesgo [15].

Los robots colaborativos están diseñados para trabajar con los operadores humanos gracias a las tecnologías como retroalimentación de fuerza, servomotores de baja inercia, actuadores elásticos y tecnología de detección de colisiones que limitan su potencia y capacidades de fuerza a niveles adecuados para el contacto. Los "cobots", que son más compactos que los robots convencionales, generalmente tienen marcos ligeros con bordes suaves y redondeados.

La norma de seguridad ISO 10218 y la especificación técnica RIA TS 15066 definen las funciones de seguridad y el rendimiento de los robots. Conforme a la TS 15066, la supervisión de la fuerza y la velocidad se basa en los datos de la aplicación, el área de contacto humano y los peligros del espacio de trabajo. Los datos de la aplicación, el posible contacto humano y los peligros del espacio de trabajo se consideran dentro de las configuraciones de seguridad calculadas de la norma.

Los robots colaborativos están diseñados para trabajar con los operadores humanos gracias a las tecnologías como retroalimentación de fuerza, servomotores de baja inercia, actuadores elásticos y tecnología de detección de colisiones que limitan su potencia y capacidades de fuerza a niveles adecuados para el contacto. Los "cobots", que son más compactos que los robots convencionales, generalmente tienen marcos ligeros con bordes suaves y redondeados [16].

En el *Anexo I Normativa de seguridad del robot* se encuentra el extracto del documento UNE-EN ISO 10218-1.

1.6. Redes de comunicación industrial

En el ámbito de la automatización de procesos industriales, la conectividad de las redes de comunicación entre los diferentes dispositivos que intervienen en el control de estos sistemas, son muy importantes. Esto asegura su correcto funcionamiento, y también facilita el control efectivo de los procesos.

La maquinaria industrial con capacidad de comunicación tiene que funcionar utilizando

un protocolo de comunicación ya sea de una forma autónoma o con la manipulación humana. Para un intercambio de datos, es necesario el control de datos y la posibilidad de poder conectar maquinas o dispositivos que sean de fabricantes distintos para que puedan ser útiles en una misma instalación. [18].

1.6.1 Características

La comunicación industrial es la conversión y transmisión de información, desde un emisor a un receptor por un medio, que ya puede ser por fibra óptica, cable de cobre, o inalámbrico etc.

Las redes de comunicación industrial están diseñadas y construidas para manejar el control en tiempo real y la integridad de los datos a la vez que se instalan en grandes plantas que pueden operar en entornos difíciles.

Hay varias formas de implementar una estructura de comunicación en función de los protocolos de comunicación necesarios una de ellas y la que se está utilizando más son los protocolos de la interconexión de sistemas abiertos (OSI).

1.6.2 Funciones

Estas comunicaciones industriales se pueden llegar a utilizar en los sistemas de control para pasar datos entre distintos PLC's o entre PLC's y los ordenadores para el control y el almacenamiento de los datos.

Esto seria los controladores más utilizados la industria para el control de automatizaciones industriales: DCS (Sistema de Control Distribuido), PLC (Controladores Lógicos Programables), SCADA (Control de Supervisión y Adquisición de Datos),

Cuando se habla de red se refiere a la conexión entre dispositivos para poder intercambiar datos entre ellos a través de ese circuito, donde esa red lleva bytes, que se envían secuencialmente entre los dispositivos conectados. Ese conjunto de datos en un solo envío se le llama paquete. La velocidad de transmisión de datos son bits por segundo (bps), pero también se expresarse en miles (Kbps) o millones (Mbps).

1.6.3 Realización de las redes

Todas las comunicaciones entre dispositivos industriales se realizan con la ayuda de los protocolos de comunicaciones industriales. Un protocolo de comunicación es un conjunto de reglas que permite la transferencia e intercambio de datos entre los dispositivos a comunicar. Esto ha permitido, una mejor gestión de los procesos de producción, un mejor acceso a la información de los dispositivos, de una manera centralizada.

Con la necesidad de integrar la información entre las fábricas de la misma empresa se

hico la integración de estos sistemas, distribuyendo las comunicaciones en varias capas: buses de campo, redes LAN y redes LAN-WAN.

Se utilizan muchos protocolos de redes de comunicación industrial, algunos de son específicos para su marca, y otros lo hacen lo más estándares posible. Eso genera un problema de compatibilidad entre dispositivos de distintas marcas muchas veces.

1.6.4 Niveles de comunicación

Los sistemas de automatización industrial están estructurados por varios niveles jerárquicos. Estos niveles tienen cada uno de ellos un nivel de comunicación que plantea diferentes exigencias sistema de comunicación.

Dependiendo de la jerarquía de los sistemas de automatización y control industrial, el flujo de datos se establece en una dirección horizontal o vertical.

El flujo vertical garantiza la comunicación a través de redes para la planificación, visualización y gestión de la producción. El flujo horizontal se establece de forma local entre los dispositivos de campo utilizando buses de datos. La manera de realizar los nexos puede ser por cable o inalámbrico.

Los sistemas de redes de comunicación industrial se pueden clasificar en diferentes categorías basadas en la funcionalidad:

- Redes a nivel de campo
- Redes a nivel de control
- Redes a nivel de información

1.6.5 Nivel de Campo

Es el nivel más bajo, es donde están los dispositivos, como sensores y actuadores, módulos de E/S y unidades de accionamiento de las máquinas.

El envio de los datos es cíclica y se caracteriza porque tiene un ciclo de bus corto. La duración real dependera de la aplicación.

En este nivel simplemente se transmiten unos pocos bytes al mismo tiempo,

Profibus DP y Profinet IO serian un ejemplo de este nivel y ofrecen las soluciones universales tanto para la automatización de fábricas como para la automatización de procesos.

1.6.6 Nivel de Control

Los PLC’s, se comunican entre sí y con los sistemas informáticos de la empresa utilizando estándares como Ethernet TCP/IP, Intranet e Internet.

Dentro de este nivel es donde están todos los sistemas informáticos de automatización

que controlan el proceso. La información se transfiere datos de unos pocos bytes a unos pocos kilobytes. Este flujo de información requiere paquetes de datos y una serie de protocolos de comunicación. Un ejemplo serio Profibus, Profinet, basado en Ethernet, ofrece una solución que está siendo bastante utilizada por las empresas para este fin.

1.6.7 Nivel de Información

Este nivel es el más alto de un sistema de automatización industrial. El controlador de este nivel es el que reúne la información de gestión de los niveles de área y controla todo el sistema de automatización con el Sistema MES y ERP.

El sistema MES está conectado directamente con el nivel de control y los datos de producción actuales. Lo que hacen es un seguimiento de la información de los elementos y de las peticiones de la planta, recopilan todos los movimientos de información.

Por otro lado, el ERP recopila y organiza los datos empresariales a través de un paquete de software integrado. El ERP tiene un software que contiene aplicaciones que automatizan funciones empresariales como la producción, la cotización de ventas, la contabilidad, etc.

Un ejemplo de este nivel seria WAN Ethernet que sería para la planificación de la fábrica y el intercambio de información de gestión.

1.6.8 Tipos de protocolo en comunicaciones

Un protocolo de comunicación, es un conjunto de normas para la comunicación entre dispositivos de una red, dependiendo del tipo de red esas normas serán diferentes. Los protocolos más comunes en la industria serían los siguiente: Modbus RTU, Ethernet TCP/IP, Modbus TCP/IP, Profinet, Profibus, Ethercat, etc.

Estos protocolos son compatibles con un gran nombre de dispositivos de marcas distintas. Lo que el rendimiento de los protocolos serie hacen que no sean buena opción para aplicaciones de alta velocidad Por ese motivo y a otras ventajas mas, Ethernet es el estándar más usado para la capa física de muchos protocolos industriales, como EtherNet/IP, Ethernet TCP/IP, Modbus TCP/IP y Profinet.

1.6.9 Topologías

Una topología de red es una representación gráfica de cómo están conectados los dispositivos entre ellos. Es la estructura topológica de una red, y definen el aspecto físico como la lógica de la red.

La topología física es la colocación de los distintos componentes de la red, incluyendo la ubicación de los dispositivos y su conexión, mientras que la topología lógica muestra

cómo fluyen los datos dentro de una red, menos su diseño físico.

La topología lógica como la física pueden ser distinta o iguales en una misma red.

En las redes industriales más frecuentes serían las siguientes:

A. Redes punto a punto

Es la conexión más sencilla es una comunicación punto a punto entre dos dispositivos. Como sería una conexión entre un PLC y un PC.

Una desventaja es que, si un dispositivo tiene que comunicarse con otros dispositivos, hay que establecer una conexión distinta.

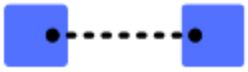


Ilustración 21. Red punto a punto. Fuente: Sicma21

B. Topología de bus

Esta conexión es en serie de dispositivos para formar una topología de línea. Todos los dispositivos están conectados a un mismo medio. Un ejemplo de esta topología seria Profibus.

Ilustración 22. Topología de bus. Fuente: Sicma21

C. Topología de estrella

En esta topología se requiere un componente de distribución que está en el centro el centro, donde nodo está conectado de forma individual al punto de conexión central, como un hub o un switch.

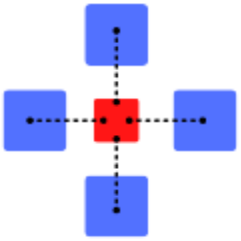


Ilustración 23. Topología de estrella. Fuente: Sicma21

D. Topología de árbol

Esta topología tiene distintos componentes de distribución, dependiendo de su tamaño, por eso se puede decir que es una estrella expandida.

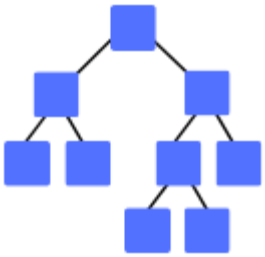


Ilustración 24. Topología de árbol. Fuente: Sicma21

1.7. IMU

Una Unidad de Medida Inercial (IMU) es un dispositivo capaz de estimar y reportar estados dinámicos específicos como velocidades angulares y aceleraciones. A partir de

estas medidas otros estados dinámicos se pueden inferir, tales como la latitud (roll y pitch), o incrementos en la velocidad y la posición de la plataforma. También puede ser que algunos lleven magnetómetro adicional, para poder medir el campo magnético alrededor del dispositivo.

Las IMUs son el componente principal de los sistemas de navegación inercial utilizados en aeronaves, vehículos aéreos no tripulados (UAVs) y otros sistemas no tripulados, así como misiles e incluso satélites.

1.7.1. Sector de uso

La navegación inercial depende solo de la contribución de los sensores directamente contenidos en la plataforma que no dependen de un sistema externo, (al contrario que el sistema GNSS) y que, por lo tanto, no son susceptibles de manipulación externa.

En un sistema de navegación inercial, donde los datos sin procesar recopilados por el IMU se procesan más tarde por una computadora que, aplicando diferentes algoritmos, es capaz de estimar la latitud, posición y velocidad de la plataforma, basándose únicamente en los datos aportados por los sensores y resultados de las estimaciones de iteraciones anteriores.

Los diferentes algoritmos ejecutados por el sistema son capaces de fusionar las entradas de múltiples sensores del IMU. Esta fusión de sensores, permite detectar datos erróneos o fuera de rango de un determinado sensor, por ejemplo, un magnetómetro afectado por el campo magnético de un objeto cercano. De esta manera, el sistema aísla dicho sensor y sus medidas compensando esta situación con el resto de sensores disponibles y reduciendo el error y la deriva en la estimación. Esto hace que el sistema sea robusto frente a fallos individuales o múltiples de sensores [17].

1.7.2. Componentes

Una IMU se compone normalmente de:

- **Acelerómetros**: miden las fuerzas gravitatorias en un sistema de coordenadas fijo. Por ejemplo, un acelerómetro en reposo sobre la superficie de la Tierra medirá '-***1g' o -9,8 m/s2. Cuando la plataforma esté en movimiento las fuerzas de inercia*** aparecen. Por esta razón, a menudo se dice que los acelerómetros proporcionan una señal acoplada. Puede tener acelerómetros en los tres ejes de coordenadas.
- **Giróscopos**: miden la velocidad angular. Un giróscopo mecánico incluye una

rueda o disco giratorio. Gracias a la conservación del momento angular, cualquier cambio en la orientación del eje de la rueda será registrado por el sensor; por lo tanto, se puede calcular el cambio de orientación de la plataforma. Se utilizan diferentes tecnologías y principios físicos en la construcción de giróscopos. Estos incluyen los giróscopos de fibra óptica (FOG) más precisos basados en el efecto Sagnac y también las unidades microelectromecánicas (MEMS) menos precisas que se basan en el cálculo de la fuerza de Coriolis por medio de diminutas estructuras vibratorias. Los giróscopos son esenciales para el cálculo de la orientación, pero pueden sufrir deriva, incluso cuando están estáticos.

- **Magnetómetros**: miden el campo magnético local. Un tipo simple de magnetómetro es una brújula, que mide la dirección del campo magnético de la Tierra en 2D. En los últimos años, los magnetómetros se han miniaturizado. El campo magnético de la Tierra es un vector tridimensional que, como la gravedad, se puede utilizar para determinar la orientación a largo plazo.

1.7.3. Cálculo de Euler

La fórmula de Euler es un concepto matemático que relaciona dos conceptos elementales de las matemáticas: los números complejos y la trigonometría. Esto lo convierte en una de las conceptualizaciones más importantes y con más aplicaciones de toda la matemática. La fórmula de Euler es una ecuación matemática fundamental basada en el número de Euler, la cual relaciona los números complejos con la trigonometría. Fue descubierta por el matemático suizo Leonhard Euler en el siglo XVIII y desde entonces ha sido utilizada en una variedad de campos, desde la física hasta la informática [18].

La fórmula de Euler se escribe como $\boldsymbol{e^{ix} = \cos(x) + i \cdot sen(x)}$ **,** donde $\boldsymbol{e}$ es la base del logaritmo natural, $\boldsymbol{i}$ es la unidad imaginaria (definida como la raíz cuadrada de -1) y $\boldsymbol{x}$ es un número real. Esta ecuación especifica que el número complejo $\boldsymbol{e^{ix}}$ es igual a la suma del número real **cos(x)** y del producto del número imaginario i por el número real **sen(x)**. La importancia de la fórmula de Euler radica en que permite expresar los números complejos en términos de números reales y trigonometría, lo que facilita su manejo y cálculo.

La demostración de la fórmula de Euler se basa en el uso de la serie de Taylor para la función exponencial y la identidad trigonométrica para coseno y seno. Primero, consideramos la serie de Taylor para la función exponencial:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots \quad \textbf{[Eq.1]}$$

Luego, reemplazamos x por ix en la ecuación anterior [Eq.1], donde i es la unidad imaginaria (raíz cuadrada de -1):

$$e^{ix} = 1 + ix + \frac{(ix)^2}{2!} + \frac{(ix)^3}{3!} + \frac{(ix)^4}{4!} + \frac{(ix)^5}{5!} + \cdots \quad \textbf{[Eq.2]}$$

Entonces, aplicamos las potencias de i y reemplazamos en la ecuación anterior [Eq.2]:

$$e^{ix} = 1 + ix - \frac{x^2}{2!} - \frac{ix^3}{3!} + \frac{x^4}{4!} + \frac{ix^5}{5!} + \cdots \quad \textbf{[Eq.3]}$$

Ahora, agrupamos los términos reales y los términos con i:

$$e^{ix} = \left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots\right) + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots\right) \quad \textbf{[Eq. 4]}$$

Finalmente, simplificamos (sustituyendo cada expresión de los paréntesis por coseno y seno de x) em [Eq.4] y logramos:

$$e^{ix} = \cos(x) + i \cdot sen(x) \quad \textbf{[Eq. 5]}$$

A. Ángulos de Euler (pitch, roll & yaw)

Los ángulos de Euler son una forma de describir la orientación de un cuerpo rígido en el espacio. Estos ángulos se pueden expresar de diferentes maneras, pero una forma común es mediante los ángulos roll, pitch y yaw. El ángulo roll se refiere a la rotación alrededor del eje X, el ángulo pitch se refiere a la rotación alrededor del eje Y y el ángulo yaw se refiere a la rotación alrededor del eje Z [19].

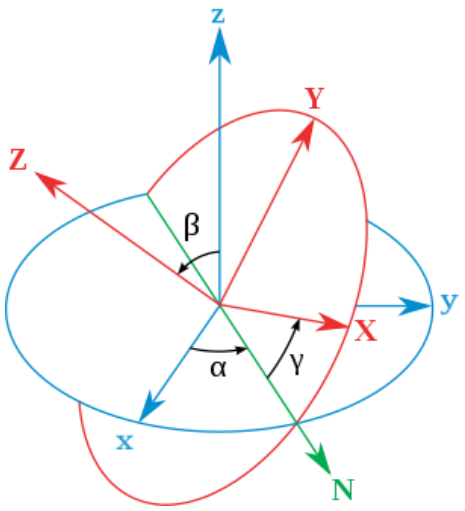


Ilustración 25. Definición geométrica de los ángulos clásicos de Euler. Fuente: Wikipedia

En una unidad de medición inercial (IMU), los ángulos de Euler se utilizan para estimar la actitud, la posición y la velocidad de la plataforma [4]. La relación entre el sistema de coordenadas del cuerpo y el sistema de coordenadas del suelo es de tres ángulos de Euler: guiñada, cabeceo y balanceo. La inclinación gira alrededor del eje X (ángulo de inclinación), la guiñada gira alrededor del eje Y (ángulo de guiñada) y el balanceo gira alrededor del eje Z (ángulo de balanceo).

2. UNIVERSAL ROBOT

Universal Robot es una empresa danesa con más de 1.000 empleados, que se dedica a fabricar brazos robóticos colaborativos de 6 ejes, seguros, flexibles e intuitivos para todo tipo de empresas y países. Consiguiendo así ser más versátil a la hora de utilizarlo para la faena que se le requiera.

Hasta la fecha ha sido capaz de vender más de 75.000 "cobots" (son los denominados robots colaborativos) en todo el mundo, que se utilizan en diferentes entornos productivos [17].

2.1. UR3 e

El UR3e consiste en un robot compacto de sobremesa, ya que tiene un formato compacto ideal para espacios de trabajo reducidos. Por sus dimensiones, es bastante versátil a la hora de su lugar de trabajo, ya sea en una mesa de trabajo o instalarlo directamente en maquinaria, por esas mismas condiciones es idóneo para aplicaciones ligeras de montaje y atornillado.

El UR3e también consta de una consola con la que se hace todo el control del robot. El UR3e también está disponible como un sistema robótico OEM [18] y con una consola de programación con 3 posiciones [17].

2.1.1. Características

Según la información del fabricante, es el robot colaborativo más flexible que tienen, dado que como hemos dicho anteriormente, es un "cobot" de sobremesa pequeño, perfecto para tareas de ensamblaje ligeras y bancos de trabajo automatizados.

Este robot pesa un total de 11 Kg, haciéndolo así poco pesado y fácil de instalar por su peso. Pero su carga útil es de 3 kg solamente, con una rotación de 360° en todos los ejes y una rotación infinita en la articulación final, equivaldría a la muñeca. Su alcance es de 500 mm y su huella, es decir, el espacio que ocupa en el espacio de trabajo, es de 128mm.

El robot colaborativo UR3 e, es un asistente perfecto para aplicaciones de pulido, ensamblaje, encolado, pulido y atornillado que requieran una calidad uniforme en el producto. También se utilizan en estaciones de trabajo independiente en cualquier lugar dado sus pequeñas dimensiones para ensamblar, recoger o colocando piezas líneas de producción optimizadas. Como hemos comentado con anterioridad, por su forma compacta y gracias a su fácil programación, se puede modificar su tarea para satisfacer las necesidades de la fabricación, que pueden ir cambiando, eso quiere decir, que el coste

para una nueva necesidad se reducirá el coste total de propiedad y un al ser tan versátil el período de amortización será muy rápido [4].

2.1.2. Lenguaje de programación

En cuanto al lenguaje de programación en la robótica no existe ningún estándar que normalice o regule este concepto. Los lenguajes genéricos más comunes que se pueden encontrar son: el Pascal que fue uno de los primeros lenguajes utilizados, el Scratch que se utiliza en la robótica educativa, Matlab por análisis de datos y desarrollo de sistemas de control, HDL (Hardware Description Language), Python que está en crecimiento en la actualidad, Java y C/C++ que es el lenguaje más utilizado en la robótica. Por otro lado, se encuentran los lenguajes que desarrolla cada fabricante de robots. De las marcas más comunes se tienen que: ABB utiliza el lenguaje Rapid, Kuka tiene el lenguaje KRL, Jaskawa utiliza INFORM, Kawasaki utiliza AS, Fanuc utiliza Karel y Universal Robots utiliza URScript.

Tal como se ha comentado anteriormente la marca Universal Robot apuesta por el lenguaje URScript y ofrece tres formas de programación con este lenguaje. Dos de estas formas se diferencian por la forma de programar y su complejidad, la tercera es una combinación de estas dos, dónde a partir de la programación intuitiva, permite llamar funciones programadas a través de scripts [20].

A. Programación PolyScope

El primer método que permite programar el robot es una programación intuitiva y esta se llevar a cabo a partir de la consola de programación del robot. Tal como se puede ver en la siguiente figura, este tipo de lenguaje es muy simple y de nivel bajo de programación, incluso en la columna izquierda tienes ayudas para usar plantillas o comandos.

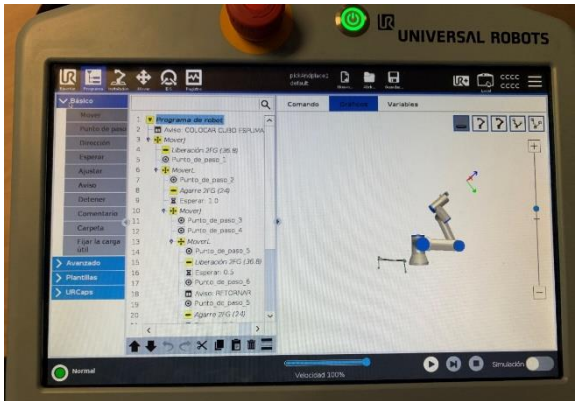


Ilustración 26. Programación intuitiva UR con comandos.

Fuente: Elaboración propia

El método de programación que se utiliza en el PolyScope está basado en bloques (en este caso bloques de funciones textuales) y un interfaz muy sencilla e intuitiva.

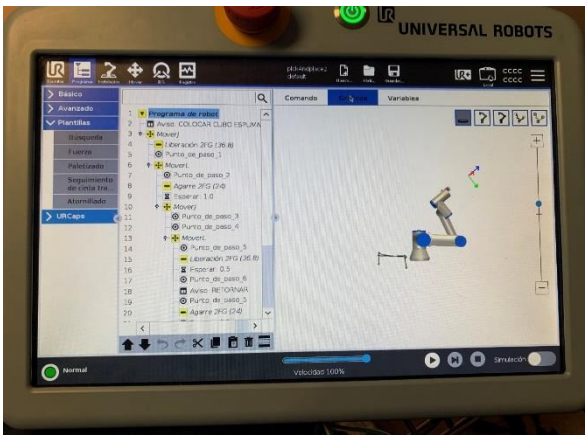


Ilustración 27. Programación intuitiva UR con plantillas.

Fuente: Elaboración propia

B. Programación por Scripts

El segundo método de programación es a partir del lenguaje de programación URScripts que es escritura directa. A diferencia de la programación anterior, permite crear programas más complejos y avanzados.

El inconveniente de este lenguaje es que desaparece la parte intuitiva obligándote a tener que tener un nivel de programación más avanzado, por lo tanto, se convierte en un hándicap para el operario que desconoce este lenguaje de programación, pero sabía utilizar la programación a través del PolyScope.

C. Programación por UR Caps

Este método de programación consiste en la combinación de los dos métodos anteriores, una combinación de la programación intuitiva de PolyScope con la opción de añadir parte de programación directamente con código URScript lo cual permite desarrollar aplicaciones más complejas.

2.1.3. Entorno de programación y simulación

Tal y como se ha comentado en el apartado anterior, la programación se puede llevar a cabo en el PolyScope. Esto permite no tener que depender de un ordenador externo conectado al robot. A pesar de que esto soluciona un problema, también es interesante disponer de una alternativa de programación para poder desarrollar los códigos sin tener que tener el robot físicamente. La propuesta del fabricante UR es poner a disposición una máquina virtual que simula con exactitud la consola de programación. Cuando se ejecuta

la máquina virtual que se denomina URSim (Universal Robot Simulator), permite escoger el modelo del robot que se quiere programar. Una vez abierta la aplicación específica se podrá comprobar que es el mismo entorno de programación que el que dispone el PolyScope.

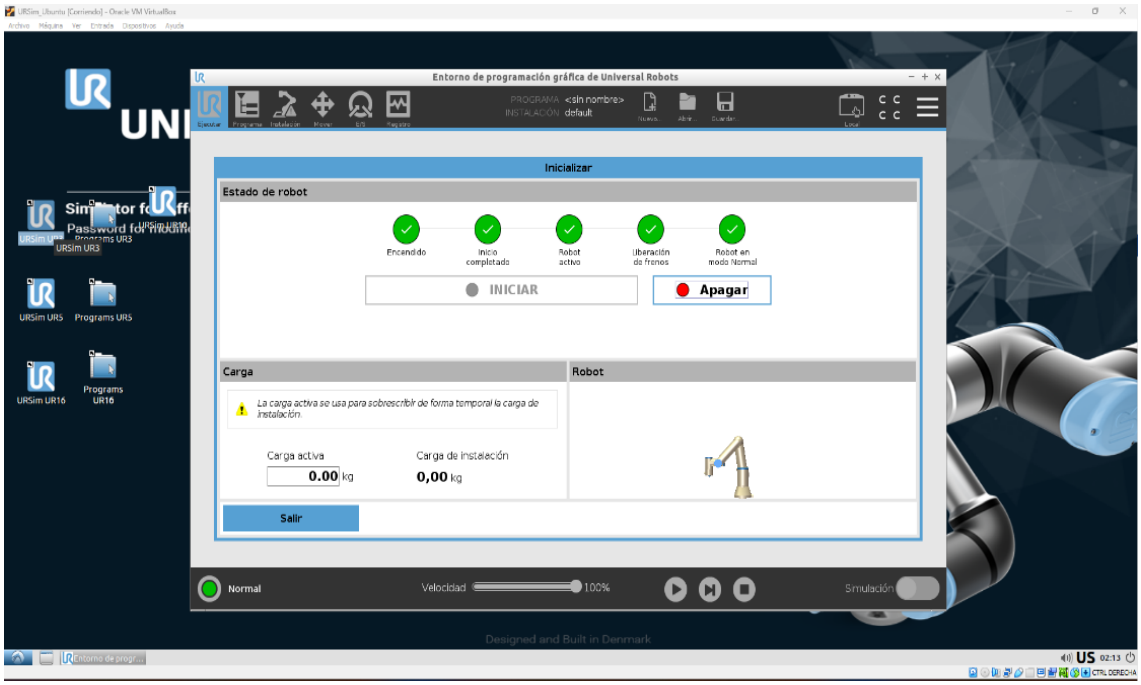


Ilustración 28. Captura máquina virtual y simulador URSim. Fuente: Elaboración propia

Esta función es una herramienta de gran utilidad en el ámbito educativo y el ámbito profesional dado que permite crear programas sin necesidad de tener todo el conjunto de los dispositivos de entrada, simular su funcionamiento e instalarlo sin necesidad de parar la producción o la línea que contiene el robot.

En un primer momento para poder utilizar el simulador para que sea controlado por los dispositivos externos, simplemente hay que ponerlo en marcha “encendiéndolo” como se puede ver en la siguiente imagen, y posteriormente conectar el microcontrolador en el mismo wifi que está conectada la máquina virtual. Importante mencionar que a diferencia del robot de verdad no hace falta ponerlo en modo remoto, ya que si no, no podrías ver como se mueve el robot, simplemente déjalo en local y veras como el robot va haciendo caso a las órdenes.

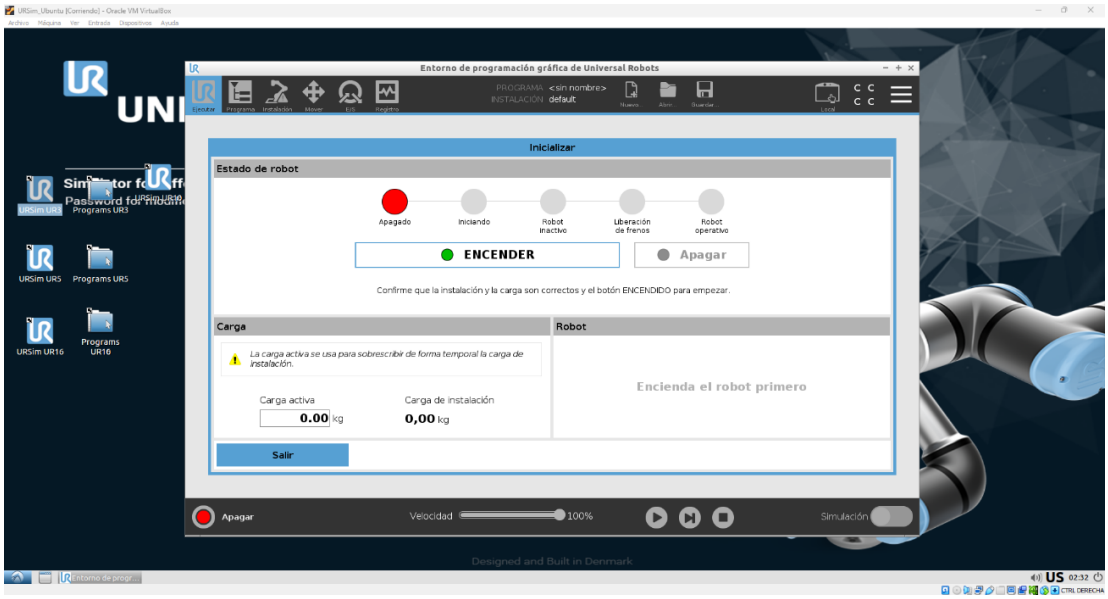


Ilustración 29. Puesta en marcha del robot simulador. Fuente: Elaboración propia

Lo que sí que hay que mirar es la dirección IP que tendrá este robot en el simulador, una vez encendido, para que se pueda realizar la comunicación por TCP/IP.

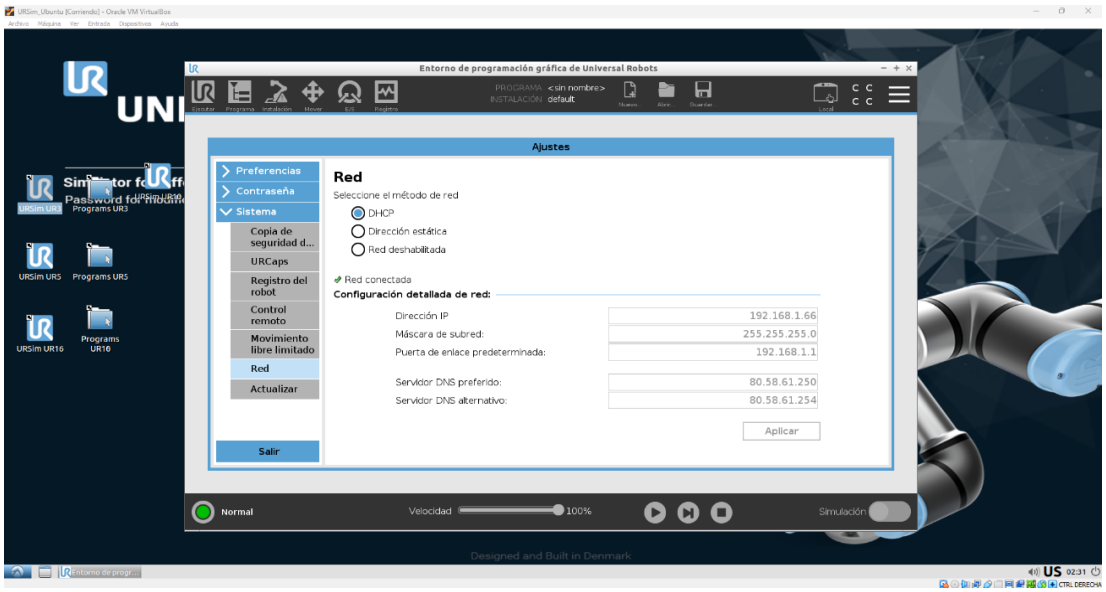


Ilustración 30. Direccion IP del robot simulador. Fuente: Elaboración propia

2.1.4. Comunicación

A partir del puerto de conexión Ethernet es posible establecer cuatro tipos de comunicaciones: Modbus TCP, Profinet y TCP/IP.

- **MODBUS TCP**: es uno de protocolo de comunicación diseñado específicamente por aplicaciones industriales destinado a la supervisión y control de equipos de

automatización que permite una comunicación en el entorno de Internet utilizando la arquitectura cliente/servidor, el protocolo TCP/IP [21]. Permite la comunicación de diferentes dispositivos de una misma red, los cuales poseen una dirección IP única, y ha acontecido un protocolo muy integrado por la industria a pesar de no haber pasado por ningún organismo de estandarización. El tipo de datos que es capaz de distinguir son: entradas digitales (discreto input), salidas digitales (coils), registros de entrada (input register) y registros de retención (holding registers).

- **POFINET**: del inglés Process Field Network, es un estándar industrial (IEC 61784-2) desarrollado por PROFIBUS International, por el intercambio de información a través de la red Ethernet por la lectura de datos y control de diferentes dispositivos. Creado para satisfacer la necesidad de intercambio de datos de manera rápida y en tiempo real (RT) que exigía la industria automática, logrando tiempo de ciclo inferiores a 31,25 μs [22].
- **TCP/ IP**: la comunicación por TCP/IP, donde también se le puede llamar Ethernet Socket, es un mecanismo de comunicación bidireccional entre dos dispositivos o dos procesos de un mismo programa basada en la filosofía cliente-servidor (el cliente hace la petición del servicio y el servidor se lo envía). A diferencia de las comunicaciones anteriores, no existe nodo de conexión dado que solo permite la comunicación directa entre dos dispositivos.

De las comunicaciones llamadas anteriormente se ha escogido trabajar con la comunicación TCP/IP porque es el único que nos ha permitido establecer una comunicación inalámbrica entre el robot y los microcontroladores.

Este tipo de comunicación permite controlar el robot desde un dispositivo externo sin necesidad de un programa que se ejecute al robot. Esta conexión, se puede llevar a cabo a través del servidor Dashboard, donde están los servidores alternativos para el control remoto del robot, proporciona acceso a funciones de todo tipo. Se tienen que hacer a través del servidor primario (puerto 30001), secundario (puerto 30002), que estos funcionan a 10 Hz o también hay la opción del real-time (puerto 30003), que funciona a una frecuencia de bucle de control total de 500 Hz. El puerto 30003, es el puerto elegido por las características mencionadas. Estos servidores son capaces de entender y ejecutar

comandos URScript, como los que se van a utilizar en este proyecto, que serían los 'movej', 'servoj' aunque hay muchos más que se podrían usar [23]. Tanto si se utiliza un puerto como otro, la programación se desarrolla a través del envío de secuencias de pedidos sin formato que son interpretadas y ejecutadas por el controlador del robot [20].

2.1.5. Entradas y salidas

Los puertos de entrada/salida y los puertos de comunicaciones que disponen se pueden encontrar en dos lugares diferentes. La gran mayoría están situados en la caja de control, pero también se pueden encontrar en el extremo del brazo articulado en el conector Lumberg de 8 pines.

En la caja de control (Figura 27), podemos encontrar los pines de conexión de entradas y salidas y conexiones de elementos de seguridad y accionamiento. Todos ellos trabajan con una tensión de 24 V. La primera fila situada a la parte más izquierda, los conectores de color amarillo, permiten conectar botones de parada de emergencia que funcionan paralelamente con el botón de parada de emergencia que hay de la consola de programación. Los conectores situados a su derecha permiten conectar botones u otros sistemas, como por ejemplo un PLC, para encender/apagar el robot sin hacer uso del botón de la consola de programación. Por otro lado, los que están situados de forma horizontal, son puertos utilizados para conectar una cinta de transporte y poder sincronizar el movimiento de este con el robot y permitir un trabajo síncrono sin necesidad de parar la cinta. Las siguientes cuatro filas de conectores de color amarillo, permiten conectar dispositivos de seguridad que actúen como entradas de señales (4 dispositivos) o salidas (4 dispositivos), por ejemplo: láseres de seguridad, plataformas sensibles, bloqueadores de puertas... Los siguientes conectores de color gris corresponden a pines de conexión de señales digitales, 16 pines de entrada y 16 pines de salida (8 dispositivos para cada tipo). Por último, el conector de color verde permite conectar dispositivos analógicos, en este caso contiene 2 entradas y 2 salidas analógicas.

En la parte inferior de la caja que contiene los dispositivos de control, se pueden encontrar (de izquierda a derecha): el conector por la consola de programación, una ranura por la tarjeta de memoria que contiene el sistema operativo, puerto de conexión Ethernet RJ-45, dos puertos de conexión USB (USB2.0 y USB3.0) y un puerto de conexión mini DisplayPort.

Respecto a la conexión Ethernet, como se puede ver en la Ilustración 27, se puede ver un cable conectado. Ese cable es el que va directamente al router que se ha instalado para poder realizar la comunicación inalámbrica por conexión wifi.

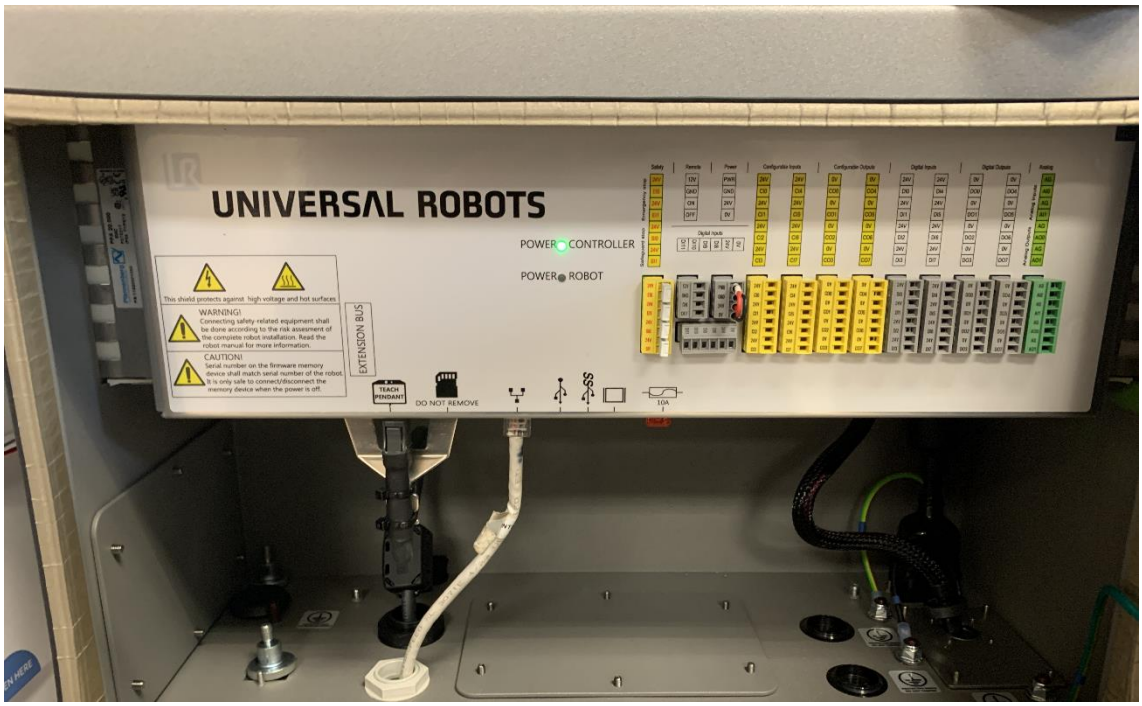


Ilustración 31. Interior controlador robot UR3e y los puertos de conexión. Fuente: Universal Robots Academy.

También comentar que el conector Lumberg, modelo KKMV 8-354, (el cable saliente conectado al robot) que se encuentra en la brida para conectar el elemento terminal, contiene 8 pines que corresponden a: un pin de tierra (GND), un pin de alimentación que se puede seleccionar entre 0V, 12V o 24V, dos pines correspondientes a la salida digital DO0 y DO1, dos pines correspondientes a las entradas digitales DI0 y DI1, dos pines correspondientes a entradas analógicas en AI2 o AI3. Los pines de las entradas analógicas también se pueden utilizar para establecer una comunicación a través del protocolo RS-485. El conjunto de estas conexiones están diseñadas para establecer una comunicación directa entre el elemento terminal y el robot y facilitar la lectura de los diferentes sensores o datos que este pueda transmitir.

3. DISPOSITIVO

En este apartado se va a hablar de las elecciones de los dispositivos que se han escogido para realizar este proyecto. A continuación, iremos una por una explicando que tipo de características que tiene cada uno de ellos.

3.1. Robot

En este punto la verdad que ya se han mencionado todas sus características principales en el punto "2.1.UR3 e". Por ese motivo también se han ido viendo cosas que son interesantes para este proyecto. Al empezar es verdad que el robot UR3 e, que se estaba en el proceso de compra por la universidad antes de empezar este proyecto, pero fue la condición indispensable de porque se ha hecho este proyecto. Es decir, que sin el robot no se habría podido plantear este proyecto. Ahora bien, se ha podido hacer, aparte de sus características físicas, su intuitivo y fácil manejo, es porque a diferencia de la gran mayoría de robots industriales, este robot se puede controlar a partir del protocolo de comunicación TCP/IP, además de tener API (Interfaz de Programación de Aplicaciones). El API es una herramienta que permite a los desarrolladores crear aplicaciones personalizadas para el robot. La documentación del API se puede encontrar en la página web de Universal Robot [36].

Para poder realizar las funciones para controlar el robot con un dispositivo externo al robot se tiene que utilizar el manual que está en el *"Anexo II Manual de funciones UR3e"*.

Ilustración 32. UR3 e. Fuente: Universal Robots e-Series Manual de usuario

3.2. Dispositivos de Comunicación

En este punto en un primer momento se planteó usar un Arduino que tuviera ya un IMU incorporado en la propia placa y que tuviera la posibilidad de comunicación Wifi para poderse comunicar con el robot. A continuación, se van a poner algunos dispositivos los cuales cumplirían esas condiciones:

- **Arduino uno wifi rev2:** Esta placa tiene un acelerómetro de 3 ejes, un giroscopio de 3 ejes. Además, cuenta con un módulo NINA-W102 Wi-Fi y Bluetooth de u-Blox [24].

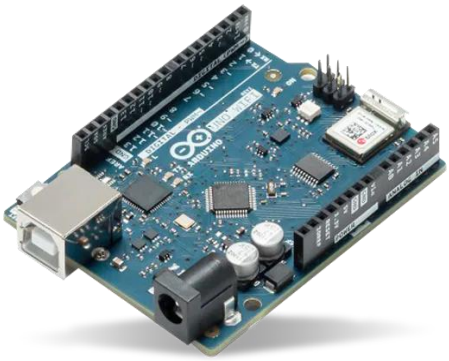


Ilustración 33. Arduino Uno Wifi Rev2. Fuente: Arduino

- **Arduino Nano 33 IoT:** Esta placa tiene un acelerómetro de 3 ejes, un giroscopio de 3 ejes y un módulo WiFi y Bluetooth integrados en el chip LSM6DS3 [25].
- **Arduino MKR WiFi 1010:** Esta placa no tiene acelerómetro, pero tiene un módulo Wifi y Bluetooth [26].
- **Arduino Nano RP2040 Connect:** Este modelo tiene un acelerómetro de 6 ejes, un giroscopio de 3 ejes, y tiene un módulo Wifi y Bluetooth integrados en el chip LSM6DSO [27].

A continuación, facilitaremos esta información en una tabla para hacerlo más visual con las características de los que tienen sensores IMU y la conexión Wifi que tiene incorporada.

Modelo	Acelerómetro	Giroscopio	Magnetómetro	IMU	Wifi y Bluetooth
Arduino Uno Wifi Rev2	3 ejes	3 ejes	No	LSM6DS3TR	NINA-W102
Arduino Nano 33 IoT	3 ejes	3 ejes	No	LSM6DS3	NINA-W102
Arduino Nano RP2040 Connect	3 ejes	3 ejes	No	LSM6DSOXTR	RP2040

Tabla 6. Comparativa de las características. Fuente: Elaboración propia

En cuanto a la capacidad de procesar la señal Wifi, son más o menos iguales. A la hora de hablar del sensor IMU, el Arduino MKR WiFi 101 queda descartado por no tiene, pero el resto se podría decir que tienen las mismas características. Al final se eligió el Arduino Uno Wifi Rev2, porque este dispositivo es uno de los que se pudieron facilitar sin la necesidad de tener que comprarlo a priori, aparte de esto tiene la posibilidad de poder acoplar una shild encima. Como desenlace de varias pruebas básicas se llegó a la conclusión que ninguno de estos IMU mencionados anteriormente tiene un magnetómetro, es por eso que se ha descartado usar el IMU de estas placas. Pero sí usar su capacidad de conexión wifi.

3.3. Dispositivos de detección inercial

Como ya se ha mencionado anteriormente, el IMU que tienen incorporados los Arduinos con comunicación wifi integrada, no son suficiente para este proyecto.

El sensor IMU que tiene la placa de Arduino que se ha seleccionado anteriormente no tiene magnetómetro y por ese motivo hay que buscar el IMU que pueda ser útil. A continuación, vamos a nombrar 3 sensores IMU distintos a los anteriores ya mencionados:

- El **MPU-6050**: [MPU-6000-Datasheet1.pdf (tdk.com)] es una IMU de 6 grados de libertad, fabricado por Invensense, que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Es uno de los IMUs más famosos por su gran empleabilidad y, gracias a su calidad precio. Se puede conectar a través de I2C o SPI.
- El **LSM6DS3TR-C**: [28] es un sensor IMU de 6 grados de libertad, fabricado por STMicroelectronics, que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Se puede conectar a través de I2C o SPI.
- El **BNO055**: [29] es un IMU de 9 grados de libertad, fabricado por Bosch, que combina un acelerómetro de 3 ejes, un giroscopio de 3 ejes y un magnetómetro de 3 ejes. A diferencia del resto de IMU's mencionados anteriormente, este sensor también proporciona información sobre la orientación absoluta del dispositivo en el espacio gracias al magnetómetro.

Una vez se sabe qué microchip cumple con las necesidades que hay, se ha realizado una búsqueda para seleccionar un dispositivo que sea compatible con la placa Arduino Uno Wifi Rev2 [30] y tenga el microchip BNO055. Con esas características se ha encontrado la placa arduino Shield 9 Axis Motion.

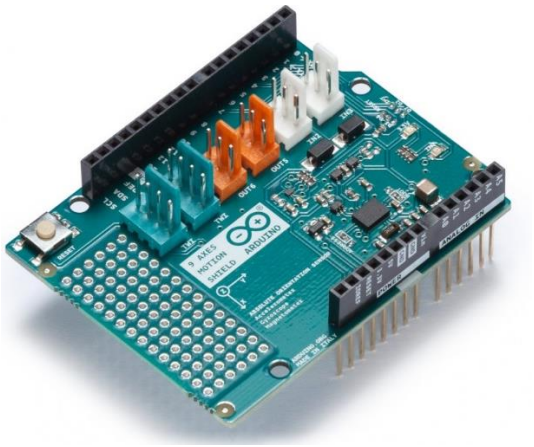


Ilustración 34. Arduino Shield 9 Axis Motion. Fuente: Arduino

4. PROGRAMA

4.1. Herramientas de programación

En este proyecto, para programar el microcontrolador y sus comunicaciones, se han usado dos Arduino uno wifi rev2. Cada uno de ellos contiene su propio Shield 9 axis motion, que es el que contiene el sensor IMU, el nombre del cual es BNO055.

Con toda la parte de hardware lista para obtener los datos del brazo humano y enviárselo al robot, hay que programarlo, para cumplir uno de nuestros objetivos: poder obtener la posición que tendría un brazo humano, ya sea en movimiento o en estático. Para ello, hay muchas herramientas, empezando por la que, cualquier persona en su momento de iniciación al mundo de Arduino, ha podido utilizar: el programa Arduino IDE. Pero para proyectos de estas dimensiones este programa nos limita a la hora de encontrar facilidades, identificar errores, modificar librerías, modificar datos internamente, etc. Es por eso que, para la programación de este proyecto se ha optado por utilizar Visual Studio Code. Este programa es un editor de código fuente gratuito y de código abierto, creado por Microsoft. Es compatible con múltiples lenguajes de programación y sistemas operativos, lo que lo hace muy versátil. Algunas de las características más destacadas de Visual Studio Code son la depuración, el control de versiones, la finalización automática de código y la personalización. Lo interesante de este programa, es que le puedes añadir extensiones, dependiendo del tipo de lenguaje o también ayudas para programar. En este caso, se ha usado para programar en Arduino, a través de PlatformIO. Aparte, también hay extensiones para identificar errores y muchas opciones más que hacen más fácil la programación. En resumen, es una herramienta muy útil para programadores que trabajan en proyectos complejos.

Otra herramienta utilizada en el proceso de programación ha sido GitHub. Esta herramienta es una plataforma de alojamiento de código fuente y control de versiones basada en la web. Lugar donde los desarrolladores almacenan y comparten sus códigos con otros programadores o consigo mismo para otros dispositivos. Además, tiene la posibilidad de colaborar en proyectos de software, como la gestión de problemas y la revisión de código. En este proyecto en concreto, se ha usado solamente para el almacenamiento de códigos generados para este, ya que, tiene dos cosas muy interesantes: la posibilidad de guardar la primera versión e ir subiendo mejoras al código máster y que se almacena el código con librerías y todos los archivos del código incluidos. Esta herramienta web también tiene la opción de sincronizarse con la aplicación Visual Studio Code, haciendo así más simple el proceso de guardar tu código en la nube.

A. Programas para simular

En la página de Universal Robot [31], nos ofrecen un simulador del robot, completamente gratis, solo se tiene que registrar como usuario. Este simulador, llamado URSim, es un software de simulación que se utiliza para la programación y simulación fuera de línea de programas de robots, también mencionado en el apartado" *2.1.3. Entorno de programación y simulación*". Como dicen ellos mismos, es un simulador que tiene algunas limitaciones, sin embargo, para pruebas previas antes de utilizar el robot físico o, simplemente, poder probar diferentes posibilidades, sin necesidad de estar con el robot físicamente, ya sirve.

Se pueden consultar todas las condiciones que necesita tu ordenador para poder llevar a cabo el proceso de instalación, así como el uso del simulador URSim, en el portal web [27]. Cómo este programa solo es compatible con el sistema operativo Linux y, para este proyecto, se ha hecho uso de un sistema operativo Windows, Universal Robot, ha creado una máquina virtual para los usuarios de UR. Esta máquina virtual contiene el simulador para UR3e, Ur5e y UR10e. Solo es posible ejecutar un simulador a la vez, en este caso, va a ser utilizado para el UR3e.

Una vez que se tiene la máquina virtual con el programa de simulación, se necesita un reproductor virtual para la máquina virtual de UR. Existen dos opciones más utilizadas [27]:

- **VMWare Player** (gratis para uso no comercial, pequeña tarifa para uso comercial): Comentar que la versión de VMWare requiere una actualización a

VM12.

- **VirtualBox** (programa gratuito): Esta opción no necesita de actualización, además de ser gratuito para todo tipo de aplicaciones. Por estos motivos ha sido el programa utilizado para la máquina virtual URsim.

  VirtualBox es un software de virtualización de código abierto que permite crear y ejecutar máquinas virtuales en el ordenador [27]. Una máquina virtual es un entorno de software que simula un ordenador, incluido el sistema operativo y el hardware. Con VirtualBox, los usuarios pueden instalar y ejecutar varios sistemas operativos en un solo ordenador. Además, este software es compatible con una amplia variedad de sistemas operativos, incluidos Windows, Linux, macOS y Solaris.

Los pasos de la instalación de la máquina virtual URsim y de VirtualBox, se pueden encontrar en el *Anexo XX URSim guía instalación.*

4.2. Librerías

En este proyecto se ha usado varios tipos de librerías y en este apartado se van a nombrar y explicar a grandes rasgos para que se han usado y las modificaciones que se les ha hecho a alguna de ellas. Ya que era necesario para el desarrollo del proyecto. Las librerías usadas son las siguiente:

- **Servo:** Esta librería para Arduino se utiliza para controlar servo motor. Estos motores tienen engranajes integrados y un eje que se puede controlar con precisión. Los servos estándar permiten que el eje se coloque en diferentes ángulos, generalmente entre 0 y 180 grados. La librería Servo permite controlar hasta 12 servos simultáneamente y hasta 48 servos dependiendo del Arduino a usar, pero en este proyecto solo se han llegado a controlar tres como mucho. Lo que hace en nuestro proyecto es cambiar los valores de grados en parámetros para que el motor se muevan en el ángulo deseado.
- **SoftwareSerial**: Esta librería hay que comentar que viene preinstalada en todos los proyectos que se hacen con la extensión PlatfromIO, eso quiere decir que ya está instalada, simplemente la tiene que llamar en el código. En caso de que te la instales en tu proyecto además de la que ya está preinstalada te va a dar errores la librería.

  Esta librería es la que permite la comunicación serie en otros pines digitales de

una placa Arduino, utilizando software para replicar la funcionalidad. Implementando nuevos pines RX y TX.

- **Arduino NineAxesMotio**: Esta librería oficial que recomendada Arduino para usar su shild [26], es con la que vamos a leer todos los datos del chip BNO055. Que principalmente son los del acelerómetro, giroscopio y magnetómetro.

  Aparte esta librería utiliza la librería Wire.h, para poder hacer la comunicación por I2C. La biblioteca Wire.h, es una biblioteca estándar de Arduino que proporciona funciones para la comunicación I2C. A demás de tener también internamente la librería del propio chip de Bosch que sería la librería BNO055.

  A esta librería se le han tenido que hacer algunas modificaciones en el .h y en el .cpp. Esto es debido a que la función con la que se le envía los valores al robot tiene que ser en radianes, como bien pone en *el "Anexo II Manual de funciones UR3e"* la función "servoj" que se le envía al robot. Eso implica que es mejor que los valores del sensor se lean directamente en radianes. Al mirar en la librería, se pudo ver que no existía ninguna manera de que se lean los movimientos pitch, roll y heading en radianes en vez de en grados, por lo tanto, se modificó la librería par que así fuera. En *Anexo VII Librerías y sus modificaciones* están explicadas todas las modificaciones que se le han añadido a la librería para que fuera así. Dicho esto, hay que comentar que si no se usa la librería modificada no funcionará correctamente el código.

- **WifiNINA**: Es una librería de Arduino que permite la conexión a redes inalámbricas utilizando el módulo NINA W101.Esta librería le permite utilizar las capacidades Wifi de Arduino Uno Wifi Rev2, Arduino Nano 33 IoT, Arduino MKR 1010 y Arduino MKR VIDOR 4000 WiFi. Además de poder instalar servidores, clientes y enviar/recibir paquetes UDP y TCP/IP a través de Wifi.

  Esta librería es compatible con todas las arquitecturas, por lo que debería poder usarla en todas las placas Arduino. La biblioteca es muy similar a las bibliotecas Ethernet y Wifi, y muchas de las llamadas de función son las mismas. La librería WifiNINA es la recomendada por Arduino para usar el Arduino Uno Wifi Rev2 [32].

  La librería WifiNINA, utiliza la librería SPI.h para la comunicación SPI. La biblioteca SPI.h es una biblioteca estándar de Arduino que proporciona funciones para la comunicación SPI [33].

4.3. Comunicaciones entre el dispositivo y el controlador del Robot

En este apartado hay que tener claro que los dispositivos que se usan en este proyecto tienen que comunicarse de alguna manera entre ellos para recibir y transmitir los datos al robot UR3 e.

En un primer instante, solo se tenía que comunicar la Shild 9 axis motion, que contiene el IMU BNO055, con el Arduino uno wifi rev2 y este con el robot UR3 e. Para poder elegir qué tipo de comunicación, se seguirá el siguiente criterio. Uno de los objetivos es evitar comunicaciones físicas entre los dispositivos usados, evitando así dificultades motrices. Pero se realizará la que se valore mejor, teniendo en cuenta la velocidad de transmisión de datos, es decir, que se intentará evitar comunicación física siempre y cuando la trasmisión de datos sea rápida.

El siguiente paso es saber qué tipo de comunicaciones se pueden usar con cada dispositivo y, qué dispositivos se tiene que comunicar entre ellos para ver su compatibilidad. En primer lugar, se va a comenzar por el sensor que está insertado en la Shield 9 axis motion. Este sensor se tiene que conectar físicamente encima de Arduino uno wifi rev2, y su comunicación es por I2C [26]. Comentar que no había otra opción de comunicación, pero este tipo de comunicación física no provocará dificultades para la movilidad, por lo tanto, cumplimos con uno de nuestros objetivos. La siguiente comunicación que tendría que haber entre dispositivos es la del Arduino Uno wifi rev2 con la del robot UR3 e.

Mediante el enlace [23] y en el manual del UR3e en el *"Anexo VIII UR3e Manual"* podemos ver que hay varios protocolos de comunicación del UR3 e. Los métodos empleados pueden ser:

- **TCP/IP**: Es una conexión por sockets a través de Ethernet que permite que dos programas intercambien cualquier flujo de datos.
- **ProfiNet**: Es un protocolo de comunicación industrial que se utiliza para la automatización de procesos.
- **EtherNetIP**: Es un protocolo de comunicación industrial que se utiliza para la automatización de procesos.
- **XMLRPC**: Es un protocolo que se utiliza para enviar mensajes entre sistemas informáticos a través de Internet.
- **MODBUS**: Es un protocolo que se utiliza para la comunicación entre dispositivos electrónicos.

También hay que investigar qué tipo de comunicaciones son posibles con el Arduino Uno

WiFi Rev2. A partir de la información encontrada en el enlace [24] y en el datasheet del NINA-W102, está en el *"Anexo III Esquemas y datasheets de los dispositivos"*, podemos ver que hay varios protocolos de comunicación:

- **WiFi**: Este Arduino tiene un módulo NINA-W102 de u-blox que proporciona conectividad WiFi. Con el que se pueden usar los protocolos de TCP/IP, UDP, WebSocket y MQTT
- **Bluetooth**: El Arduino con el módulo NINA-W102 también proporciona conectividad por Bluetooth.
- **USB**: El Arduino, se puede conectar a un ordenador mediante USB para programación y depuración, gracias al microcontrolador, ATmega4809 de 8 bits.
- **I2C**: El Arduino tiene dos puertos I2C, que se pueden utilizar para comunicarse con los dispositivos externos, gracias al microcontrolador, ATmega4809 de 8 bits.
- **Serial**: El Arduino, tiene un puerto serie que se puede utilizar para comunicarse con otros dispositivos, gracias al UART que está integrado en el microcontrolador, ATmega4809 de 8 bits.

Una vez sabemos los distintos tipos de comunicación que tienen cada uno. Podemos ver que la única comunicación factible entre estos dos dispositivos tendría que ser por TCP/IP. De esta manera se cumple el objetivo de evitar hacer comunicaciones físicas entre dispositivos.

Para la comunicación por TCP/IP, se necesita crear una conexión como cliente con el servidor del robot, además de tener que usar también la dirección IP de dicho robot, ya que, cuando el Arduino se conecte al servidor deberá detectar a quién le tiene que comunicar, ya que, es posible que pudiera haber más dispositivos en ese servidor. Es muy importante conectar los dos dispositivos al router. Hacer mención que el router estará conectado físicamente por cable Ethernet al robot para poder realizar esta comunicación entre dispositivos. Al final se ha conectado un router privado, es decir, independiente a la red de la universidad, para que enrote más rápido, además de que no tenga puertos ocupados para otras tareas, haciendo así que la conexión sea más fluida entre los dispositivos.

Esta primera explicación de las comunicaciones entre estos dispositivos es un primer intento para conseguir el objetivo principal que es que el robot imite le brazo humano, pero realizándolo con un solo IMU. Al ser imposible realizarlo con un solo sensor, se hizo otro planteamiento en el cual entra en juego otro sensor más. Por ese motivo, ahora

se tiene que plantear una nueva comunicación.

Como se ha hecho anteriormente, ahora se van a plantear los diferentes tipos de protocolos de comunicación posibles entre los dos sensores con el Arduino, para poder enviar al robot una trama de datos, con todos estos juntos, además que estos lleguen con un orden coherente con ambos sensores. Para la comunicación de este escenario, se ha preferido usar otro Arduino Uno Wifi Rev2 para poder usar un protocolo de comunicación inalámbrico. Hacer mención que uno de los dos Arduino estará comunicado por TCP con el robot, ya que, como se ha comentado anteriormente, es la única comunicación factible. Ahora falta definir la comunicación entre los dos Arduino. Para ello se describe a continuación los tipos de comunicación estudiadas hasta llegar a la definitiva:

- **Bluetooth**: La comunicación por bluetooth se ha descartado porque, aunque los dos dispositivos están cerca, es mucho más lenta que la comunicación wifia, aunque es más segura. Pero, como se ha mencionado antes, el orden de prioridades es la velocidad de transferencia de datos y después la comunicación inalámbrica.
- **TCP**: Este protocolo fue el primero en probarse para comunicar ambos Arduino. Pero, al añadir la segunda comunicación por TCP con el robot, solo hacía el primer ciclo completo del código una sola vez. Se tendría que indagar más en la librería utilizada para descartar si la librería utiliza timers compartidos o es cosa del microcontrolador que no puede gestionar dos comunicaciones TCP, pero una como server y otra como client. En el *"Anexo IV Código"* se puede observar el código que se ha mencionado.
- **UDP**: Este protocolo fue el siguiente en utilizarse, ya que era una comunicación más rápida aun no siendo tan segura con la recepción de todos los datos. Al realizar la comunicación por UDP entre los Arduino, a la vez que la del robot por TCP se observa algún problema, puesto que después de varias pruebas seguía funcionando la comunicación entre los dispositivos, pero tardaba entre 3 y 6 segundos en recibir los datos de uno al otro. Se sospecha que es por el mismo tema que con el TCP, pero hay que seguir indagando. En el *"Anexo IV Código"* se puede observar el código que se ha generado para este tipo de comunicación.

  Entonces, viendo que el error pasaría igual en el resto de los protocolos, hemos pasado al siguiente grupo de comunicaciones, que serían las que necesitas conexión física entre los dispositivos.
- **I2C:** para hacer uso de este protocolo, en primer lugar, hay que cambiar la

dirección física de la shild 9 Axsis motion. Para ello hay que acceder a los esquemas de la placa, que está en el *"Anexo III Esquemas y datasheets de los dispositivos"* y se podrá ver lo que hay en la Figura35 donde se puede identificar la dirección dependiendo del valor de la entrada. Si en la entrada COM3/I2C_ADDR le entran 3,3 V será la dirección 0x29 y si tiene el puente en el GND le entraran 0V y la dirección es 0x28. Como dato interesante, es necesario comentar que la dirección predeterminada de la shield es 0x28, pero la del sensor BNO055 solamente sería la dirección por defecto 0x29. Esto se puede encontrar en el datasheet del sensor en el *"Anexo III Esquemas y datasheets de los dispositivos*".

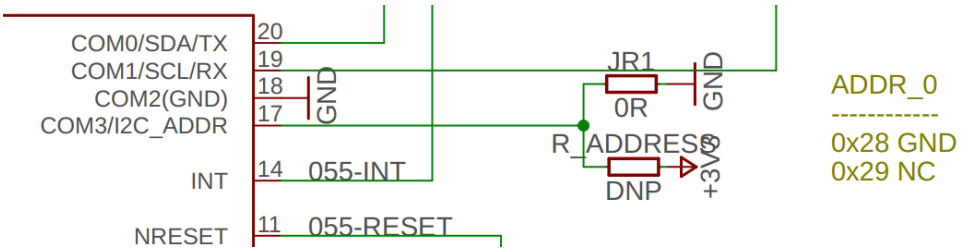


Ilustración 35. Direccion fisica de shild 9 Axsis motion. Fuente: Arduino

Una vez cambiada la dirección física, hay que tener en cuenta la conexión física entre las dos Shilds y el Arduino. Para ello se tiene que colocar una de las dos shild encima del Arduino, creando así la conexión física de una de ellas. La shild restante tiene que conectar los pines SDA y SCL con los pines SDA y SCL del Arduino, respectivamente. Después se debe conectar el pin GND de la Shield al pin GND del Arduino además del pin de 5 voltios del Arduino al de la shild parar alimentarla.

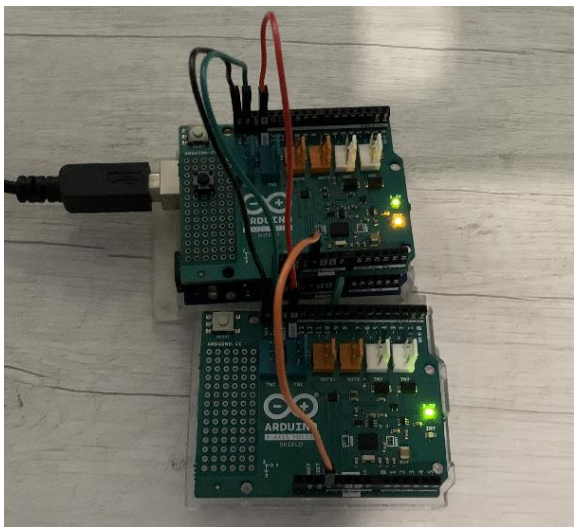


Ilustración 36. Conexión física I2C. Fuente: Elaboración propia

Una vez hecho todo esto, se han probado varios códigos con varios cambios hechos. Lo único que se ha conseguido es que lea datos de uno de los sensores. Se ha probado el código individualmente y ha funcionado con las dos shields, pero con un mismo código ha sido imposible. Es verdad que esta comunicación, en teoría, es capaz de abarcar más de una a la vez, pero no se ha podido saber qué es lo que estaba incorrecto. El código está en el *"Anexo IV Código"*.

- **Serial:** Esta opción ha sido la última posible. Primero de todo hay que tener en cuenta que ahora, cada vez que se inicia el sensor en el código, hay que poner la dirección física que le corresponde a cada sensor. Dependiendo si es la 0x28 o la 0x29. Después, hay que conectar cada una de las shild a un Arduino distinto. También hay que tener en cuenta que hay una parte de conexión física entre dispositivos. Se tiene que conectar el pin Tx de un Arduino con el Rx del otro, y lo mismo con Rx de uno y el Tx del otro. Además de conectar sus GND, también hay que alimentar a ambos Arduino.

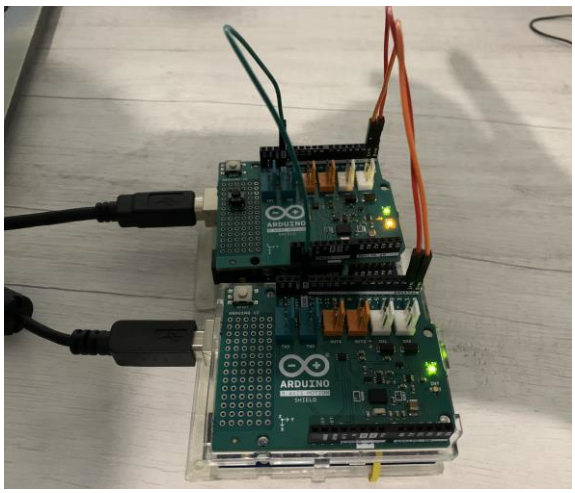


Ilustración 37. Conexión física puerto serie. Fuente: Elaboración propia

Para esta comunicación hubo algunos problemas y no se encontraba qué estaba mal. Así que sé probo con una librería "SoftwareSerial" que crea una comunicación serial virtual. Con este método si funcionaba, pero es bastante limitada en cuanto a la velocidad Baud rate ya que No se podía superar más de 9600. El código está en el *"Anexo IV Código"*.

Por esto, se ha proseguido con la comunicación puerto serial convencional. En el *"Anexo XI Comprobación de la comunicación serie"* están los pasos que se han hecho para comprobar la comunicación serie. Se pueden observar pruebas hechas con el osciloscopio para intentar leer el dato que tendría que estar saliendo del

Arduino por el puerto serie. Al ver que no salía ninguna señal, pero que el mismo código y las mismas conexiones, con dos Arduinos Uno, si funcionaba, se miró el datasheet más detalladamente de la placa Arduino Uno Wifi Rev2. Se encontró que para comunicarse por el puerto serie, con el monitor seriel del pc, hay que utilizar "Seria1." y, que para enviar o recibir datos por el serial, hay que usar el "Serial1.".

Una vez averiguado esto, se procedió a crear un protocolo de comunicación por el puerto serial, entre los dos Arduino para que siguieran los pasos y se comunicasen los dos Arduino por orden. Este método finalmente ha sido efectivo para cumplir con el objetivo final de que el robot imite el brazo humano. El código de la comunicación serie está en el *"Anexo IV Código"*.

Se ha creado un protocolo de comunicación muy simple mediante el puerto serie. La explicación de este protocolo se puede ver en *el "Anexo XII Protocolo de comunicación "*.

- **SPI**: Este microcontrolador tiene un puerto de comunicación SPI. Este protocolo de comunicación es el utilizado por el microcontrolador para comunicarse con el chip wifi NINA-W102. Por este motivo no se ha utilizado con otros fines.

Por último, es necesario explicar la configuración del router que se va a usar. Para mayor comodidad, se ha modificado el nombre y la contraseña de serie. Además, se ha configurado el router para que ponga la misma dirección IP al robot UR3 e, y así no tener que estar cambiado la IP asignada en el código que se usa para identificar el robot cada vez. Dicha configuración se especificará en el *Anexo IX Configuración del Router*.

Ilustración 38. Router Xiaomi. Fuente: Elaboración propia

4.4. Obtención de datos del sensor

En este proyecto se ha utilizado el IMU BNO055, que está incorporado dentro del Arduino shield 9 axis motion. El chip BNO055, es un sensor con diferentes mediciones, las cuales, están definidas en el *"Anexo III Esquemas y datasheets de los dispositivos"*. Las más interesantes para este proyecto son, el Magnetómetro, Acelerómetro y el Giroscopio.

En un primer lugar, para usar estos tres sensores debe tener en cuenta que se pueden calibrar, más concretamente, se pueden auto calibrar, como lo exponen el datasheet del *"Anexo III Esquemas y datasheets de los dispositivos".*

A continuación, se explicará el proceso de calibración llevado a cabo. Comentar que cada sensor tiene unos pasos específicos a seguir. Para ello, en primer lugar, es necesario saber en qué nivel de calibración se encuentra el elemento. Siendo 0 nada y 3 totalmente calibrado (solo pueden ser números enteros). Este dato se puede obtener mediante las funciones: "uint8_t readAccelCalibStatus(void);" "uint8_t readGyroCalibStatus(void);" "uint8_t readMagCalibStatus(void);" están en *Anexo VII Librerías y sus modificaciones.*

Para la calibración del acelerómetro, según en el *Anexo III Esquemas y datasheets de los dispositivos,* este se debe colocar en 6 posiciones estables diferentes durante un período de pocos segundos, asegurándose de que existe un movimiento lento entre 2 posiciones estables. Las 6 posiciones estables podrían estar en cualquier dirección. Es decir, lo que se debe hacer es colocar el sensor sobre una superficie plana, cada una de sus caras durante un par de segundos por cada una de ellas. Se puede saber si el proceso se está realizando correctamente si por ejemplo en la función "Serial.print(mySensor.readAccelCalibStatus());" aparece un valor igual a 1 (raramente se llegará a ver un 2). La primera posición en la que aparezca el valor 1 es a la que se tendrá que volver cuando vuelva a salir un 1, eso provocará que el sensor esté calibrado y el valor que salga sea directamente 3.

Para la calibración del giroscopio, según en el *Anexo III Esquemas y datasheets de los dispositivos,* se tiene que colocar el dispositivo en una única posición estable durante un período de pocos segundos para permitir mediante la función "Serial.print(mySensor.readGyroCalibStatus());", se obtenga como resultado un valor igual a 3, confirmando así su correcta calibración.

Para la calibración del magnetómetro, según en *Anexo III Esquemas y datasheets de los dispositivos*, primordialmente hay que evitar la cercanía tanto del hierro duro como del hierro blando, ya que, son susceptibles a las distorsiones de los campos magnéticos externos y, por tanto, para garantizar la precisión adecuada, hay que tener en cuenta que no puede haber ningún dispositivo electrónico cerca ni imanes. Ahora, sabiendo eso, el siguiente paso será hacer movimientos aleatorios, por ejemplo, escribir el número '8' en el aire, haciendo así que con la función "Serial.print(mySensor.readSystemCalibStatus());", aparezca un 3, confirmando que está calibrado.

4.5. Metodología

A continuación, se explicará la metodología seguida para la ejecución del movimiento del brazo robot. Un punto importante a tener en cuenta en este proceso es la implementación de un pulsador para controlar la transición entre el proceso de calibración y la toma de datos del sensor. Por esa razón, se ha instalado un botón en la Shield para que, una vez el sensor esté calibrado, se tenga el tiempo suficiente para colocarlo en el brazo humano, y así poder empezar con toda la toma de datos en la posición correcta del sensor y, poder evitar problemas.

Para implementar este botón, en primer lugar, se ha decidido qué tipo de flanco se iba a querer detectar, si el flanco negativo o el positivo. En este caso se ha elegido el flanco negativo, ya que, al despulsar, es posible que se provoquen menos movimientos al empezar la lectura en los sensores. Para el caso del flaco positivo, al tener que pulsar encima del sensor, puede empezar la lectura erróneamente.

A continuación, se hicieron varias pruebas, empezando por un pulsador con una resistencia externa y un led, todo esto conectado al Arduino. Al final se acabó usando la resistencia pull-up interna del Arduino para el circuito del pulsador. Después se investigó y se descubrió que, la placa Arduino Uno wifi rev2, tiene un led interno que es el pin 25 con el que podríamos comprobar si se detectaba el flanco negativo, evitando así, más elementos, como es el led y la resistencia. De esta manera, solo se tendría que añadir a la shield un pulsador, el cual se conectará al pin12 y al GND. El código está en el *Anexo IV Código.*

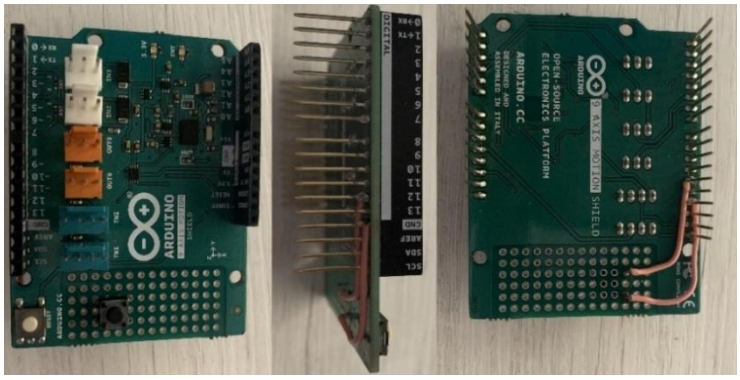


Ilustración 39. Conexión pulsadora para interno Arduino Shield 9 Axis Motion. Fuente: Elaboración propia

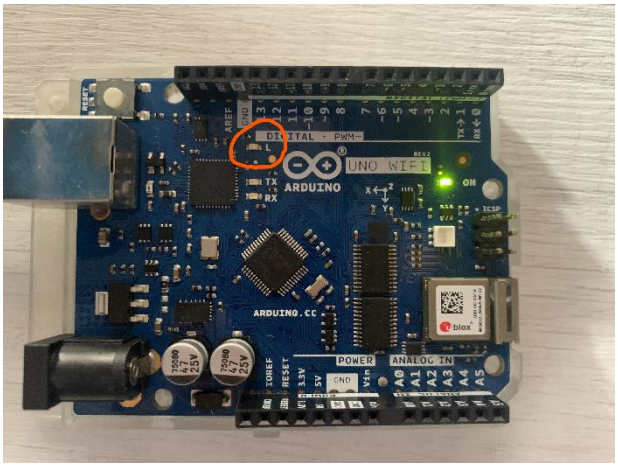


Ilustración 40. Led interno Arduino Uno Wifi Rev2. Fuente: Elaboración propia

4.5.1. Primer método

En un primer momento, cuando se empezó a investigar sobre la realización del objetivo final, conseguir que el robot imite al humano, se encontró una empresa del tipo industrial llamada "Nordbo Robotics" que tiene un dispositivo semejante a un lápiz el cual se comporta de manera muy similar a lo que se quería obtener con el brazo robot. En concreto, esta empresa lo usa para grabar los movimientos que, posteriormente, el robot tendrá que hacer, siguiendo el mismo recorrido [34] [35].

A partir de este concepto inicial, se intentó hacer la lectura de la posición del brazo humano con un solo IMU, utilizando principalmente los sensores del acelerómetro y el magnetómetro. La idea era que, con el acelerómetro, se obtendría la lectura de la aceleración del dispositivo en todos los ejes.

Con esta información, la teoría nos dice que se puede calcular la velocidad y, a su vez, con la velocidad, se puede saber la distancia que ha recorrido el dispositivo. Por último, con el magnetómetro sabremos donde está el norte y, por lo tanto, saber la orientación de la mano.

Con todos estos datos correctos, es decir, una vez se conozca la distancia recorrida en los tres ejes y la orientación que tiene el dispositivo, se tiene que introducir la siguiente función "movel (pose, a=1.2, v=0.25, t=0, r=0)" que está en el *Anexo II Manual de funciones UR3e*. Esta función, al enviársela al robot, calculará internamente los motores que tendrá que activar para llegar a esas distancias y orientación marcadas por la lectura del dispositivo. Provocando que el robot recorra las mismas distancias en los tres ejes y con la misma orientación que el brazo humano. Hay que puntualizar que, con este método, el brazo robot no imitaría las posturas del brazo humano, sino que solamente el TCP del robot recorrería la distancia que haría el brazo humano con la correspondiente orientación. Es decir, el robot recibiría una distancia y una orientación, con lo que este robot, internamente, elegiría qué motores mover para llegar a ese punto con esa orientación.
Este método no ha funcionado correctamente. Se ha deducido que es por dos motivos: la precisión de los datos del acelerómetro y la acumulación de ese error con respecto del tiempo de toma de datos. El problema está en pasar los valores del acelerómetro a distancia recorrida en las condiciones que hay que hacerlo y la precisión de dichos valores.

A continuación, se explica con más detalle todo el proceso que se ha seguido para intentar llegar de unos valores de aceleración a unos de distancia recorrida.
En primer lugar, hay que ser consciente de que el acelerómetro solo lee el valor cada cierto periodo de tiempo. Entre periodo y periodo, el microcontrolador tiene que estar haciendo otras acciones. Partiendo de esta premisa, se ha tenido que hacer una hipótesis. Esta hipótesis es que, de lectura de datos a lectura de datos, se dice que esa aceleración es una aceleración constante. Haciendo esta hipótesis hay que tener mucho cuidado con el tiempo de ciclo del código, es decir, cuanto tiempo pasa entre lectura y lectura. Porque esta hipótesis realmente puede ser verdad o no serlo. Pero mientras menos tiempo pase entre lectura y lectura, más preciso será. La siguiente gráfica es un ejemplo de una conversión de señal analógica a una señal digital, donde la señal analógica seria la aceleración y la señal digital sería los datos que envía el sensor.

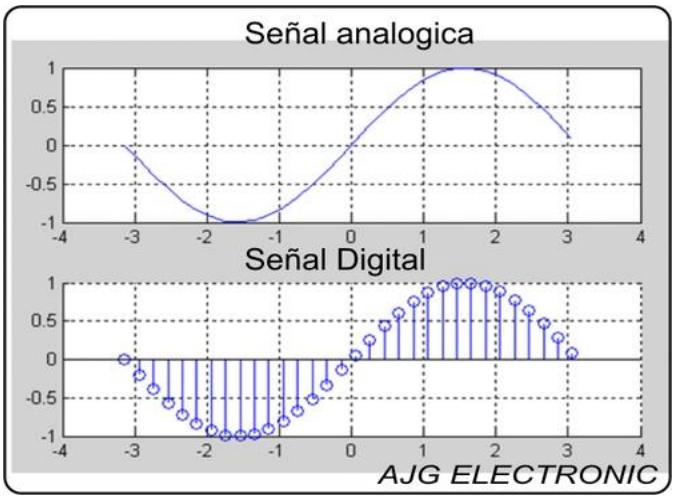


Ilustración 41. Ejemplo para entender la hipótesis mencionada. Fuente: entendiendo los convertidores ad/da

Una vez hecha esta hipótesis y haber definido la aceleración entre punto y punto de lectura, podemos utilizar las fórmulas de la aceleración constante. La fórmula vectorial de la velocidad para un movimiento con una aceleración constante es:

$$\vec{v}=(\vec{v0})+\vec{a}\cdot t \qquad \textbf{[Eq. 6]}$$

Ahora, si esta fórmula la desglosamos para los tres ejes de coordenadas, de los que se va a tener que calcular su velocidad, será la misma para los tres ejes, pero simplemente cambiando el valor de la aceleración correspondiente al de cada uno de ellos:

$$v=v0+a\cdot t \qquad \textbf{[Eq. 7]}$$

v: Es la velocidad en el eje en ese momento dado.

v0: Es la velocidad inicial en ese eje, es decir, la velocidad anterior.

a: Es la aceleración constante media en ese eje.

t: Es el tiempo transcurrido desde el inicio del movimiento.

La fórmula vectorial de la distancia recorrida con una aceleración constante sería:

$$\vec{d}=(\vec{d0})+(\vec{v0})\cdot t+1/2\cdot\vec{a}\cdot t^2 \qquad \textbf{[Eq. 8]}$$

Ahora, si esta fórmula la desglosamos para los tres ejes de coordenadas, de los que se van a tener que calcular su distancia, será la misma parar los tres ejes, pero simplemente cambiando el valor de la aceleración y de la velocidad correspondiente al de cada uno:

$$d=d0+v0\cdot t+12\cdot a\cdot t2 \qquad \textbf{[Eq. 9]}$$

d: Es la distancia recorrida en ese eje en concreto.

d0: Es la posición inicial en ese eje, es decir, la posición anterior.

v0: Es la velocidad inicial en ese eje, es decir, la velocidad anterior.

a: Es la aceleración constante media en ese eje.

t: Es el tiempo transcurrido desde el inicio del movimiento.

Se pueden encontrar todas las fórmulas de forma más detallada en el *Anexo VI Teoría de Cinemática.*

Una vez se han definido las fórmulas físicas que se tienen que utilizar para intentar usar este método, el siguiente paso es generar el flujograma para, posteriormente, generar el código. Este flujograma, al ocupar mucho, está disponible en el *Anexo V Flujogramas*, sería el Flujograma 1.

Antes de generar el código directamente y hacer que el UR3e reciba datos erróneos y haga movimientos donde colisione, se ha hecho el código para comprobar si todos los valores tienen sentido. Este código se puede ver en el *Anexo IV Código*, además del flujograma, que es en el *Anexo V Flujogramas*, sería el Flujograma 2.

A continuación, se pasa a explicar los problemas que han aparecido con este método y, por lo tanto, por qué no ha funcionado correctamente. En un primer momento, se ha podido ver que los valores del acelerómetro ya detectaban valores estando quietos. Por ese motivo, se hace la parte de filtro de datos, proceso que consiste en eliminar esos valores.

```
void FiltodeDatos() {   //Hacer un filto para evitar falsas lecturas

// Verificar si el valor de accelerationX está dentro del rango -0.20 y 0.20
if (accelerationX >= -0.20 && accelerationX <= 0.20) {
  accelerationX = 0.0;  // Establecer el valor a 0 si está entre -0.20 y 0.20
} else {  // Si el valor de accelerationX está fuera de entre -0.20 o 0.20, ajusta su valor en función de su signo
  accelerationX = (accelerationX > 0.0) ? accelerationX - 0.20 : accelerationX + 0.20;
}
```

Ilustración 42. Filtro de toma de datos en estático. Fuente: Elaboración propia

En la Ilustración 42 solo se ha enseñado el eje x porque primero de todo nos centramos en uno de los tres ejes, aunque el eje y se utiliza es el mismo código.

En siguiente lugar, se ha deducido que, probablemente esos valores que considerábamos residuales, son provenientes de la fuerza de la gravedad que, en el estado inicial, la tendría que leer completamente el acelerómetro en el eje z pero, al no estar completamente horizontal, el resto de acelerómetros detectaban esas leves aceleraciones. Igualmente, al hacer el filtro y solo centrarnos en el eje x no tendría que haber ningún problema, ya que,

la inclinación sería la misma al desplazar el sensor un metro, capturar dato y volver, tendría que haber dado un valor cercano a 0.

Por otro lado, al utilizar las fórmulas físicas anteriormente comentadas, se comprueba como la velocidad anterior no vuelve a valer 0. Una vez que está parado el sensor, eso provoca lecturas muy erróneas, ya que, los valores de velocidad y de distancia nunca disminuirán solo se mantienen o aumentan en caso de desplazarse. Por ese motivo, en el código se puso que, si el acelerómetro, durante 2 ciclos seguidos su valor es de 0, la velocidad anterior pasaba a valer 0, intentando así arreglar esa incongruencia que está pasando. Hay sospechas de que al tomar datos del acelerómetro cada x tiempo, no detecta la desaceleración correctamente, provocando así que la velocidad anterior no sea contrarrestada por la aceleración negativa multiplicada por el tiempo.

```
void CalculosVelocidad(){
///Calculo de la veloccidad con los valores del accelerometro

 TiempoDatos= TiempoActual/1000.0;

 //Calcular velocidad con acceleraccion costante
 vx=v0x+(accelerationX*TiempoDatos);
 vy=v0y+(accelerationY*TiempoDatos);
 vz=v0z+(accelerationZ*TiempoDatos);

 //Guardar valor anterior o poner a velocidad anterior 0 si la velocidad lleva dos ciclos seguidos 0
 if (accelerationX == 0) {
   contx++; //Contador para detectar los 0 del acelerometro X

 } else {
   // Si el valor actual no es 0 o no es consecutivo con el anterior, reiniciar el contador de ceros
   contx = 0;
 }
 if (contx >= 2) { // Si detecta dos 0 segidos detecta que esta parado y borrar la velocidad anterior
   v0x = 0;
 } else {
   v0x=vx;  //si no pasa lo de los dos 0 seguidos el valor aterior es igual al actual para poder hacer el caluclo de la velocidad con acceleraccion constante
 }
```

Ilustración 43. Filtro para la velocidad anterior dentro de función cálculo de la velocidad. Fuente: Elaboración propia

Otro aspecto muy importante a tener en cuenta fue el tiempo de muestreo, para hacer los cálculos de las velocidades y de las distancias recorridas. Para hacer los cálculos, lo que se necesita, es saber cuánto tiempo tarda, de la lectura anterior a la actual. Se ha puesto un medidor de tiempo antes de tomar el dato y, haciendo el resto con el tiempo anterior, se obtiene el tiempo que ha pasado entre la toma de datos anterior y la actual.

Después de toda esta explicación se puede ver que hay problemas con la lectura de datos y que por ello los cálculos generados no son totalmente lo realistas que se necesitan. Al ver que no funcionaba correctamente y que las soluciones a los problemas no eran del todo óptimas, se volvió al plan que se había planteado en un primer momento.

4.5.2. Segundo método

En un primer momento, ya se optó por utilizar las lecturas de los ángulos de Euler que son el Roll, Pitch y Heading/Yaw, pero se tendrá que utilizar dos sensores en vez de uno. Estos ángulos de Euler son una forma común de describir la orientación tridimensional

de un objeto o sistema en el espacio. Principalmente, estos ángulos se utilizan en campos como la navegación, aeronáutica, robótica y la animación por ordenador, para representar el movimiento. A continuación, se explicará cada uno de los tres movimientos que hay:

- **Roll**: también llamado ángulo de balance o ángulo de inclinación, al medir los ángulos respecto del eje x del dispositivo. Esto quiere decir que es el ángulo en el que un objeto o sistema gira alrededor de su eje longitudinal, este es la línea que atraviesa el objeto de extremo a extremo. Ejemplo más visual sería, como si fuera, una línea que va de la cabeza a los pies de una persona tumbada boca arriba. Esto hace que este ángulo mide cuánto se inclina el objeto hacia la izquierda o hacia la derecha con respecto a su posición horizontal.
- **Pitch**: también llamado ángulo de cabeceo o ángulo de elevación, al medir los ángulos respecto del eje y, del dispositivo, es decir, es el ángulo en el que un objeto o sistema gira alrededor de su eje lateral, que es la línea que atraviesa el objeto de lado a lado. Un ejemplo más visual sería, como si fuera, una línea que va de una oreja a la otra oreja en una persona tumbada boca arriba. Esto hace que mida cuanto se inclina el objeto hacia arriba o hacia abajo con respecto a su posición horizontal.
- **Heading o Yaw**: también llamado ángulo de guiñada, al medir los ángulos respecto del eje z, del dispositivo. Esto quiere decir que el ángulo en el que un objeto o sistema gira alrededor de su eje vertical, está en la línea que apunta hacia arriba desde el objeto. Un ejemplo más visual sería, como si fuera, una línea desde la parte superior de la cabeza hasta los pies de una persona. Esto hace que mida la dirección en la que el objeto está apuntando en el plano horizontal, como una brújula apuntando hacia el norte, sur, este u oeste.

Con los tres ángulos de Euler (Pitch, Roll y el Heading/Yaw) ya definidos, se puede determinar los movimientos en todas direcciones, proporcionando una orientación tridimensional de un objeto o sistema. El código final que determina la lectura de movimientos y orientación tridimensional del brazo humano que serán imitados por el brazo robot, está basado en este tipo de ángulos.

El primer contacto con estas tres lecturas de ángulos se hizo conectando físicamente a unos servomotores paso a paso, para poder ver bien si el movimiento que se generaba en el sensor era lo que se movían los motores paso a paso. El código de esta prueba está en el anexo *Anexo IV Código*.

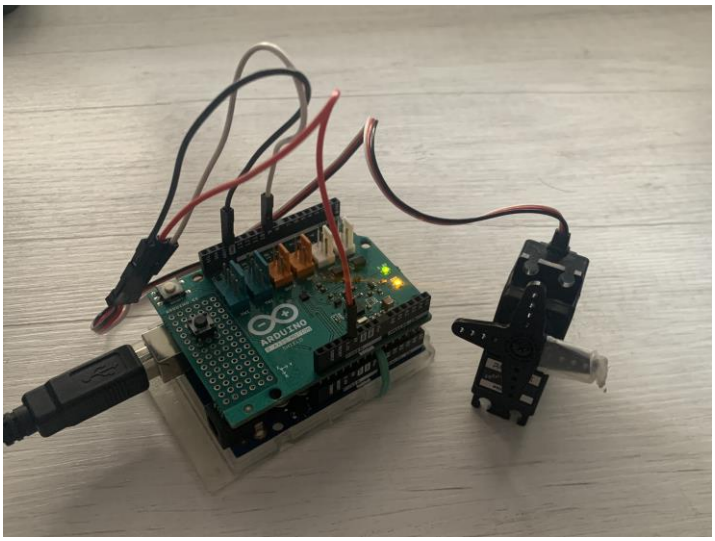


Ilustración 44. Filtro para la velocidad anterior dentro de función cálculo de la velocidad. Fuente: Elaboración propia

La conexión del servo con el Arduino es el cable rojo a 5 v del Arduino, el cable negro al GND del Arduino y el cable blanco es el pin donde le envías los ángulos que previamente se extraen con Euler roll o pitch o heading. Hay que tener en cuenta que el movimiento máximo de estos motores es de 180º como mucho. Como se puede ver en la imagen anterior, se ha puesto una pegatina en una de las aspas para poder ver más fácil el movimiento.

En un primer momento se consiguió hacer el control de los motores aplicándole los ángulos leídos por el sensor. Pero a partir de cierto punto de giro se podía ver que el motor hacia movimientos incoherentes con el movimiento del sensor. Eso podía ser por el límite del motor y por el rango del sensor. Por ese motivo se, fue a investigar el rango de ángulos en el que se está dando cada uno de los tres movimientos.

A. Rangos de ángulos del sensor

Una vez ya se conocen las funciones de como determinar los grados de movimiento de los tres ejes y conseguir uno de los objetivos (detectar todos los movimientos del brazo humano), se tiene que investigar que rango de movimiento en ángulos son capaces de leer. Para ello, en un primer momento, se hizo una toma de datos girando en los tres ejes hacia el mismo sentido, para ir tomando los datos y poder determinar su rango en cada eje.

En primer lugar, se cargó el código que está en el *Anexo IV Código* en el microcontrolador, para controlar la lectura del BNO055, es decir, el sensor. Entonces, una vez cargado el código por el monitor serie, se pueden observar por pantalla los ángulos

de los tres ejes. En un primer momento, se parte de un estado inicial con el sensor totalmente horizontal encima de la mesa. En este punto se hace la toma de los tres ángulos en cada uno de sus ejes en el estado inicial.

Una vez partido de este inicio, se escoge uno de los ejes y se procede a ir girando a sí mismo del propio eje, unos 90º con respecto de la mesa (es decir, del estado inicial) y se va a ir haciendo toma de datos. Este proceso se repite hasta dar la vuelta entera al sensor y con el resto de ejes. Una vez realizado la toma de datos de los tres ejes no se puede deducir el rango correctamente del Roll y, por este motivo, se tiene que repetir el proceso, pero en lugar de desplazarlo cada 90º se hace cada 45º.

A continuación, se muestran una serie de tablas resumen del proceso realizado con los tres ejes.

Pitch, EjeX

Giro del IMU	0	1/4	1/2	3/4	1
Datos leídos	2.31º	87.19º	-179.69º	-90.06º	0.19º
Valor aproximado	0º	90º	-180	-90º	0º

Tabla 7. Rango de ángulos del Pitch. Fuente: Elaboración propia

Roll, Eje Y

Giro del IMU	0	1/8	1/4	3/8	1/2	5/8	3/4	7/8	1
Datos leídos	0.69º	44.19º	88.56º	44.50º	1.37º	-40.13º	-87.25º	-44.69º	0.06º
Valor aproximado	0º	45º	90º	45º	0º	-45º	-90º	-44.6-9º	0º

Tabla 8. Rango de ángulos del Roll. Fuente: Elaboración propia

Heading, Eje Z

Giro del IMU	0	1/4	1/2	3/4	1
Datos leídos	183.00º	273.44º	9.06º	100.12º	189.88º
Valor aproximado	180º	270º	0º	90º	180º

Tabla 9. Rango de ángulos del Pitch, Roll y Heading. Fuente: Elaboración propia

Como podemos observar, el valor del sensor no es exactamente igual al valor aproximado que tendría que dar. Esto se debe a dos motivos, uno y primordial es que, el movimiento del sensor es manual y eso provoca algunos grados de imprecisión. El segundo motivo, es porque en esta toma de datos no se ha tenido en cuenta el punto inicial de los tres ejes, es decir, en la posición en la cual los tres movimientos están marcando 0 grados respectivamente. Igualmente, para deducir su rango nos es un poco indiferente.

Una vez explicado esto, ya podemos deducir, viendo los valores leídos en cada caso, el rango que tiene cada movimiento. El rango de Pitch (eje X) podemos ver que va de 180º a -180º, el rango de Roll (eje Y) va de 90º a -90º y el rango de Heading (eje Z) de 0º a 360º.

Esta información, se ha podido corroborar posteriormente con el datasheet del sensor BNO055 que está en el según en el *Anexo III Esquemas y datasheets de los dispositivos*, el cual determina el rango de los tres movimientos. Gracias a la prueba anterior ahora se conoce que el Roll, cuando te pasas del 90º o del -90º empieza a decrementar el número. Por lo tanto, en caso de sobrepasar el rango, se conoce como va a actuar el sensor dando datos angulares y, en caso de alguna anomalía, se puede deducir si es por este motivo o no.

Rotation angle	Range (Android format)	Range (Windows format)
Pitch	+180° to -180° (turning clockwise decreases values)	-180° to +180° (turing clock-wise increases values)
Roll	-90° to +90° (increasing with increasing inclination)	
Heading / Yaw	0° to 360° (turning clockwise increases values)	

Tabla 10. Rango de ángulos del Pitch,Roll y Heading. Fuente: "datasheet"

Una vez ya se conocen los rangos angulares de los tres movimientos que se obtienen del sensor, hay que saber cuáles son los del robot para poder sincronizarlos correctamente.

B. Rangos de ángulos del Robot

Para empezar, hay que ser consciente de que el robot tiene 6 grados de libertad. Cada uno de ellos podría tener un rango distinto. Por ese motivo, en primer lugar, hay que revisar su ficha técnica que está en el *Anexo III Esquemas y datasheets de los dispositivos*, donde se puede encontrar la siguiente tabla:

Movimiento		
Repetibilidad según la norma ISO 9283	± 0,03 mm	
Movimiento de ejes	Rango de trabajo	Velocidad máxima
Base	± 360°	± 180°/s
Hombro	± 360°	± 180°/s
Codo	± 360°	± 180°/s
Muñeca 1	± 360°	± 360°/s
Muñeca 2	± 360°	± 360°/s
Muñeca 3	Infinito	± 360°/s
Velocidad TCP estándar	1 m/s (39,4 in/s)	

Tabla 11. Rango de movimiento angular de los 6 motores del UR3 e. Fuente: "data sheet"

Con lo que se puede observar en la Tabla 11 y en el apartado de movimiento de la consola del robot, se pueden sacar algunas conclusiones. Una de ellas es que todos los motores, menos el de la Muñeca 3, tienen un límite de 2 vueltas completas (el de la muñeca 3 es infinito). Otra conclusión es el rango angular que tienen estos motores. Estos rangos van de 360º a -360º. El de la muñeca 3, es el mismo rango, pero, para este almacena la cantidad de vueltas que lleva para un sentido o para el otro, como se puede ver en la siguiente imagen, que lleva -2 vueltas.

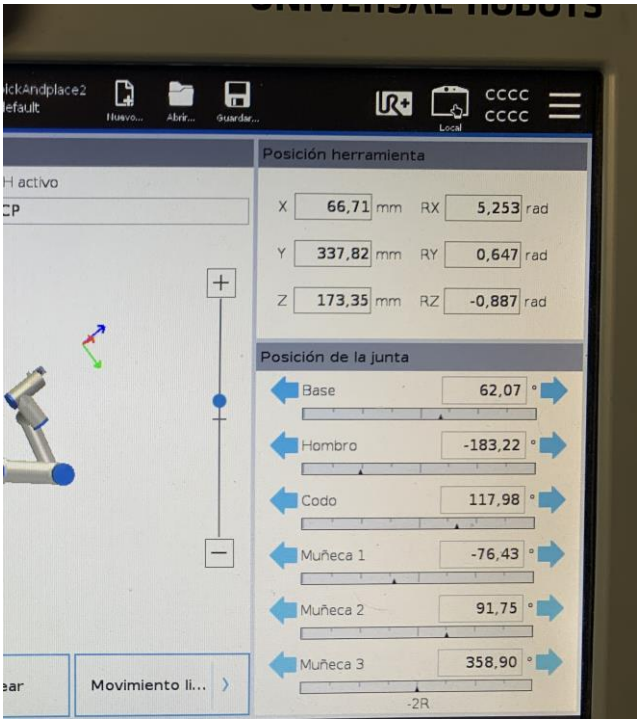


Ilustración 45. Grados del UR3 e con Muñeca 3, en -2 vueltas. Fuente: Elaboración propia

C. Determinar la posición inicial del robot

Una vez que se tienen los dos tipos de rangos que se tiene que sincronizar, hay que decidir que postura del robot se va a definir como postura inicial. Esta postura tendrá que ser la misma en el robot que en el brazo. Para llevar a cabo esta decisión hay que saber las limitaciones físicas del robot, de la similitud y de las diferencias de un brazo robot con respecto a un brazo humano. Hay motores del robot que, para evitar colisiones, no pueden girar en ciertas posturas. Aparte, un ejemplo sería que poner el brazo robot totalmente horizontal, no es lo mismo que el de un brazo humano totalmente en horizontal. Ahora, fijándose en estas observaciones y con varias pruebas realizadas, incitó con el robot físicamente. Se ha acabado fijando como posición inicial el brazo humano completamente extendido en horizontal, determinado así el símil de esa postura con respecto al brazo robot. De esta manera, todos los movimientos del brazo humano son posibles de ser realizados por el brazo robot. En la siguiente imagen se puede observar dicha posición inicial.

Ilustración 46. Posición inicial del UR3 e. Fuente: Elaboración propia

D. Detección del brazo humano

En primer lugar, hay que ser consciente de la necesidad de dos chips BNO055, para poder detectar todas las posiciones del brazo humano. Porque, como ya se ha explicado, los movimientos Pitch, Roll y Heading detectan el ángulo de tres ejes de libertad, por lo tanto, quedan otros tres para llegar a los seis que son los que tiene un brazo.

Para la correcta lectura de los grados de libertad hay que tener muy en cuenta dónde están

esos ejes de libertad y, partiendo de eso, colocarlos en el sitio correcto para que cada una de las lecturas sea diferente a la del otro sensor y así conseguir una lectura útil.

El primer sitio donde se pueden generar tres movimientos diferentes (Pitch, Roll y Heading) es la muñeca, es decir, los tres posibles movimientos que se pueden hacer con esta parte del cuerpo se pueden realizar con respecto a tres ejes distintos unos de los otros. Para el resto del brazo, se pueden identificar tres articulaciones distintas: el giro del tronco, el hombro y el codo. Podemos definir cada uno de ellos con un movimiento distinto, haciendo así que tengamos los otros tres Pitch, Roll y Heading. En la imagen que se muestra a continuación se pueden apreciar los distintos grados de libertad del brazo con respecto el robot y donde están situados.

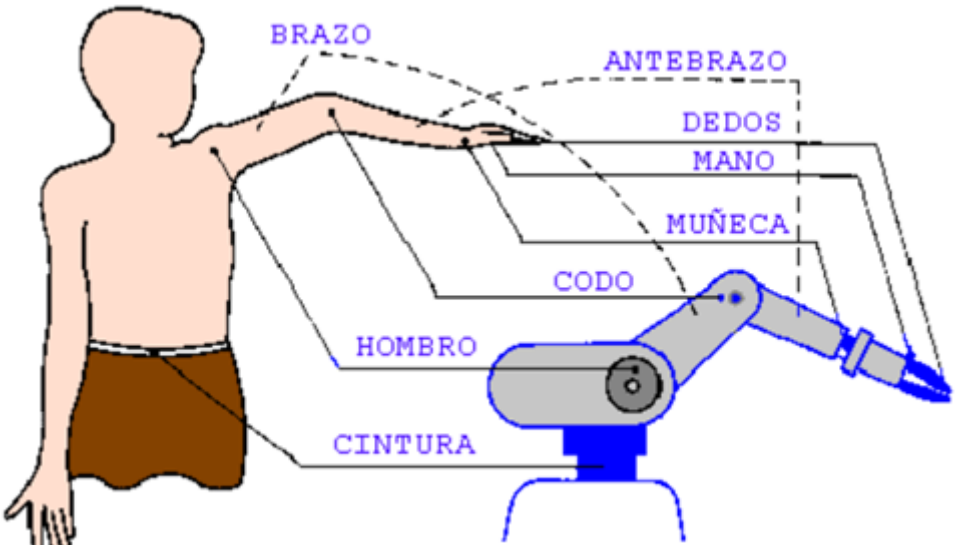


Ilustración 47. Semejanza de un brazo manipulador con la anatomía humana. Fuente: Robots industriales.

Con esta premisa, el primer IMU estará situado en la palma superior de la mano, detectando los tres últimos grados de libertad y, el otro, estará colocado en la parte superior del codo, en el lado más cercano a la mano, teniendo en cuenta que el brazo está extendido con la palma de la mano hacia abajo. Para hacerlo más fácil a continuación se puede la posición exacta.

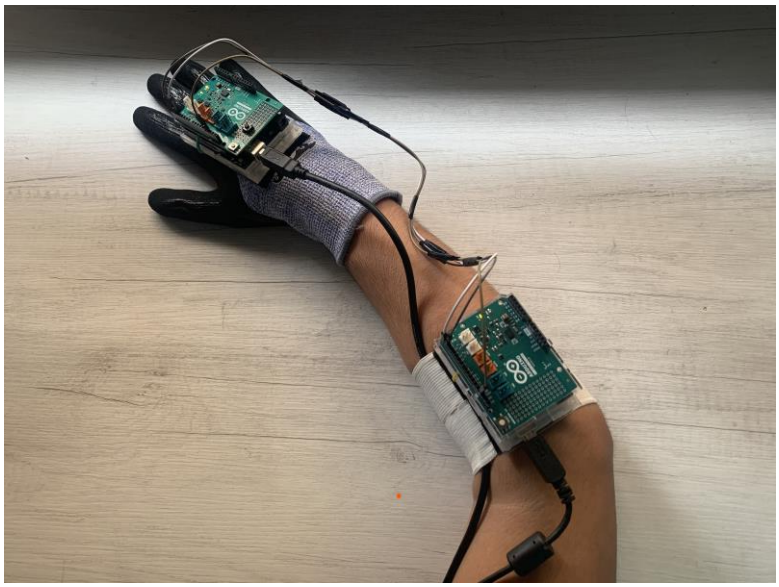


Ilustración 48. Posición de los IMU en el brazo humano. Fuente: Elaboración propia

E. Sincronismo con los dos brazos

En este paso se deber tener en cuenta toda la información recolectada en los apartados anteriores, los rangos de los ángulos de los dos dispositivos, tanto del sensor como del robot, y la posición inicial del robot, e implementarla toda junta para que el movimiento de ambos brazos sea igual.

Algo bastante importante es que el brazo robot no tiene los tres primeros motores con un movimiento con respecto a cada uno de los tres ejes (ejes x, y, z). Es decir, que los motores del codo y del hombro del robot se mueven los dos respecto del eje, en este caso el eje x. Eso significa que hay un movimiento que no podrá imitar el robot físicamente, el Roll.

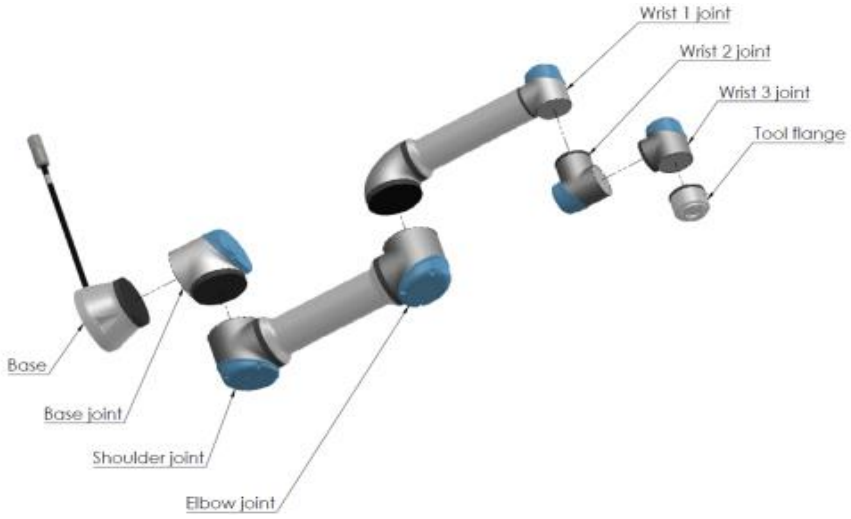


Ilustración 49. Partes y motores del robot. Fuente: Manual Ur3 e.

Conociendo esta limitación existente, se debe escoger cuál de los dos motores será controlado con el Pitch del sensor. La elección ha sido meramente de movilidad para poder realizar tareas con el robot, ya que, no es exactamente igual la movilidad con el hombro recto que con el codo extendido. En las siguientes imágenes se puede ver cómo queda la distribución de los movimientos con respecto de cada motor del robot.

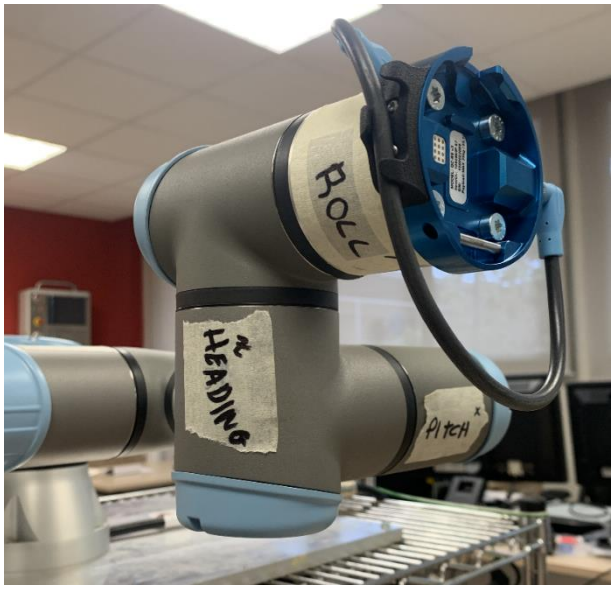


Ilustración 50. La Muñeca3(Roll Y), Muñeca 2 (Heading Z) y Muñeca 1(Pitch X). Fuente: Elaboración propia

Ilustración 51. El Codo (Pitch X) y Base (Heading Z). Fuente: Elaboración propia

El siguiente punto a seguir es fijarse en qué grados está el robot en esa posición inicial que hemos definido anteriormente. Sabiendo eso y su rango, se tendrá que adaptar los valores que lee el sensor con los que tiene el robot para que se muevan con la misma referencia angular. En esta imagen se pueden observar los valores angulares que tiene

cada uno de sus motores en la consola del UR3e.

Ilustración 52. Posición inicial del UR3 e con los grados de cada motor. Fuente: Elaboración propia

Como se puede ver es [0, -π rad , 0, 0, +π/2 rad, 0] donde se obtienen que movimientos van en cada motor [Heading Hombro, será siempre -π rad, Pitch Codo, Pitch, Heading, Roll]. Como se ha comentado anteriormente, el de -π rad, se va a mantener estático. Ahora igual, pero con los rangos.

- **Rangos del robot** [-2π rad a 2π rad, -π rad, -2π rad a 2π rad, -2π rad a 2π rad, -2π rad a 2π rad, -∞ a ∞]
- **Rangos en el que los sensores enviarían sus datos** [0 a 2π rad, -π rad, π rad a -π rad, π rad a -π rad, 0 a 2π rad, -π rad a π rad].

De esta manera se puede ver que todos menos los dos Heading, tienen los rangos de los sensores compatibles con los del robot pero con menos recorrido. Para obtener el rango del Heading Hombro se deben hacer cálculos para que en vez de ser de 0 rad a 2π rad sea π rad a -π rad, mientras que el otro Heading, se tiene que tomar en cuenta que cuando inicializa el sensor en la posición inicial tendrá que marcar como si estuviera en π/2 rad.

A continuación, se plantearán los cálculos que se han introducido en el código. Se ha generado una función que pasa los datos que van de 0 rad a 2π rad, pase de π rad a -π rad.

```
//Funcion para que el Eje Z en Radianes y tenga un rango de -PI rad a PI rad
float convertir_EjeZ_RAD(float angle) {
  if (angle > PI) {
    angle = angle - (2*PI);
  }

  return angle;
}
```

Ilustración 53. Función de código para pasar un rango de 0 a 2π a otro de -π a π. Fuente: Elaboración propia

El cálculo que se genera en esta función, es que, si el ángulo medido es más grande que πrad, se resta 2πrad al valor tomado por el sensor, haciendo así la parte negativa de este rango. Ahora bien, el Heading del otro sensor, que es el que empieza en π/2 rad, no es necesario utilizar esta función. Simplemente, se le sumará π/2 rad, en el inicio, provocando que los límites de movimiento este de 0 rad a π/2 rad.

F. Punto de referencia de los grados

Por último, e igual de importante, hay que saber el punto de referencia de cada uno de los movimientos. Se ha podido identificar que antes de la calibración, los datos de los ángulos toman esa posición inicial, como 0º. Pero como se auto calibra el sensor a medida que vas haciendo los movimientos que podrían calibrarlo, el punto de referencia cambia a mitad de la transmisión de datos. Con eso hay que tener claro que primero hay que calibrar siempre y después hacer los cálculos, si no, no tendrán sentido los valores.

Lo que se sabe es que el punto de referencia del Heading está referenciado a partir de donde está el norte. Lo encuentra con el magnetómetro. El Pich y el Roll, están referenciados a partir del acelerómetro que detecta la aceleración de la gravedad, haciendo que la horizontal sea su punto de referencia.

Ahora se tienen que realizar los cálculos para que se tenga el punto de referencia con respecto de la posición inicial, pero como la posición inicial del robot es en horizontal, solo habrá que cambiar el punto de referencia de los Heading. Los cálculos a realizar se pueden observar en el siguiente código.

```
//Le resta el valor que toma el sensor - el valor inicial
  ejez0=(valorz-iniz);
if(valorz<iniz){ // el la valor que toma el sensor es mas pqueño que el valor inicial
  ejez0=(2*PI)-ejez0; //la resta sacas el valor mas grande que puedes
                      //sacar simplemnete restando y le sumas el valor inicial.
}
```

Ilustración 54. Función de código para cambiar su punto de referencia. Fuente: Elaboración propia

El siguiente paso es sumarle los $\pi/2$ rad que tiene el robot en su posición inicial en el motor de la muñeca 2. Este proceso no es solo sumarle $\pi/2$ rad, sino que hay que cambiar el punto de referencia a $\pi/2$ rad. En el código se ha calculado de la siguiente manera.

```
//Aumenta el angulo de Z en 90 para que coincida con los angulos del robot
    ejez=ejez0+(PI/2);

    if(ejez>(2*PI)){  //Si el angulo que da es mayor a una vuelta

    ejez=ejez-(2*PI);  //Se le resta una vuelta para acabar todo el marjen del rango del sensor

    }
```

Ilustración 55. Función de código para aumentar $\pi/2$ para el rango del robot. Fuente: Elaboración propia

En este punto se identificó un posible error de imitación, ya que hay que tener en cuenta que el sensor de la muñeca puede detectar los ángulos del otro sensor. Para poder eliminar esas lecturas que no interesan que lea, porque se acumularan a su movimiento, tendremos que restarlas. Simplemente hay que preocuparse de las lecturas de la muñeca.

```
//Recivir datos del esclavo
    if (Serial1.available() >= 12) { // Espera a que se reciban 12 bytes
    ejex1 = Serial1.parseFloat();  //Guarda el dato dels sensor en variable ejex1
    ejey1 = Serial1.parseFloat(); //Guarda el dato dels sensor en variable ejey1
    ejez1 = Serial1.parseFloat(); //Guarda el dato dels sensor en variable ejez1

    }

ejex=ejex-ejex1;  //Eliminar lecturas de grados residuales del ejex

ejez=ejez-ejez1;  //Eliminar lecturas de grados residuales del ejex
```

Ilustración 56. Función de código para eliminar lecturas residuales de la Muñeca 3 y Muñeca 2. Fuente: Elaboración propia

El siguiente paso ha sido probar el código realizado, donde se ha detectado que el robot imitaba correctamente el brazo humano, pero tenía un error que no se supo ver antes, y es que hacia las órdenes al revés. Es decir, si el brazo humano gira hacia la derecha, el

brazo robot gira hacia la izquierda. Al identificar qué le ha pasado a todos los motores, hay que hacer los cálculos correspondientes para que los valores sigan el orden opuesto al que estaban haciendo hasta ahora. Hay que tener en cuenta los rangos y sobre todo el de la muñeca 2 que tiene $\pi/2$ rad en el estado inicial, ya que es el único que no se soluciona cambiando el signo. A continuación, se vera la parte donde cambia de sentido los grados de la muñeca 2:

```
//Cambia el sentido de giro para evitar que el robot haga el moviemiento contrario
  if(ejez>PI/2){ //Si es mayor que 90 grados
    ejez=ejez-(PI/2); //al valor medido le restas 90
    ejez= (PI/2)-ejez; // y ahora le restas la diferencia a 90 quedandote lo contrario
    if(ejez<0){    // Si el resultado de los calcuos es un valor 0 o negativo
      ejez=(2*PI)+ejez; // Se le suma una vuelta entera
    }
  }
  else{    //En caso de no ser mayor que 90
    ejez= (PI/2)-ejez;  //Hace lo calculos contrarios que arriva para que te quede lo contrario
    ejez= (PI/2)+ejez;
  }
```

Ilustración 57. Función para cambiar el sentido a la Muñeca 2. Fuente: Elaboración propia

Hay que mencionar que en un primer momento se tenían dudas de si sería necesario implementar un real-time, con una interrupción de tiempo. Pero al ver que la parte de ejecución del código no tardaba más de 31 milisegundos y que el puerto del servidor 30003 es de 1/500Hz= a 2 ms, en vez del resto de puertos que van a 1/10Hz = 0,1 s. Se observó que eligiendo el puerto 30003 ya no habría ningún problema debido al tiempo de ejecución del programa.

G. Flujograma

Gracias a todo el proceso anteriormente descrito y el flujograma obtenido, se puede desarrollar todos los pasos para realizar el código final. En el *Anexo V Flujogramas*, el Flujograma 3 es el flujograma del código final. Esto ha ayudado a programar el código evitando errores de proceso, y posteriormente a la comprensión y entendimiento de las demás personas que tengan que trabajar con este código en un futuro. Además, también se puede encontrar los dos códigos finales de los dos Arduino están en el *Anexo IV Código*.

5. PRESUPUESTO

A continuación, se muestra el presupuesto total que se a necesitado para llevar a cabo el proyecto. Para ello se ha dividido en tres grupos diferentes: los materiales, el tiempo de ingeniería que se ha invertido.

5.1. Presupuesto de materiales

En este apartado se mostrará el presupuesto para la compra de materiales necesarios para la elaboración del montaje del robot, creación del dispositivo que detecta la posición del brazo humano y el material extra para la parte de investigación. El presupuesto para tener el material para montar el robot es el siguiente:

MATERIAL	PROVEEDOR	CANTIDAD	PRECIO
UR3 e	ELEKTRES	1	17.908€/u
Prensaestopa de M20	Ware Leads	1	10,96€/10 u
Cable Ethernet	UGREEN	1	9,99€/u
Wifi router inalámbrico	TP-Link	1	19,89€/u
Estantería 2 baldas con ruedas	Esmelux	1	244,55€/u
Plancha de aluminio	CNCrobotica	1000x1000x10mm	198,44€
Tornillo de cabeza hexagonal M6*20mm	NOLDAR	6	2,30€/20u
Arandela de M6	NOLDAR	6	2,30€/20u
Tuercas hexagonales autoblocantes de M6	AERZETIX	6	10,10€/10u

Tabla 12. Precio material para instalar el UR3 e. Fuente: elaboración propia.

El presupuesto total del material necesario para poder llegar a hacer la instalación del robot es de **18.479,41€.**

Por otro lado, el presupuesto de los materiales necesarios para la fabricación del dispositivo que detecta la posición del brazo humano, viene dado por la siguiente tabla:

MATERIAL	PROVEEDOR	CANTIDAD	PRECIO
Arduino Uno Rev2	Arduino	2 unidades	46,70 €/u
Arduino Shild 9 Axis Motion	Arduino	2 unidades	28,70€ /u

Guante de trabajo Krytech 582	RS	1 unidad	22,51€ un par
Cinta elástica de xxmm	SOFTY	44 cm	1,60€/m
Velcro de 20mm de ancho	Rietlow	86 cm	1,22€/m
Pulsador para circuitos	HUAZIZ	1 unidad	0,04€/u
Cable puente 15 cm macho a macho (cable plano)	WHADDA	unidades>6	4,95€/40Pins
Cable puente 30cm macho a hembra (cable plano)	WHADDA	3 unidades	4,95€/40Pins
Cable USB 2.0 tipo A/B	MYAMIA	2 unitats	3,96€/u

Tabla 13. Precio material para instalar el UR3 e. Fuente: elaboración propia.

El presupuesto total de los materiales para el dispositivo que detecta la posición del brazo humano es de **192,93€**.

Por último, el material usado para la investigación y desarrollo del proyecto:

MATERIAL	PROVEEDOR	CANTIDAD	PRECIO
Led Kingbright verde	RS	1	3,87€/5u
Resistencia de 220 Ω	RS	1	3,64€/10u
Resistencia de 15 KΩ	RS	1	3,91€/10u
Protoboard	RS	1	17,82€/u
Servomotor	RS	3	22,54€/u

Tabla 14. Precio del material para investigación. Fuente: elaboración propia.

El presupuesto total del material para investigación y desarrollo es de un total de **41,89€.** Por lo tanto, el presupuesto total respecto a los materiales necesarios es de un total de **18.714,23€.**

5.2. Presupuesto del tiempo de ingeniería

En este apartado se pretende dar un presupuesto aproximado a las horas de ingeniería dedicadas en este proyecto: investigación, programación y el diseño i fabricación del soporte del sensor. Una vez sabemos esto hay fijar la cantidad de personas están implicadas y que tarifa tiene cada uno. En este caso se ha estimado que, para realizar estas diferentes tareas de este proyecto, las ha realizado una ingeniera junior. Como seria normal en una situación real, estas tareas se dividirían entre diferentes personas y eso implica que este presupuesto se vería modificado considerablemente.

TIPOS DE PERSONAL	TARIFA (€/H)
Ingeniero/a Junior	20

Tabla 15.Sueldo de Ingeniero/a Junior. Fuente: elaboración propia.

Ahora una vez definido el precio por hora, se deben definir por grupos todos los puntos desarrollados durante la elaboración del proyecto, así como la cantidad de horas que se han empleado en cada uno de ellos.

	HORAS	TARIFA (€/H)	TOTAL (€)
Montaje del Robot	15	20	300
Investigación	50	20	1.000
Programación	70	20	1.400
Creación de los soportes del sensor	10	20	200
Elaboración memoria técnica	50	20	1.000
Elaboración anexos memoria	30	20	600
Gestión administrativa	10	20	200
TOTAL	-	-	**4.700€**

Tabla 16. Costes totales de horas de ingeniería. Fuente: elaboración propia.

Por lo tanto, el presupuesto total para las horas de ingeniería que se han invertido en este proyecto sería de **4.700€**.

5.3. Presupuesto total del proyecto

A continuación, se pretende calcular el presupuesto aproximado total del proyecto, teniendo en cuenta todos los puntos anteriormente desarrollados.

Por un lado, se ha generado el presupuesto necesario para comprar todo el material

necesario y este valor a dado **18.714,23€.**

Por último, también están contabilizadas las horas de ingeniería empleadas en el proyecto da un valor de **4.700€.**

Por lo tanto, el coste total de este proyecto, sumando todas las partes, tendría un valor de **23.414,23€.**

A continuación, se muestra mediante el siguiente gráfico circular, el desglose de cada uno de los costes previamente estudiados y que representan cada uno de ellos en relación al total.

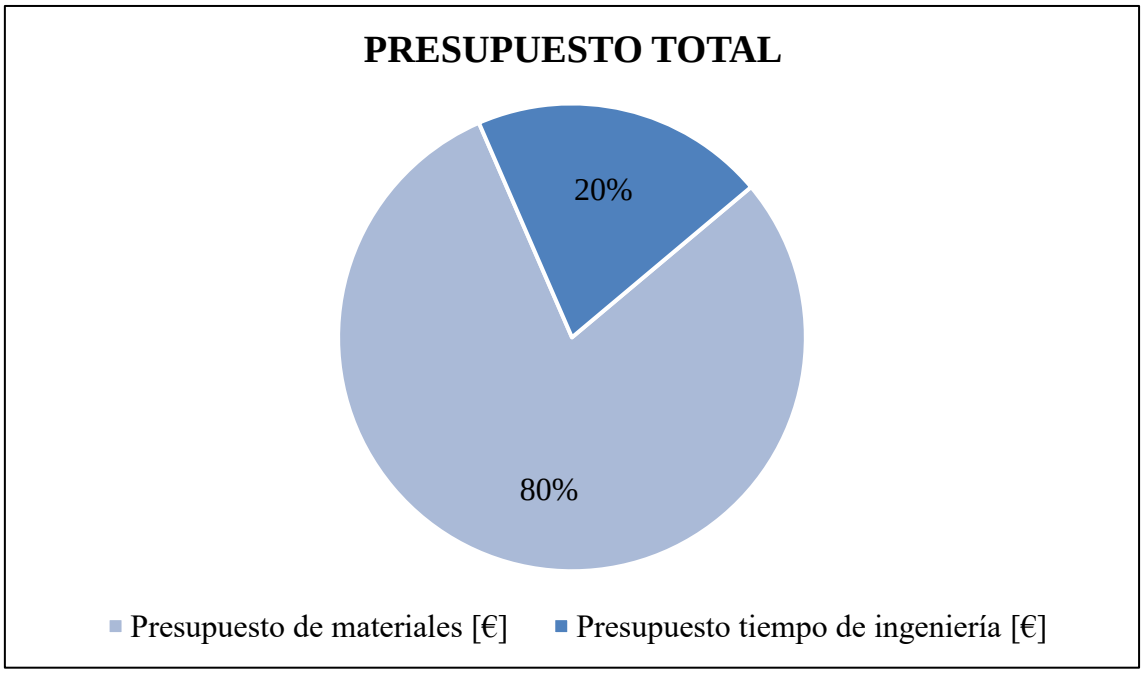


Ilustración 58. Distribución de costes del proyecto. Fuente: elaboración propia.

CONCLUSIONES

Una vez finalizado este proyecto se puede afirmar que cumple con las bases fijadas en la fase inicial. El objetivo principal de esta investigación, es conseguir que el brazo robot sea capaz de imitar el brazo humano, para poder llegar a realizar tareas a distancia sin la necesidad de estar en la misma sala que el robot, se ha podido desarrollar satisfactoriamente.

En este proyecto se ha podido observar la viabilidad y eficacia del control de los movimientos del brazo robot UR3e de Universal Robot, a través de la tecnología de los sensores inerciales y las distintas comunicaciones, haciendo posible la imitación del movimiento del brazo humano con el robot. Se ha obtenido una imitación muy realista del movimiento humano, demostrando así su viabilidad para aplicaciones futuras en entornos industriales y de investigación.

Aunque el proceso de calibración de los distintos sensores es algo engorroso, una vez calibrados y dentro de los parámetros, el sistema demuestra ser estable y de poder adaptarse a diferentes tareas.

Sin embargo, se han podido identificar ciertas limitaciones. En primer lugar, se han encontrado limitaciones angulares de los sensores con respecto el rango de movilidad del robot UR3e, ya que, existen diferencias físicas entre un brazo humano y el robot UR3e, aunque las dos tiene la misma capacidad motriz en el espacio. Además, la necesidad de tener el dispositivo del brazo humano conectado a un ordenador, dificulta el movimiento de este.

En conjunto, este proyecto aporta conocimientos para futuras investigaciones y desarrollos sobre el control de robots mediante imitación de movimiento, para así hacer una aportación al sector la automatización industrial y mejorar la interacción hombre-máquina en diversas aplicaciones.

AGRADECIMIENTOS

En primer lugar, quisiera dar las gracias a mi tutor de proyecto, Óscar de Sousa, y al equipo de STL (Serveis Tècnics de Laboratori), por ayudarme en todo lo que necesitara, enseñarme tantísimas cosas nuevas y hacer de este proyecto mucho más ameno y llevadero, disfrutándolo al máximo pese a su dificultad.

También, quería agradecer a Antonio Camacho por introducirme en el mundo de los sensores inerciales y de la programación.

Por último, agradecer a toda mi familia y amigos por su constante apoyo, ánimo y paciencia conmigo durante estos largos meses de duro trabajo.

WEBGRAFIA

[1] Tipos de robots: clasificación, aplicacions y ejemplos. Disponible en: https://www.telefonica.com/es/sala-comunicacion/blog/tipos-de-robots-clasificacion-aplicaciones-y-ejemplos/

[2] Cobots y robòtica colaborativa – Que son, que funciones pueden realitzar y sus ventajas. Universal Robots. Disponible en: https://www.universal-robots.com/es/cobots-robots-colaborativos/

[3] ¿Qué es un cobot o robot colaborativo? IEBS. Disponible en: https://www.iebschool.com/blog/que-es-cobot-robot-colaborativo-tecnologia/#:~:text=Un%20cobot%20o%20robot%20colaborativo%2C%20es%20aquel%20que%20opera%20en,compartido%20y%20de%20forma%20eficiente.

[4] El UR3e de Universal Robots. Disponible en: https://www.universal-robots.com/es/productos/robot-ur3/

[5] Molpeceres Jorge, María, "Simulación robòtica colaborativa con ros" , 2020. Disponible en: https://uvadoc.uva.es/bitstream/handle/10324/41143/TFG-I-1490.pdf?sequence=1

[6] Las cinco generaciones de la robòtica. Disponible en: https://automatismosmundo.com/las-5-generaciones-de-la-robotica/

[7] Tema 5.4. Robots Industriales. Disponible en: 5.4 Robots industriales (mec.es)

[8] C. Bittner, H. Bode, and A. Christ, "The Safety Compendium," 2013. https://www.eltron.pl/files/photos/producers/457/8ff/4578ffa1c74da6fb257a8f464de8607d75575e77.pdf

[9] Omron. Una guia ràpida para la Seguridad de robots colaborativos. Disponible en: Una guía rápida para la seguridad de robots colaborativos | Omron

[10] Omron. Seguridad para robots colaborativos. Disponible en: Seguridad para robots colaborativos | Omron

[11] ¿Qué es una IMU y para qué se utiliza? . Disponible en: IMU: Qué es una IMU y para qué se utiliza | UAV Navigation

[12] Demostración y aplicación de la fórmula de Euler. Disponible en: ¿Qué dice la fórmula de Euler? | Demostración y aplicaciones (micalculadoracientifica.com)

[13] Euler angels, Wikipedia. Disponible en: Euler angles - Wikipedia

[14] Calculatin displacments using Accelerometer amb Gyroscope (MPU6050). Disponible en: position - Calculating displacement using Accelerometer and Gyroscope

(MPU6050) - Stack Overflow

[15] Ángulos de Euler Yaw, Pitch, Roll. Disponible en: Ángulos de Euler Yaw, Pitch, Roll - programador clic (programmerclick.com)

[16] Redes de comunicación industrial: todo lo que necesitas saber. Disponible en: Redes de Comunicación Industrial: todo lo que necesitas saber (sicma21.com)

[17] Universal Robots. Disponible en: https://www.universal-robots.com/es/

[18] E-Series OEM de Universal Robots. Disponible en: https://www.universal-robots.com/es/productos/oem-robots/

[19] Consola de programación e-series con dispositivo de validación de 3 posiciones. Disponible en: https://www.universal-robots.com/es/productos/e-series-3pe/

[20] Universal Robots, "The URScript Programming Language" 2013. Disponible en: https://www.zacobria.com/pdf/universal-robots-scriptmanual-en-v-1-8.pdf

[21] "Open modbus/TCP specification" 1999. Disponible en: https://wingpath.co.uk/docs/modbus_tcp_specification.pdf

[22] Profinet- the leading Industrial Ethernet Standard. Disponible en: https://www.profibus.com/technology/profinet/

[23] Comunicación inteligente y colaborativa entre robots, Universal Robots. Disponible en: Comunicación Inteligente Robots | Universal Robots (universal-robots.com)

[24] UNO Wifi Rev2, Arduino. Disponible en: **UNO WiFi Rev2 | Arduino Documentation**

[25] Nano 22 IoT, Arduino. Disponible en: **Nano 33 IoT | Arduino Documentation]**

[26] MKR Wifi 1010, Arduino. Disponible en: **MKR WiFi 1010 | Arduino Documentation**

[27] Nano RP2040 Connect, Arduino. Disponible en: **Nano RP2040 Connect | Arduino Documentation**

[28] Datasheet. Disponible en: SM6DS3TR-C Datasheet(PDF) - STMicroelectronics (alldatasheet.com)

[29] Smart sensor: BNO055, Bosch. Disponible en: Smart Sensor BNO055 | Bosch Sensortec (bosch-sensortec.com)

[30] 9 Axis Motion Shield, Arduino. Disponible en: 9 Axis Motion Shield | Arduino Documentation

[31] Offline simulator – e-series - URSim for non Linux 5.9.4, Universal Robots. Disponible en: https://www.universal-robots.com/download/software-e-series/simulator-

non-linux/offline-simulator-e-series-ur-sim-for-non-linux-594/

[32] Host a Web Server on the Arduino UNO Wifi Rez 2. Disponible en: Host a Web Server on the Arduino UNO WiFi Rev2 | Arduino Documentation

[33] WiFiNINA, Arduino. Disponible en: WiFiNINA - Arduino Reference

[34] Mimic: Deburing, Nordbo Robotics. Disponible en: https://www.youtube.com/watch?v=8ijD-E7a-EE

[35] Mimic with IR Tracker: Demostration, Nordbo Robotics. Disponible en: https://www.youtube.com/watch?v=WqaS1p9BVHA

[36] Universal Robots - URCap - Documentos de referencia de API (universal-robots.com)

ANEXOS

ANEXO I: NORMATIVA DE SEGURIDAD DEL ROBOT

ANEXO II: MANUAL DE FUNCIONES UR3E

ANEXO III: ESQUEMAS Y DATASHEETS DE LOS DISPOSITIVES

ANEXO IV: CÓDIGO

ANEXO V: FLUJOGRAMAS

ANEXO VI: TEORÍA DE CINEMÁTICA

ANEXO VII: LIBRERÍAS Y SUS MODIFICACIONES

ANEXO VIII: UR3E MANUAL

ANEXO IX: CONFIGURACIÓN DEL ROUTER

ANEXO X: URSIM GUÍA INSTALACIÓN

ANEXO XI: COMPROBACIÓN DE LA COMUNICACIÓN SÈRIE

ANEXO XII: PROTOCOLO DE COMUNICACIÓN