Escuela
Politécnica
Superior

Teleoperación de un robot para acciones de manipulación con dispositivos y sensores de bajo coste

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Daniel Pastor Bernal
Tutor/es:
Pablo Gil Vázquez
Santiago Timoteo Puente Méndez

Julio 2023

Universitat d'Alacant
Universidad de Alicante

Teleoperación de un robot para acciones de manipulación con dispositivos y sensores de bajo coste

Autor
Daniel Pastor Bernal

Tutor/es
Pablo Gil Vázquez
Fisica, Ingenieria De Sistemas Y Teoria De La Señal
Santiago Timoteo Puente Méndez
Fisica, Ingenieria De Sistemas Y Teoria De La Señal

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2023

Preámbulo

"El uso de la robótica está cada día más en auge, sobretodo en procesos industriales donde las empresas emplean robots para realizar tareas que puedan resultar peligrosas para el operario. Este proyecto surge de mi interés en conseguir poder introducir la robótica en ámbitos más lúdicos, empleando así dispositivos de bajo coste y software de código libre para teleoperar un robot. De esta forma lograr que más gente pueda iniciarse en la robótica y aprender jugando"

Agradecimientos

En primer lugar, me gustaría agradecer a mis tutores Santiago T.Puente y Pablo Gil por su orientación y apoyo. Su experiencia y conocimientos han sido fundamentales para la realización de este trabajo.

También quiero expresar mi gratitud a todos los profesores y compañeros que me han ayudado a lo largo de estos años de carrera a adquirir los conocimientos necesarios para poder llevar a cabo este proyecto.

Merecen una mención todas las fuentes de información, libros, artículos científicos y recursos en línea que he utilizado para investigar y desarrollar mi TFG. Sin el acceso a estos recursos, mi trabajo no habría sido posible.

Todos los aprendizajes
más importantes de la vida
se hacen jugando

Francesco Tonucci.

Índice general

Índice de figuras

Índice de tablas

Índice de Códigos

1 Introducción

La robótica ha experimentado un gran auge en los últimos años, especialmente en la industria, donde ha revolucionado la forma en la que se trabaja y se fabrican los productos. Los robots industriales han permitido una mayor eficiencia y productividad, así como una reducción de costes y una mejora en la calidad de los productos. Sin embargo, la robótica también tiene mucho potencial en otros ámbitos, como la educación y el entretenimiento.

Hasta hace poco, la robótica se limitaba a ámbitos muy especializados y estaba fuera del alcance de la mayoría de las personas debido a su alto costo y complejidad. Sin embargo, el surgimiento de dispositivos de bajo costo como Arduino ha hecho posible que cualquier persona con un poco de conocimiento en programación pueda crear su propio robot y explorar el mundo de la robótica.

Esto ha abierto un sinfín de posibilidades en ámbitos como la educación y el entretenimiento. Ahora es posible enseñar a los niños sobre robótica y programación de una manera más accesible y divertida, y también crear robots para juegos y competiciones.

En este trabajo se busca teleoperar un brazo robótico empleando dispositivos y sensores de bajo coste, en concreto se utilizará la placa Arduino. Tambíen, el uso de software libre como ROS y Arduino Integrated Development Environment (IDE) es una opción popular en la robótica, especialmente para aquellos que buscan desarrollar soluciones personalizadas y de bajo costo. Estas herramientas tienen muchas ventajas en términos de accesibilidad, flexibilidad y colaboración, lo que las hace ideales para una amplia variedad de aplicaciones robóticas.

ROS es un software libre muy utilizado en la robótica que proporciona una plataforma para desarrollar y ejecutar aplicaciones robóticas complejas. ROS tiene capacidad para abstraer la complejidad de los sistemas robóticos y proporcionar un conjunto de herramientas de software útiles para los desarrolladores.

Por otro lado, el IDE de Arduino también es una herramienta de software libre muy popular utilizada para programar placas de desarrollo de Arduino. Este IDE proporciona una interfaz gráfica fácil de usar que permite a los usuarios programar y depurar aplicaciones en lenguaje C o C++. Arduino se ha convertido en una opción popular para la robótica de bajo coste, especialmente para aquellos que buscan desarrollar robots y sistemas de control personalizados con un presupuesto bajo.

La principal ventaja de utilizar software libre como ROS y Arduino es su accesibilidad y su capacidad para adaptarse a una amplia variedad de necesidades y requerimientos. Al ser software libre, se puede personalizar, modificar y distribuir libremente, lo que significa que los

desarrolladores pueden adaptarlo a sus necesidades específicas. Además, el uso de software libre también fomenta la colaboración y el intercambio de conocimientos, lo que a su vez puede conducir a una innovación más rápida y eficiente.

Ciertamente, es importante mencionar que trabajar con herramientas de software libre también presenta algunas limitaciones. En primer lugar, puede requerir una curva de aprendizaje adicional, ya que los usuarios deben familiarizarse con el entorno de desarrollo y sus funcionalidades. Además, la compatibilidad de hardware puede ser un factor a considerar, ya que algunos dispositivos pueden requerir configuraciones adicionales o no estar completamente integrados. Otro aspecto a tener en cuenta es el soporte técnico, que puede ser limitado y depender en gran medida de la comunidad de usuarios. Aunque estas limitaciones pueden presentar desafíos, en el caso específico de este proyecto, se ha logrado trabajar de manera satisfactoria superando estos obstáculos.

2 Marco Teórico

2.1 Teleoperación

La teleoperación hace referencia al control remoto de dispositivos o sistemas a distancia. Su origen se remonta al año 1947 en la industria nuclear, donde permitía al operario manipular material radioactivo desde un lugar seguro.

Figura 2.1: Primer manipulador maestro-esclavo (Emmanuel Nuño Ortega, 2004)

Este enfoque se expandió a diversas industrias, desde la exploración espacial y submarina, hasta la medicina y la industria manufacturera. La teleoperación permite realizar tareas en entornos peligrosos o inaccesibles para los humanos, añadiendo al mismo tiempo una mayor precisión.

2.1.1 Telerrobótica

La telerrobótica, por otro lado, es una rama de la teleoperación que estudia todo lo relacionado con el control remoto de robots. Esto ha sido posible gracias a la combinación de dos áreas que han tenido un gran desarrollo con el paso del tiempo: la robótica y las comunicaciones.

2.1.2 Telepresencia

Es importante definir también el termino de la telepresencia, un concepto relacionado con la teleoperación que busca proporcionar una sensación inmersiva, proporcionando la sensación de estar presente físicamente en un lugar remoto. Es decir, la sensación que se da al usuario de un sistema teleoperado de encontrarse en el lugar donde se encuentra el dispositivo o sistema que se está controlando. Esta tecnología se hace posible mediante diferentes disciplinas como la realidad virtual, realidad aumentada y otros enfoques de retroalimentación sensorial, que

logran crear una ilusión perceptual permitiendo al usuario interactuar y percibir el entorno remoto como si estuviera allí mismo.

2.1.3 Elementos de un sistema teleoperado

- Operador: Persona que se encarga de controlar y/o supervisar el dispositivo teleoperado empleando una interfaz. La intervención de este puede ser continua (Control directo) o puntual (Control supervisado).
- Dispositivo teleoperado o esclavo: Dispositivo o sistema ubicando en un lugar remoto que es controlado a distancia por el operador.
- Interfaz: Dispositivos que permiten la interacción entre el operador y el sistema teleoperado. Proporciona al operador los medios necesarios para recibir la información relevante y enviar los comandos para llevar a cabo acciones en el sistema remoto.
- Control: Módulo encargado de transformar los comandos enviados desde una interfaz en acciones entendibles por el dispositivo teleoperado.
- Canales de comunicación: Medio por el que se transmite la información de manera bidireccional entre el operador y el esclavo. Pueden ser cables, redes inalámbricas o incluso conexiones satelitales, dependiendo de la distancia y las necesidades de la teleoperación. Es fundamental contar con canales de comunicación confiables y de baja latencia para garantizar una interacción fluida y en tiempo real.
- Sensores: Elementos del sistema teleoperado que permiten captar información tanto del entorno como del propio sistema. Esta información que viaja a través de los canales de comunicación hacia el operador permitiendo una mejor toma de decisiones.

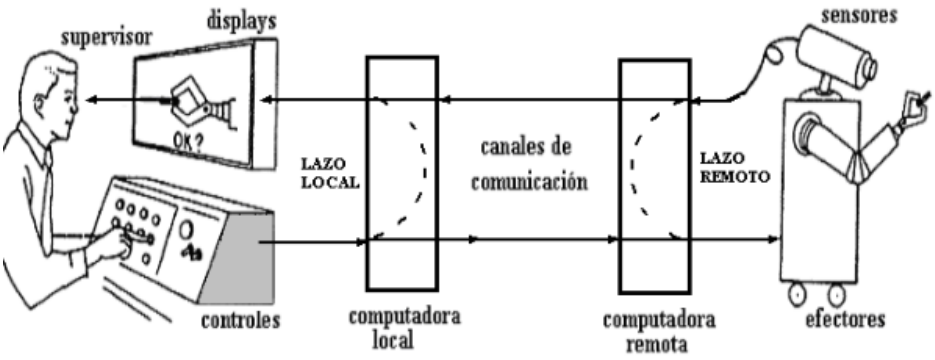

Figura 2.2: Elementos de un sistema teleoperado (Emmanuel Nuño Ortega, 2004)

2.2 Teleoperación en robots manipuladores

La teleoperación de robots manipuladores implica la utilización de dispositivos de interfaz para enviar comandos de control al robot y recibir información en tiempo real sobre su estado

y entorno. Uno de los dispositivos de interfaz comunes utilizados en la teleoperación de robots manipuladores es la consola de teleoperación.

La consola de teleoperación es un dispositivo portátil que proporcionan varios fabricantes de robots que otorgan al operador una interfaz para controlar el robot de manera eficiente y segura. Generalmete consta de un panel táctil, botones programables y joysticks para el control de movimientos.

Estos dispositivos ofrecen una variedad de funciones que facilitan la programación, el control y la supervisión del robot. Permiten al operador realizar acciones como mover los brazos del robot, cambiar entre diferentes modos de operación, ajustar la velocidad, configurar trayectorias de movimiento y realizar diagnósticos en tiempo real. Algunos ejemplos de fabricantes conocidos que ofrecen estos dispositivos son ABB con su Flex Pendant, Fanuc con su iPendant y KUKA con su SmartPad.

(a) Flex Pendant de ABB

(b) iPendant de Fanuc

(c) Smart Pad de KUKA

Figura 2.3: Ejemplos de consolas de teleoperación

3 Objetivos

En este proyecto se pretende teleoperar un robot manipulador con dispositivos y sensores de bajo coste (Unidad de Medición Inercial (IMU), ARDUINO, etc.). El objetivo es hacer uso de este tipo de dispositivos para detectar cambios de orientación y/o posición, y desarrollar algoritmos para interpretar esos cambios y generar acciones y movimientos en un robot remoto. La implementación de la comunicación entre dispositivos se llevará a cabo utilizando Robot Operating System (ROS), una plataforma de software ampliamente utilizada en robótica.

- Objetivos concretos

    1. Conocer el funcionamiento y hacer uso de sensores y dispositivos de bajo coste:

       Se realizará un estudio sobre el funcionamiento de los sensores y dispositivos de bajo coste para adquirir el conocimiento necesario para su programación y configuración. Apartado 6.2

    2. Implementar la comunicación entre dispositivos con ROS:

       Se utilizará ROS como plataforma de comunicación para establecer la conexión entre los dispositivos y el robot remoto. Se desarrollarán los mecanismos de comunicación necesarios para enviar y recibir datos entre los dispositivos y el robot. Apartado 6.2

    3. Desarrollar una aplicación para detección y/o reconocimiento de movimiento:

       Se emplearán técnicas de procesamiento e interpretación de señales y datos para detectar cambios de orientación y/o posición capturados por los sensores. Apartado 6.3

4 Hardware y software

En este apartado, se presentarán tanto el hardware como el software utilizados en el desarrollo del proyecto, es decir los componentes físicos utilizados, las herramientas informáticas y los programas utilizados para implementar y ejecutar el sistema.

A continuación, se detallarán tanto el hardware como el software clave utilizados en el proyecto. Esto incluirá los entornos de desarrollo, las bibliotecas y las plataformas de software utilizadas, junto con una breve explicación de su función y su contribución al proyecto. En cuanto al hardware, se detallarán los componentes físicos empleados, como sensores, microcontroladores,etc. Esta información proporcionará una visión general de los recursos utilizados.

4.1 Hardware

4.1.1 Robot UR5

Figura 4.1: Robot UR5 (PELEGRÍ, 2022)

El robot UR5 es un robot colaborativo o *cobot* de la marca Universal Robots, se trata de un robot ligero, versátil y adaptable. Tiene una carga útil de 5kg y un radio de acción de 850mm que lo convierten en uno de los robots más polivalentes a la hora de automatizar tareas.

Al igual que sus hermanos UR3, UR10, UR16 y UR20 destaca su facilidad de programación y manejo, su ligereza y su tamaño compacto. Esto hace del UR5 una opción rápida, flexible y asequible para el mundo de la automatización con robots colaborativos.

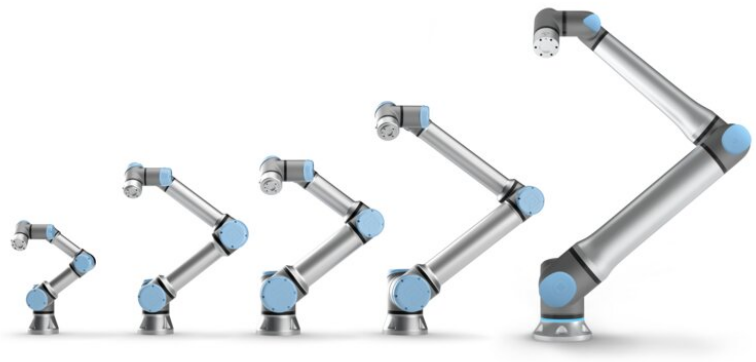

Figura 4.2: Universal Robots

Otro aspecto destacable es la alta capacidad de reconfiguración que permite a las pequeñas y medianas empresas adaptarse a los cambios en la cadena de producción sin la necesidad de una alta inversión de capital.

Sus funciones de protección también es uno de sus puntos fuertes, pues posee un sistema de seguridad integrado por 17 funciones entre las que destaca la capacidad de operar a una menos velocidad ante la cercanía de un operario, o la capacidad de detenerse en caso de detectar una colisión. Gracias a esto los robots de Universal Robots son capaces de operar sin la necesidad de vallados de seguridad, lo que supondría un aumento de los costes en la integración del robot a la cadena de producción.

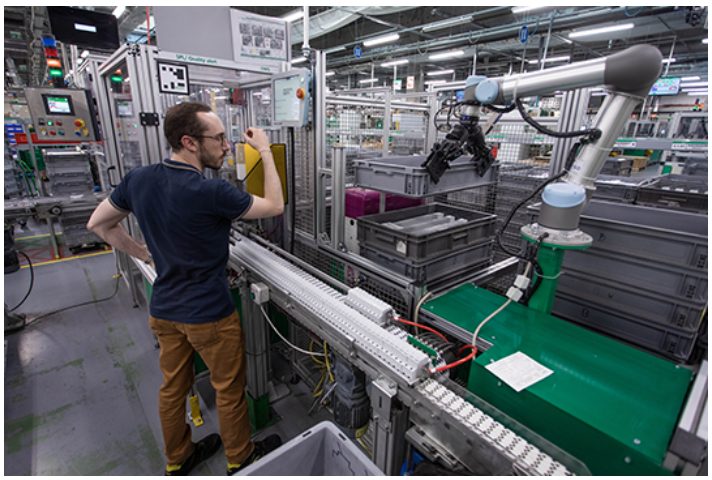

Figura 4.3: UR5 sin vallado de seguridad (PELEGRÍ, 2022)

En cuanto a su cinemática, se caracteriza por su brazo robótico de 6 grados de libertad, lo que le permite un amplio rango de movimiento en su operación. Cada una de sus articulaciones posee un rango de trabajo específico, lo que determina los límites físicos de su desplazamiento. Es importante tener en cuenta estas restricciones cinemáticas al teleoperar el robot, ya que cualquier movimiento deseado debe estar dentro de los límites establecidos por su diseño.

Su espacio de trabajo se define por la combinación de los rangos de trabajo de cada una de sus articulaciones. Esto determina el volumen en el cual el robot puede moverse sin obstáculos, teniendo en cuenta las restricciones impuestas por su estructura mecánica. Es fundamental tener en consideración el espacio de trabajo del robot al realizar operaciones de teleoperación, para evitar colisiones o movimientos fuera de su capacidad.

Además, es importante mencionar las limitaciones de velocidad y precisión. Aunque es un robot colaborativo, su diseño está optimizado para realizar tareas industriales con alta precisión y repetibilidad. Sin embargo, es necesario considerar estas limitaciones al teleoperar el robot, ya que los movimientos en tiempo real pueden verse afectados por diversos factores como la latencia de la comunicación o la capacidad de respuesta del controlador.

4.1.2 Arduino

Figura 4.4: Arduino logo

Arduino es una plataforma de electrónica de código abierto que está diseñada para facilitar la creación de proyectos interactivos. Consiste en una placa de hardware y un entorno de desarrollo de software que permite programar y controlar diferentes componentes electrónicos de manera sencilla.

La placa Arduino se basa en microcontroladores, pequeños chips programables con una Unidad Central de Procesamiento (CPU), memoria y puertos de entrada y salida. Estos microcontroladores son el cerebro de la placa Arduino que se encargan de ejecutar los programas desarrollados en su entorno Arduino IDE. Se trata de software libre, pues permite a cualquiera utilizar, crear y modificar aplicaciones para las placas Arduino.

Al tratarse también de hardware libre cualquier persona o empresa puede replicarlo, es decir Arduino permite que otros puedan crear sus propias placas, diferentes entre ellas pero manteniendo la misma base en cuanto a funcionalidad. Es por esto que Arduino se entiende más como un proyecto y no como un modelo concreto de placa y, es posible encontrar en el mercado una gran variedad de tipos diferentes de placas, con mayores o menores prestaciones, lo que permite ajustar la elección de la placa a las necesidades del proyecto a desarrollar.

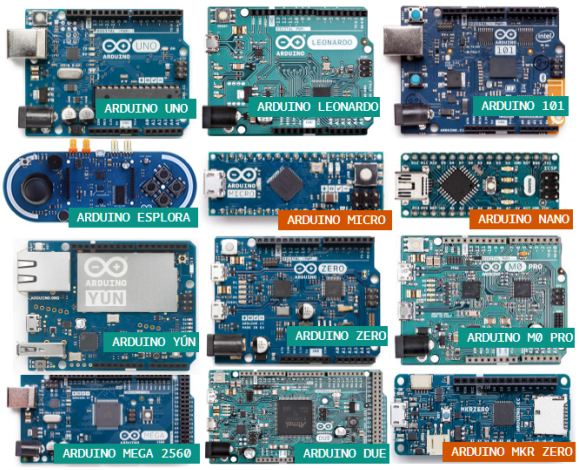

Figura 4.5: Diferentes tipos de placas (FERNÁNDEZ, s.f.)

Arduino Uno	30€	Arduino Leonardo	25€
Arduino 101	42€	Arduino Esplora	28€
Arduino Micro	23€	Arduino Nano	26€
Arduino Yún	24€	Arduino Zero	30€
Arduino M0 Pro	45€	Arduino Mega 2560	50€
Arduino Due	50€	Arduino Mkr Zero	47€

Tabla 4.1: Precios aproximados de placas Arduino en Amazon

Para el caso concreto de este proyecto se ha decidido emplear una placa Arduino UNO, se

trata de una de las placas de menor coste pero con suficiente capacidad de procesamiento para este proyecto.

Figura 4.6: Arduino UNO

4.1.3 Sensores de bajo coste

Otro elemento fundamental para el desarrolo del proyecto son los sensores/accionadores de bajo coste, necesarios para poder capturar valores con los que generar ordenes para ejecutar movimientos.

4.1.3.1 IMU

Una IMU es un dispositivo que combina varios sensores, acelerómetro, giroscopio y, en ocasiones, magnetómetro. Permiten obtener información sobre orientación, aceleración lineal y velocidad angular.

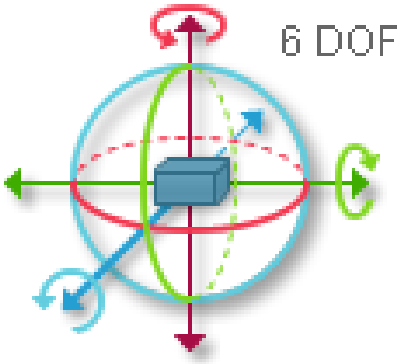

Figura 4.7: Grados de libertad de una IMU

El acelerómetro mide las aceleraciones lineales en los tres ejes (X, Y y Z), permitiendo determinar cambios de velocidad y aceleración. El giroscopio, por otro lado, mide la velocidad angular, el cambio en el ángulo rotacional por unidad de tiempo.

Para Arduino esta disponible la IMU MPU6050, que incorpora acelerómetro y giroscopio, pero no magnetómetro.

Figura 4.8: Mpu6050

En este proyecto se utilizará este dispositivo para obtener valores de orientación que sirvan de variable de control para lograr la teleoperación del robot.

4.1.3.2 Joystick

Otro elemento de bajo coste utilizado como accionador para obtener más variables de control es un joystick, un dispositivo de entrada que consiste en una palanca que se puede mover en diferentes direcciones acompañado de un botón. Este movimiento se transmite a través de sensores como potenciómetros que detectan la posición y envían las señales eléctricas al dispositivo, en este caso, a Arduino.

El joystick utilizado en este proyecto, en su configuración estándar, puede detectar movimientos en cuatro direcciones principales: arriba, abajo, izquierda y derecha. Estos movimientos direccionales permiten controlar variables en el proyecto.

Además de estas cuatro direcciones principales, el joystick también permite movimientos diagonales, lo que significa que se pueden codificar ocho direcciones en total. Sin embargo, para este proyecto tan solo será necesario emplear dos direcciones principales (arriba y abajo) y el botón que incorpora.(Llamas, 2016a)

Figura 4.9: Joystick para Arduino

4.2 Software

4.2.1 ROS

ROS es una plataforma de software de código abierto utilizada en robótica. A pesar de que se le llama "sistema operativo", no se trata de un sistema operativo tradicional como Windows, es más bien una colección de frameworks que facilitan el desarrollo de software para robots.

Figura 4.10: ROS logo

Proporciona una infraestructura robusta y flexible para la programación de robots que permite comunicar diferentes componentes del sistema. Está diseñado para ser modular y escalable, pudiéndose utilizar en una gran variedad de robots.

Una característica clave es su arquitectura de comunicación basada en mensajes. Emplea un sistema de intercambio de mensajes asíncrono que permite una comunicación eficiente facilitando el desarrollo de sistemas complejos, ya que cada elemento puede probarse por separad antes de ser integrado al sistema completo.

ROS también ofrece herramientas y bibliotecas para tareas comunes en robótica, como la percepción, la planificación de trayectorias, la simulación y la visualización.Estas herramientas permiten a los desarrolladores aprovechar funcionalidades ya existentes y acelerar el proceso de desarrollo de aplicaciones robóticas.

Es ampliamente utilizado en investigación y desarrollo de robots debido a su naturaleza de código abierto y su enfoque en la reutilización de software. Existe una gran cantidad de recursos, documentación y comunidades activas que respaldan y promueven el uso de ROS.(Delgado, 2017)

Para este proyecto se ha empleado la version de ROS Noetic, ya que es la que se encuentra disponible para el sistema operativo que se está empleando (Ubuntu 20.4). Su instalación en muy sencilla, una vez actualizados los paquetes, se puede instalar mediante una solo instrucción en la consola del sistema.

Código 4.1: Instalación de ROS Noetic

```
1        sudo apt install ros-noetic-desktop-full
```

Se puede consultar la página oficial http://wiki.ros.org/noetic/Installation/Ubuntu para llevar a cabo la instalación.

4.2.2 Gazebo

Gazebo es un simulador de código abierto ampliamente utilizado en robótica y sistemas autónomos. Proporciona un entorno de simulación 3D que permite a los desarrolladores probar y validar algoritmos, controladores y sistemas robóticos antes de implementarlos en hardware real.

Figura 4.11: Gazebo logo

Gazebo ofrece una amplia gama de características y funcionalidades para la simulación pre-

cisa de robots y entornos virtuales. Permite modelar objetos físicos, como robots, sensores, actores y entornos, con detalles realistas, incluyendo propiedades físicas como masa, fricción y colisiones. Además, proporciona motores de física de alto rendimiento que calculan interacciones realistas entre los objetos simulados.

Además de la simulación de física, Gazebo también permite simular sensores como cámaras, láseres y otros dispositivos de percepción utilizados en robótica. Esto permite a los desarrolladores probar algoritmos de percepción y control utilizando datos simulados antes de implementarlos en hardware real.

La flexibilidad y las capacidades de simulación de Gazebo lo convierten en una herramienta popular en la comunidad de robótica. Es utilizado tanto en investigación académica como en desarrollo industrial para acelerar el proceso de diseño, depuración y validación de sistemas robóticos antes de la implementación física.(*Capítulo 3. Entorno de simulación ROS/Gazebo*, s.f.)

Si a la hora de instalar ROS se ha decidido realizar la instalación completa, Gazebo viene instalado por defecto. En caso de no haber seleccionado la intalación completa es posible instalarlo mediante la siguiente instrucción en la consola del sistema.

Código 4.2: Instalación de Gazebo para Noetic

```
1 sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-ros-control
```

Se puede consultar https://classic.gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros como guía para la instalación.

4.2.3 IDE Arduino

El Entorno de Desarrollo Integrado de Arduino, es una aplicación de software que proporciona un entorno de programación específico para el desarrollo de proyectos utilizando la plataforma Arduino. Es una herramienta fundamental para escribir, compilar y cargar código en las placas Arduino.

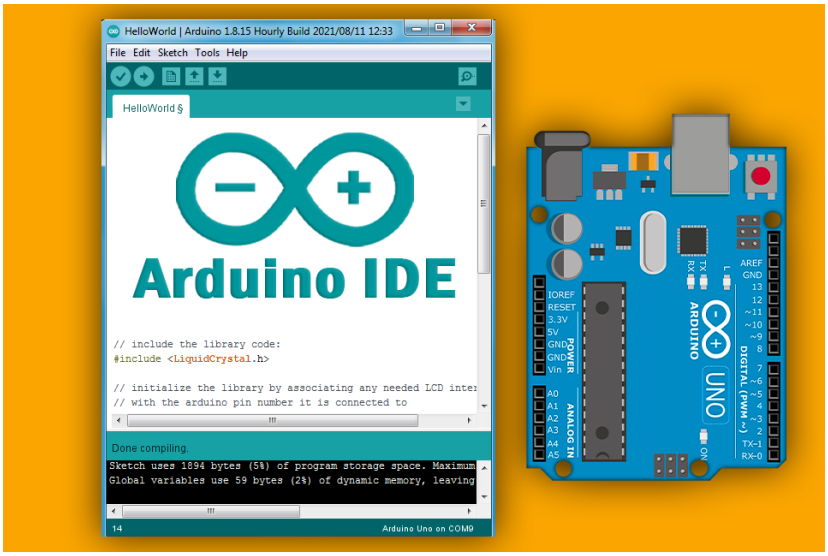

Figura 4.12: IDE Arduino (AndProf, 2023)

Una de las características destacadas del IDE de Arduino es su capacidad de trabajar con una amplia variedad de placas Arduino, lo que permite a los usuarios seleccionar y configurar fácilmente la placa específica con la que están trabajando.(CEAC, 2018)

Para su instalación se puede seguir el tutorial que se encuentra en https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE.

4.2.4 Librerías clave utilizadas

4.2.4.1 ros.h

Se trata de una biblioteca específica de Arduino que permite la comunicación entre una placa Arduino y el sistema de comunicación de ROS. Proporciona las funciones y estructuras necesarias para establecer la conexión y la comunicación bidireccional entre una placa Arduino y un sistema ROS. Permite que la placa Arduino envíe y reciba mensajes y comandos a través del sistema de comunicación de ROS.

Utilizando esta librería es posible configurar una placa Arduino para generar un nodo ROS, lo que le permite interactuar con otros nodos y componentes de ROS. Permite enviar datos desde la placa Arduino a ROS, como información de sensores o estado de actuadores.

Este recurso ha sido necesario para conseguir enviar los datos obtenidos con Arduino a ROS empleando mensajes propios de este sistema.
En el siguiente enlace se encuentra una guía para la instalación y configuración de esta libreria http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup.

4.2.5 roboticstoolbox para Python

Consiste en un conjunto de herramientas y funciones para el análisis, simulación y control de robots en entornos robóticos. Está diseñada para facilitar el desarrollo de algoritmos y aplicaciones relacionados con la robótica, en este caso, utilizando el lenguaje de programación Python.

Se basa en la Robotics Toolbox original desarrollada en MATLAB. La versión para Python busca ofrecer características y funcionalidades similares, permitiendo a los usuarios aprovechar las capacidades de Python para el desarrollo de aplicaciones robóticas.

Ofrece funcionalidades como la representación de robots, cálculos de cinemática directa e inversa, cálculo de dinámica, control de robots y simulación.

4.2.6 Otros recursos externos

4.2.6.1 ROS-Industrial Universal Robot meta-package

Se trata de un repositorio de GitHub que proporciona las herramientas para simular y controlar los robots de Universal Robots en el entorno de simulación Gazebo. Incluye archivos de descripción del robot, controladores y otros recursos necesarios para establecer la conexión y la comunicación entre Gazebo, ROS y los robots de Universal Robots.

Para este proyecto ha sido un punto clave, pues ha permitido generar la simulación del robot UR5 y configurar los controladores virtuales necesarios junto con sus respectivos *topics*. Esto ha facilitado enormemente el proceso de desarrollo y prueba del proyecto, ya que se ha podido trabajar en un entorno simulado que emula con precisión el comportamiento del robot real.

5 Metodología

En esta sección, se describe el enfoque y los procedimientos utilizados para llevar a cabo el trabajo, destacando los métodos y técnicas empleados. El problema que se busca resolver es conseguir teleoperar un robot con dispositivos de bajo coste empleando un control cartesiano. Esto implica la capacidad de controlar los movimientos y orientaciones del robot en los ejes X, Y y Z, así como las rotaciones alrededor de dichos ejes. De esta forma es posible realizar movimientos lineales, desplazamientos en el espacio tridimensional y rotaciones en cualquier dirección. Para ello el método seguido se puede dividir en 3 grandes apartados:

1. Adquisición de datos de la IMU y el joystick en Arduino

2. Comunicación de Arduino con ROS

3. Configuración del nodo de teleoperación en Python

5.1 Adquisición de datos de la IMU y el joystick en Arduino

5.1.1 Aproximación 1

Inicialmente se consideró la posibilidad de utilizar dos IMUs para asociar directamente las 6 variables que generan ambos dispositivos a los 6 grados de libertad que tiene el robot. Este enfoque permitiría establecer una correspondencia directa entre cada grado de libertad del robot y una variable de control proporcionada por las IMUs. De esta forma sería posible teleoperar de forma simultánea tanto posición como orientación en todo el espacio de trabajo del robot.

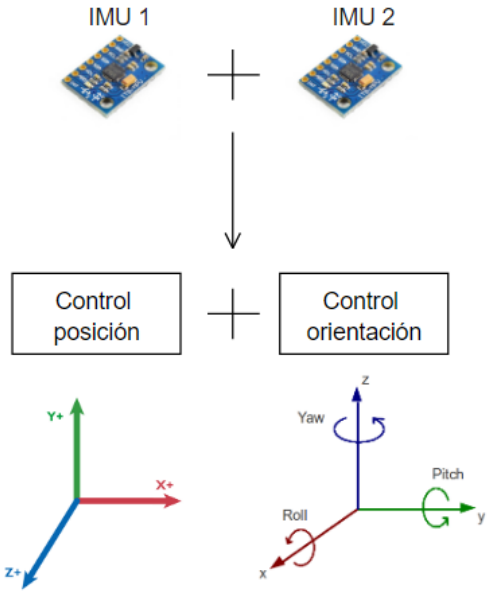

Figura 5.1: Esquema aproximación 1

5.1.2 Aproximación 2

La segunda opción para la adquisición de datos trata de emplear una IMU que controle los giros y desplazamientos en X e Y y un joystick para controlar el giro y desplazamiento en Z. El joystick incluye también un botón que permite añadir una funcionalidad extra, al ser pulsado alterna de modo de control (posición u orientación) lo que permite teleoperar el robot con solo 2 variables de la IMU y 1 del joystick. Esto sería similar al funcionamiento que tienen las consolas de programación que ofrecen los fabricantes de robots, controlando la posición mediante incrementos positivos o negativos en cada eje.

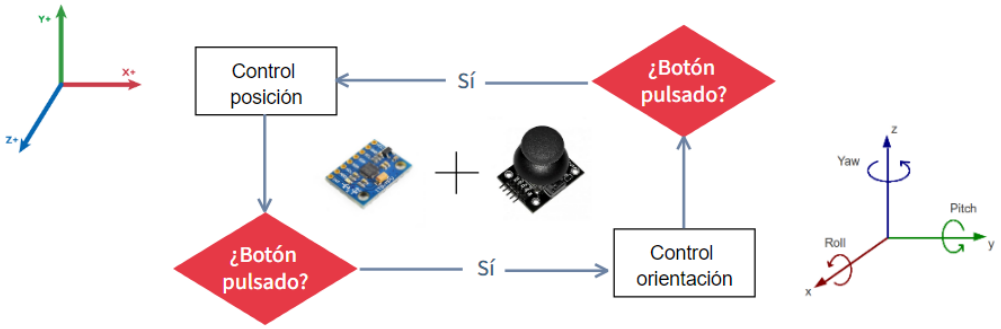

Figura 5.2: Esquema aproximación 2

5.1.3 Propuesta final

Para obtener una medida precisa de los ángulos es necesario combinar las medidas del acelerómetro con las del giroscopio, aplicando un filtro complementario que reduce el ruido en las medidas del acelerómetro y la deriva en las medidas del giroscopio. Para combinar los ángulos α que se obtienen del acelerómetro y del giroscopio se aplica un factor de ponderación β que determina la contribución relativa de cada medida en el cálculo del ángulo final.

$$\alpha\_final = \beta \cdot \alpha\_giroscopio + (1 - \beta) \cdot \alpha\_acelermetro \tag{5.1}$$

Sin embargo, esto no es posible para el giro en el eje Z, ya que este no se ve afectado por la fuerza de la gravedad que es utilizada para obtener las medidas con el acelerómetro. Es por esto que no es posible obtener el giro en Z de forma precisa para ser utilizado en la teleoperación.

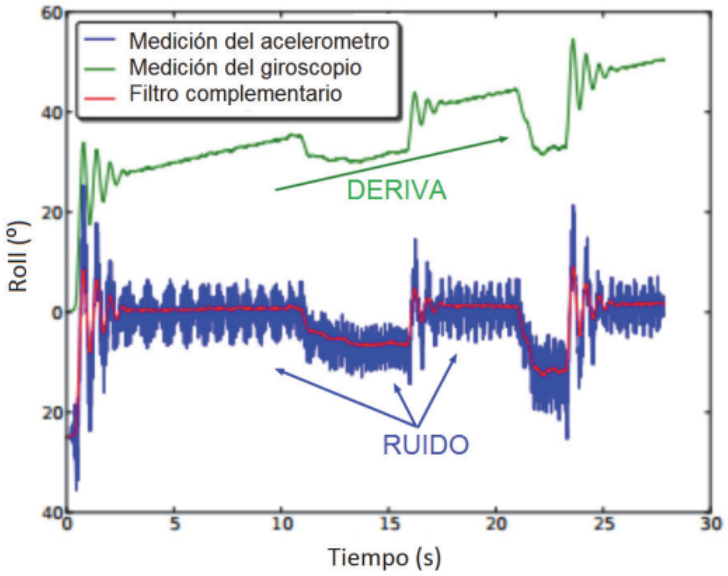

Figura 5.3: Filtro complementario (Apuntes asignatura Sensores)

Esto limita las IMUs a solo 2 variables útiles para la teleoperación, por lo que se descartó este planteamiento inicial (Apartado 5.1.1) y se optó por la emplear el segundo planteamiento (Apartado 5.1.2).

Este enfoque basado en incrementos proporciona una forma conveniente de controlar la posición del robot, ya que el usuario puede ajustar gradualmente las coordenadas de posición y observar los cambios en tiempo real. Además, al utilizar incrementos, se evitan comandos abruptos que podrían generar movimientos bruscos o inesperados en el robot.

5.2 Comunicación de Arduino con ROS

Para lograr la comunicación entre Arduino y ROS se ha empleado la librería *ros.h* que permite crear *nodos* y acceder a *topics* desde Arduino. Se ha creado un nodo *publisher* y el *topic* 'imu' donde se publica el mensaje 'imu_msg' de tipo *String*. Se ha empleado este tipo de variable ya que consume pocos recursos y es fácil de usar para codificar variables desde el código en Arduino y decodificarlas desde el código en ROS. Las variables codificadas y enviadas en esta cadena de caracteres serán los ángulos en los ejes X e Y obtenidos mediante la IMU, el valor del movimiento en el eje Y del joystick y el valor del botón del joystick. Con esto será suficiente para lograr la teleoperación empleando la segunda aproximación (Apartado 5.2).

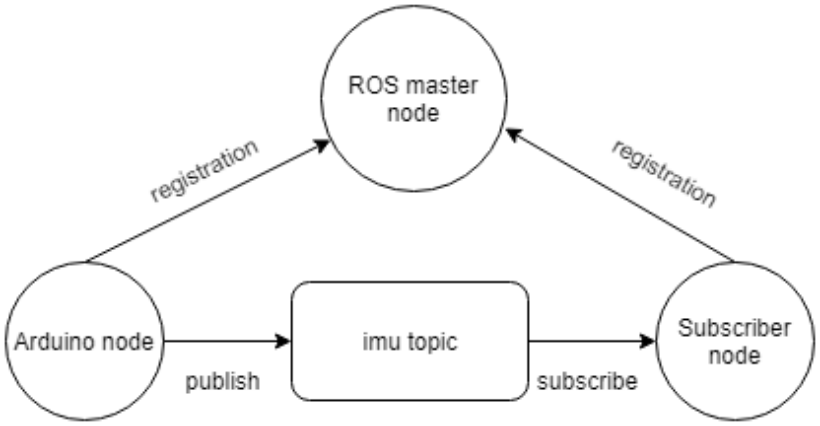

Figura 5.4: Comunicación (Said, 2019)

5.3 Configuración del nodo de teleoperación en Python

Una vez se tienen los datos en el código en ROS ya solo queda convertir los valores de los ángulos y el joystick a variables articulares para ser enviadas al robot. Se explicará el proceso brevemente y se profundizará más en el capítulo 6, concretamente en el apartado 6.3.

Dado que se busca un control cartesiano mediante incrementos, en primer lugar se ajustan los datos a los incrementos que se utilizarán para controlar las variables de posición y orientación.

En el caso de la IMU, los ángulos se utilizan como base para calcular los incrementos de posición y orientación. Dependiendo de la sensibilidad deseada y las características del robot, se establecen los valores de incremento para cada eje.

En cuanto al joystick, los movimientos realizados por el usuario se traducen en incrementos, los cuales se ajustan según los límites y la precisión requerida. Como se ha comentado en la sección anterior, al pulsar el botón del joystick se realiza un cambio de modo que permite alternar entre el control de posición y el control de orientación.

Una vez se obtienen los datos de posición y orientación del sistema teleoperado, se realiza el cálculo de la cinemática inversa. La cinemática inversa es un proceso matemático que permite determinar las variables articulares necesarias para lograr una determinada configuración espacial del robot. En otras palabras, se utiliza para calcular los ángulos y posiciones de las articulaciones del robot en función de la posición y orientación deseadas.

Una vez obtenidas las variables articulares finales a partir de la cinemática inversa, se envían a través del correspondiente *topic* en el entorno ROS. En este caso, se publican los valores de las variables articulares finales en el *topic* correspondiente, para que el controlador del robot los reciba y los utilice para mover las articulaciones en la posición requerida.

Al enviar los valores de las variables articulares finales al *topic*, se activa el controlador del robot, el cual interpreta y ejecuta los comandos recibidos. El controlador se encarga de actuar sobre los motores y las articulaciones del robot, siguiendo los valores articulares finales proporcionados. De esta manera, el robot se mueve y se posiciona de acuerdo con los datos recibidos, permitiendo que los movimientos del sistema teleoperado se reflejen en el robot real o virtual.

6 Desarrollo

En este apartado se describirá en detalle el proceso seguido en el proyecto para lograr el control teleoperado del robot. A lo largo de esta sección, se abordarán los pasos clave y las implementaciones realizadas, centrándose en los aspectos técnicos y las decisiones tomadas para alcanzar los objetivos propuestos.

Se explicará el diseño del hardware empleado, incluyendo la disposición y conexión de los componentes relevantes, como la IMU y el joystick, así como su integración con Arduino. Se mostrarán esquemas o diagramas que ilustren la conexión de los elementos.

El desarrollo del firmware de Arduino será abordado en detalle, explicando cómo se programaron los dispositivos para adquirir los datos de la IMU y el joystick, y cómo se utilizó la librería ros.h para enviar estos datos a través de mensajes ROS. Se incluirán fragmentos de código relevante y se proporcionará una explicación de su funcionamiento.

En paralelo, se describirá el desarrollo del nodo ROS en Python, que recibirá los datos enviados por Arduino a través de ROS. Se explicará cómo se realizan los cálculos y procesamientos necesarios para obtener las variables articulares finales, así como la publicación de dichas variables en el correspondiente *topic*.

Un aspecto fundamental del proyecto será la implementación de la cinemática inversa, que permitirá convertir los datos de posición y orientación en valores articulares para controlar el robot. Se detallará el algoritmo utilizado para realizar estos cálculos.

Finalmente, se describirá la integración con el controlador del robot, explicando cómo se establece la comunicación y se envían las variables articulares finales para lograr el movimiento del robot. Se mencionarán posibles configuraciones o ajustes necesarios para garantizar una interacción correcta entre el sistema teleoperado y el robot.

6.1 Diseño del hardware

Las conexiones del dispositivo son relativamente sencillas, en el caso de la IMU se conectan a 2 entradas analógicas como se puede observar en la siguiente imagen.

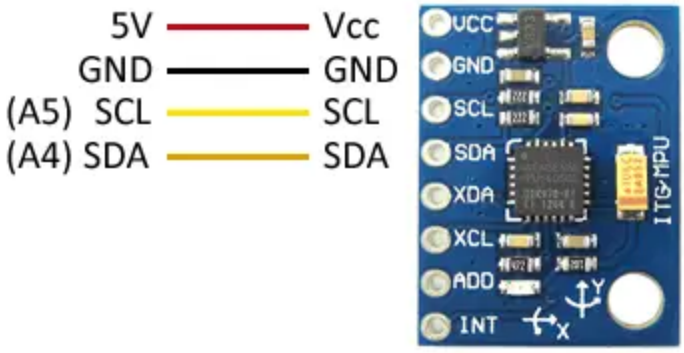

Figura 6.1: Conexiones IMU

Para la conexión del Joystick a la placa se pueden seguir las siguientes indicaciones. Conectando a 2 entradas analógicas, correspondientes a los ejes, y a una digital correspondiente al botón.

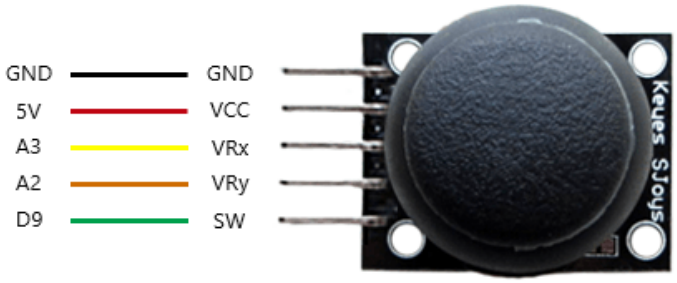

Figura 6.2: Conexiones joystick

En el caso de las conexiones de ambos dispositivos, es importante mencionar que además de las entradas analógicas y digitales, también se deben conectar las líneas de suministro de energía y tierra (VCC y GND) para asegurar un funcionamiento adecuado. Estas conexiones son necesarias para proporcionar la alimentación necesaria a los componentes y establecer una referencia común de tierra entre ellos.

6.2 Programación del firmware de Arduino

Una vez las conexiones están realizadas se procede a la lectura de datos de los sensores.Para ello se obtienen los valores de los registros correspondientes, estos pueden consultarse en el mapa de registros.Estos valores se procesan para convertirlos a unidades del sistema internacional y posteriormente se aplica un filtro complementario para obtener las rotaciones y reducir el error que se pueda generar en la lectura, ya que el acelerómetro tiene mucho ruido y el giroscopio tiene una deriva en la medida como ya se comentó previamente en el apartado

5.1.3 donde se mostró una gráfica al respecto (Imagen 5.3).(InvenSense, 2013)

El filtro complementario realiza una combinación ponderada de las medidas de ambos dispositivos, teniendo en cuenta sus características y proporcionando una salida más confiable y precisa. Para lograr esto, se utilizan operaciones simples que consumen pocos recursos, lo cual es especialmente importante al trabajar con microcontroladores de bajo coste, como en el caso de este proyecto. Utiliza filtros digitales paso-bajo para el acelerómetro y paso-alto para el giroscopio.

Las fórmulas empleadas para convertir los valores de aceleración y velocidad angular en ángulos que puedan ser combinados con el filtro son las siguientes.

$$\theta_x = \tan^{-1} \cdot \frac{a_x}{\sqrt{a_y^2+a_z^2}} \tag{6.1}$$

$$\theta_y = \tan^{-1} \cdot \frac{a_y}{\sqrt{a_x^2+a_z^2}} \tag{6.2}$$

Donde:

- $\theta_x \rightarrow$ Es el ángulo de giro en el eje X.
- $\theta_y \rightarrow$ Es el ángulo de giro en el eje Y.
- $a_x \rightarrow$ Es el valor de aceleración en el eje X.
- $a_y \rightarrow$ Es el valor de aceleración en el eje Y.
- $a_z \rightarrow$ Es el valor de aceleración en el eje Z.

$$\theta_x = \theta_{x0} + \omega_x \cdot \triangle t \tag{6.3}$$

$$\theta_y = \theta_{y0} + \omega_y \cdot \triangle t \tag{6.4}$$

Donde:

- $\theta_x \rightarrow$ Es el ángulo de giro en el eje X.
- $\theta_y \rightarrow$ Es el ángulo de giro en el eje Y.
- $\theta_{x0} \rightarrow$ Es el valor previo del ángulo de giro en el eje X.
- $\theta_{y0} \rightarrow$ Es el valor previo del ángulo de giro en el eje Y.
- $\omega_x \rightarrow$ Es el valor de la velocidad angular en el eje X.
- $\omega_y \rightarrow$ Es el valor de la velocidad angular en el eje Y.
- $\triangle t \rightarrow$ Es el valor del incremento de tiempo.

Una vez obtenidos los valores necesarios para la teleoperación, tanto de la IMU como del joystick, se deben comunicar estos datos a ROS. Para ello, como ya se ha comentado anteriormente, se ha empleado la librería *ros.h* que permite crear *nodos* y acceder a *topics* desde Arduino. Se ha creado un nodo *publisher* y el *topic* 'imu' donde se publica el mensaje 'imu_msg' de tipo *String*.

Para el envío del mensaje se ha empleado una cadena de caracteres, concatenando las diferentes variables separadas por letras como se muestra a continuación.

Código 6.1: Codificación mensaje en C++

```
String data = "A" + AX + "B"+ AY + "C" + Joy + "D" + But + "E";
```

Se ha empleado esta forma ya que consume pocos recursos del microcontrolador y es muy sencilla de decodificar mediante código en *python*, solo es necesario seleccionar el fragmento de la cadena que se desea guardar en cada variable. Para ello se indica el índice de las letras empleadas en la codificación como se observa a continuación.

Código 6.2: Decodificación mensaje en Python

```
Ax = float(imu.data[imu.data.index('A')+1:imu.data.index('B')])
```

De esta sencilla forma es posible almacenar cada valor en una variable para poder continuar con la gestión de los datos desde el nodo de ROS encargado de la teleoperación.

6.3 Desarrollo del nodo ROS en Python

Una vez la información ha sido recibida a través del microcontrolador, procesada para enviarse y publicada en su *topic* correspondiente, se accede a ella desde otro nodo ROS. El objetivo principal de esta etapa es llevar a cabo el procesamiento final de los datos físicos para obtener las correspondientes velocidades articulares que se enviarán al robot.

Un aspecto que es necesario mencionar son las relaciones que existen entre los movimientos del dispositivo y los movimientos del robot. Durante la teleoperación en modo posición, los ángulos medidos por la IMU controlan los desplazamientos en los ejes X e Y del robot. Por ejemplo, si la IMU detecta una inclinación positiva en el eje X, el robot se desplazará en la dirección positiva del eje X. Del mismo modo, una inclinación negativa en el eje X hará que el robot se desplace en la dirección negativa del eje X. Lo mismo ocurre con el eje Y. Asimismo, cuando se utiliza el joystick, si se mueve en su dirección Y positiva, el robot se desplazará en el eje Z positivo, mientras que si se mueve en su dirección Y negativa, el robot se desplazará en el eje Z negativo. Todos estos movimientos se realizan desde el sistema de referencia de la base del robot. Por otro lado, en el modo de teleoperación de orientación, se utiliza un enfoque similar, donde los valores de inclinación en los ejes X e Y de la IMU controlan la orientación en los ejes X e Y de la herramienta del robot, mientras que el joystick controla la orientación en el eje Z.

En este nodo destacan 3 partes diferenciadas. Una encarga de la decodificación de los datos y la conversión de estos datos a incrementos en la posición y orientación; otra que se encarga de realizar la cinemática inversa; y, por último, una que realiza la publicación de las posiciones articulares en el *topic* correspondiente del robot.

En la primera parte, se decodifican los datos como se ha mencionado antes y aplican factores establecidos como variables globales que ajustan los ángulos recibidos a incrementos en posición y orientación. Destacar que antes se aplican offsets ya que al ser dispositivos de bajo coste tienen errores en la medida y, a pesar de situar la IMU en reposo, devuelve una pequeña inclinación.

Una vez obtenidos los incrementos, se comprueba si se ha pulsado el botón, es decir, se comprueba si se quiere controlar la posición o la orientación y se modifican en cada momento las variables adecuadas.

En la segunda parte, se recibe la información de posición y orientación anterior y, empleando la librería Roboticstoolbox se realiza la cinemática inversa, obteniendo de esta forma las posiciones articulares correspondientes a la pose recibida. Destacar que para el correcto funcionamiento de esta parte es necesario indicar la posición articular en la que se encuentra el robot a la hora de realizar el cálculo de la cinemática inversa, para conseguir un movimiento final fluido y lógico.

Código 6.3: Ejemplo de uso para la función de la cinemática inversa

```
1    Joints_value = robot.ik_LM(Cartesian_Pose, q0=q0)
```

Siendo q_0 el valor de la posición articular anterior. La clase *robot* debe ser creada anteriormente cargando el modelo que quiera emplearse. En este caso el UR5, se haría mediante el siguiente código.

Código 6.4: Cargar modelo del robot UR5

```
1    robot = rtb.models.UR5()
```

En la última parte, una vez configurada la parte del *publisher*, se envían las posiciones articulares al *topic* correspondiente generado por la simulación del robot en Gazebo.

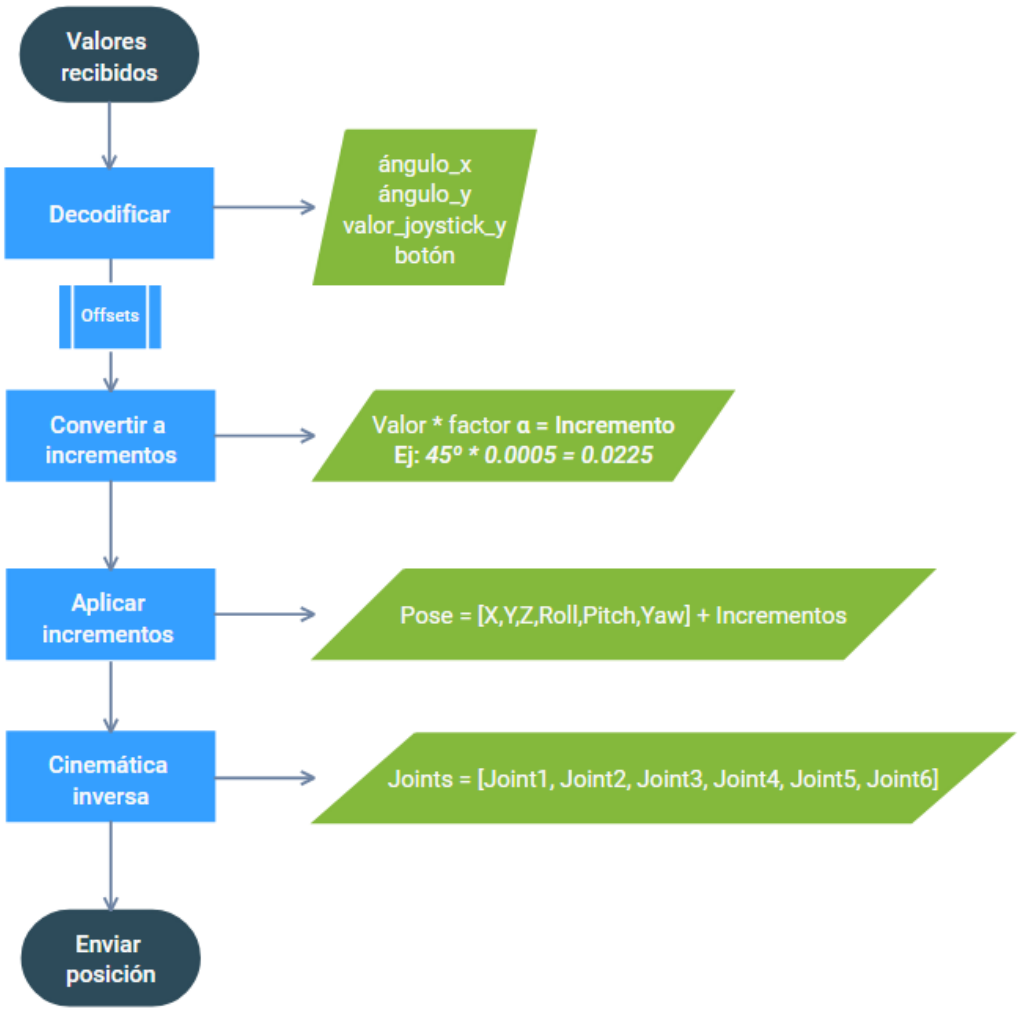

Figura 6.3: Diagrama de flujo

7 Resultado

En esta sección se explicará como ejecutar el proyecto y el resultado final obtenido.

7.1 Ejecución

A continuación se detallan los pasos a seguir para ejecutar el proyecto. Los códigos desarrollados se encuentran subidos a un Repositorio de GitHub.

1. Subir el programa de Arduino a la placa: En primer lugar, se debe cargar el programa de Arduino en la placa. Esto se realiza utilizando el entorno de desarrollo de Arduino, donde se selecciona la placa correspondiente y se elige el puerto de comunicación adecuado. Al subir el programa a la placa, se habilita la comunicación con ROS y se prepara para enviar los datos de la IMU y el Joystick.

Figura 7.1: Subir programa a la placa

2. Lanzar el .lauch de Gazebo del UR5: Una vez que el programa de Arduino está cargado en la placa, es necesario lanzar el archivo .launch proporcionado por Universal Robots en su repositorio. Este archivo configura el entorno de simulación en Gazebo y carga el modelo del robot UR5. Al lanzar este archivo, se crea una representación virtual del robot y se generan los *topics* de control en ROS. Estos *topics* permiten enviar comandos al robot y recibir información de su estado.

Código 7.1: Ejecutar script para lanzar la simulación

```
1        roslaunch ur_gazebo ur5_bringup.launch
```

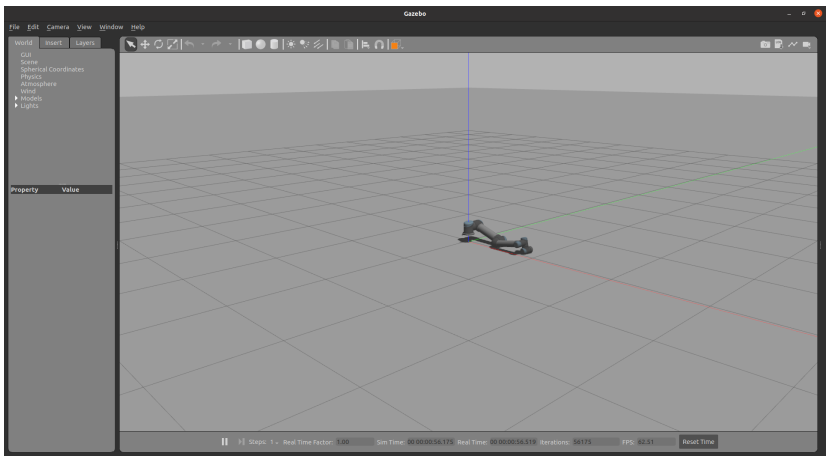

Figura 7.2: UR5 en Gazebo

3. Abrir la comunicación serie: Una vez que el programa de Arduino está en funcionamiento y el lanzamiento de Gazebo del UR5 se ha realizado con éxito, se debe establecer la comunicación serie entre Arduino y ROS. Para ello se debe ejecutar la siguiente instrucción en la terminal.

Código 7.2: Abrir comunicación serie

```
1        rosrun rosserial_python serial_node.py /dev/ttyACM0
```

Es importante asegurarse de que el puerto "/dev/ttyACM0" sea el correcto y corresponda al puerto en el que está conectada la placa Arduino.

4. Ejecutar el código Python: Una vez que la comunicación serie está establecida, se debe ejecutar el código Python que se encarga de recibir los datos de la IMU y el Joystick, realizar los cálculos necesarios y enviar los comandos al robot UR5.

Código 7.3: Ejecutar script para la teleoperación

```
1        rosrun nombre_del_paquete cartesian_teleop.py
```

Al ejecutar este código, se inicia el procesamiento de los datos recibidos, se realiza la cinemática inversa y se envían los comandos adecuados al robot UR5 para controlar su posición. Destacar que al ejecutar el código el robot se coloca inicialmente en una posición obtenida de forma experimental lejos de límites articulares.

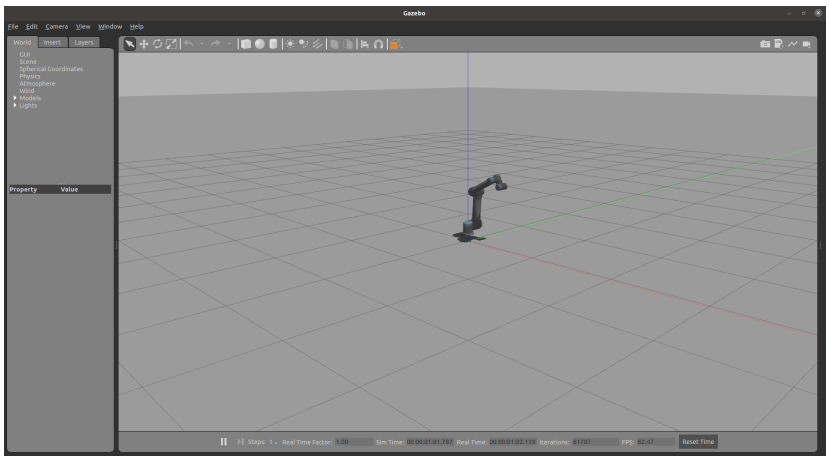

Figura 7.3: UR5 en posición inicial

Una vez realizados estos pasos ya es posible controlar el robot manipulador simulado en Gazebo mediante la IMU y el joystick.

7.2 Análisis movimiento simple

En este apartado, se presentan los resultados del análisis del movimiento simple realizado mediante el control teleoperado del robot. Se presentan las gráficas generadas a partir de los datos de ángulo en el eje X recibidos desde la IMU y la posición enviada al robot en el eje X.

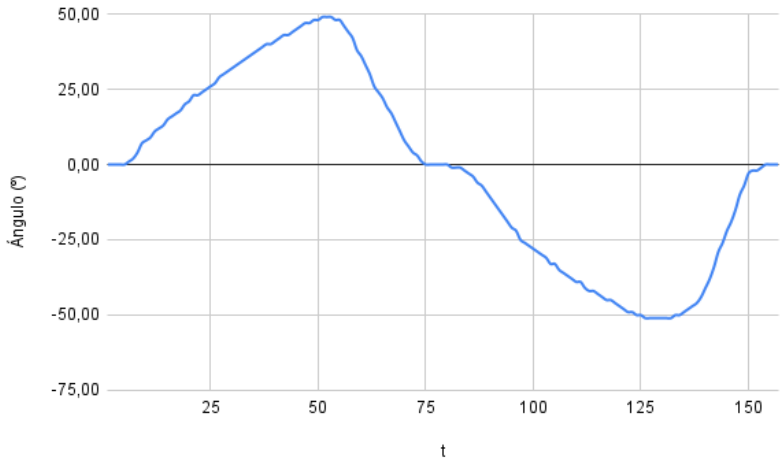

Figura 7.4: Gráfica de valores del ángulo en el eje X obtenidos con la IMU

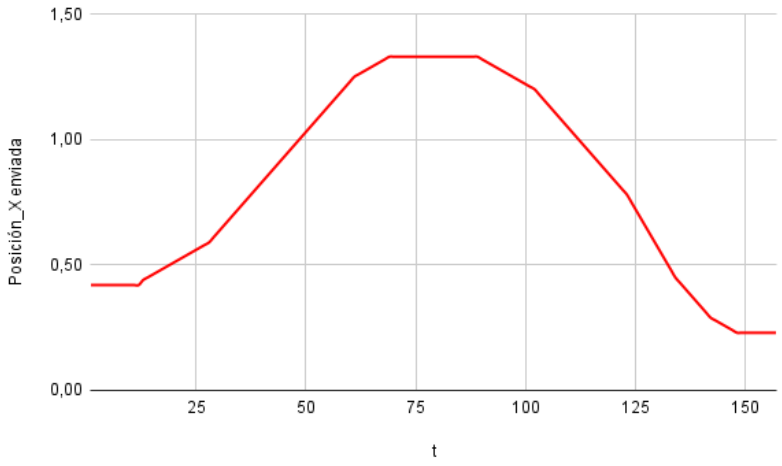

Figura 7.5: Gráfica de valores de posición X enviada

Se puede observar como a medida que aumenta el ángulo de inclinación de la IMU, aumenta la magnitud del incremento en la posición. Esto puede observarse en t=50, donde existe un máximo en el ángulo que se ve reflejado en la pendiente de la gráfica de posición, donde esta es máxima. Lo mismo ocurre con valores negativos.

Se puede apreciar que el aumento en la posición no es lineal con respecto al ángulo de inclinación, sino que existe una relación no lineal que muestra un incremento acelerado a medida que el ángulo aumenta. Esta característica permite ajustar la precisión en la generación del movimiento, es decir, es posible reducir la velocidad de movimiento y aumentar la precisión del proceso si se teleopera con pequeñas inclinaciones.

También se puede observar como al comenzar la inclinación no se produce un movimiento inmediato en la posición del robot. Esto se debe a que se estableció un umbral para evitar que valores pequeños de inclinación se traduzcan en incrementos mínimos en la posición. Esta decisión fue tomada de manera consciente debido a las posibles imprecisiones asociadas al uso de dispositivos de bajo coste utilizados en el sistema.Sin embargo, introducir este ”punto cero” implica un pequeño retraso en la respuesta inicial del sistema.

7.3 Vídeo demostración

A continuación se encuentra el enlace a un vídeo demostración del funcionamiento del dispositivo desarrollado durante este proyecto.

Figura 7.6: Pulsar para acceder al vídeo

8 Conclusiones

En este proyecto, se ha logrado desarrollar un sistema que permite controlar un robot UR5 mediante una combinación de una IMU y un joystick, utilizando la plataforma ROS y Arduino. El sistema ha demostrado ser efectivo en la captura de datos de orientación y posición, así como en el cálculo de las variables articulares necesarias para mover el robot.

El uso de la IMU y el joystick proporciona una interfaz intuitiva y flexible para controlar el robot. La integración de ROS y Arduino ha facilitado la comunicación y el intercambio de datos entre los componentes del sistema.

8.1 Comparación con consolas de programación

Al comparar el dispositivo desarrollado en este proyecto con las consolas Flex Pendant ofrecidas por los fabricantes de robots, se observan varias ventajas y diferencias significativas. En cuanto al costo, el dispositivo basado en Arduino y ROS resulta más económico en comparación con las consolas Flex Pendant, lo que lo hace más accesible para proyectos de menor presupuesto.

Comparativa de coste	
Dispositivo desarrollado	Flex Pendant ABB
Arduino UNO 30€ IMU 2€ Joystick 2€ Protoboard 5€	Dispositivo completo 2000€
TOTAL: 39€	TOTAL: 2000€

Tabla 8.1: Comparativa de coste del dispositivo desarrollado y Flex pendant ABB

En cuanto a la precisión, el sistema desarrollado ha demostrado ser capaz de capturar y procesar datos con una precisión adecuada para la teleoperación del robot UR5. Si bien no se cuenta con valores numéricos específicos, es importante tener en cuenta que al trabajar con dispositivos de bajo coste, es posible que existan limitaciones inherentes a dichos dispositivos que puedan afectar ligeramente la precisión de los movimientos. Sin embargo, se han minimizado al máximo estos errores mediante software, permitiendo obtener resultados satisfactorios en términos de control y movimiento del robot.

Es importante mencionar que, debido a la naturaleza de la teleoperación, el usuario no necesita tener una visualización precisa de la posición exacta a la que está moviendo el robot. El enfoque principal radica en permitir al usuario realizar los movimientos deseados de manera intuitiva, ajustando y refinando la posición a medida que sea necesario.

También es importante destacar que las consolas ofrecidas por los fabricantes pueden ofrecer funcionalidades más avanzadas y personalizadas específicamente para el robot en cuestión, como puede ser la programación de trayectorias y depuración de código.

8.2 Posibles ampliaciones

A pesar de los logros alcanzados en este proyecto, existen algunas áreas en las que se pueden realizar mejoras. Una posible mejora sería la incorporación del dispositivo en una pieza impresa en 3D, lo que permitiría una integración más robusta y estética del hardware.

Además, se podría considerar la adición de una pantalla LCD a Arduino, que proporcione información en tiempo real sobre el estado del robot y las acciones realizadas. Esto facilitaría la visualización y el seguimiento de las operaciones realizadas durante el control del robot.

8.3 Uso de software libre

Otro aspecto importante a destacar es la naturaleza de este proyecto, basado en software y hardware de código libre, que brinda la oportunidad a cualquier persona interesada de replicar o mejorar el sistema desarrollado. La disponibilidad de las herramientas utilizadas, como Arduino y ROS, en combinación con la documentación y recursos disponibles en línea, facilita el acceso y la comprensión de los conceptos necesarios para llevar a cabo proyectos similares.

La filosofía de código abierto fomenta la colaboración y el intercambio de conocimientos, lo que amplía las posibilidades de desarrollo y mejora de este tipo de sistemas. Además, al ser accesible para la comunidad, se fomenta la participación de diferentes personas con diferentes habilidades y experiencias, lo que puede llevar a nuevas ideas y soluciones innovadoras.

Bibliografía

Alomar, A. (2017). *Tutorial 23 esp8266 – obtener inclinación con mpu6050 (gy-521).* http://www.sinaptec.alomar.com.ar/2017/10/tutorial-23-esp8266-obtener-inclinacion.html. Descargado de http://www.sinaptec.alomar.com.ar/2017/10/tutorial-23-esp8266-obtener-inclinacion.html

Analusia Pamela Camacho Ríos, D. R. P. S. (2008). *Diseño de un sistema robótico teleoperado para fines didácticos.* Descargado de https://tesis.ipn.mx/bitstream/handle/123456789/185/camachorios.pdf?sequence=1&isAllowed=y

AndProf. (2023). https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/. Descargado de https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/

Capítulo 3. entorno de simulación ros/gazebo. (s.f.). Descargado de https://biblus.us.es/bibing/proyectos/abreproy/5139/fichero/PFC-+Por+cap%C3%ADtulos%252F3-Entorno+de+simulaci%C3%B3n.pdf

CEAC. (2018). *Qué es el ide de arduino en robótica.* https://www.ceac.es/blog/que-es-el-ide-de-arduino-en-robotica. Descargado de https://www.ceac.es/blog/que-es-el-ide-de-arduino-en-robotica

Corke, P. (s.f.). *Robotics toolbox.* https://petercorke.com/toolboxes/robotics-toolbox/. Descargado de https://petercorke.com/toolboxes/robotics-toolbox/

Delgado, D. O. (2017). *Qué es ros (robot operating system).* https://openwebinars.net/blog/que-es-ros/. Descargado de https://openwebinars.net/blog/que-es-ros/

Download and install arduino ide. (s.f.). https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE. Descargado de https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE

Emmanuel Nuño Ortega, L. B. V. (2004). *Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente.* Descargado de https://upcommons.upc.edu/bitstream/handle/2117/570/IOC-DT-P-2004-05.pdf

Erlerobotics. (s.f.). *Creado paquetes de ros.* https://erlerobotics.gitbooks.io/erlerobot/content/es/ros/tutorials/creating_a_ros_package.html. Descargado de https://erlerobotics.gitbooks.io/erlerobot/content/es/ros/tutorials/creating_a_ros_package.html

FERNÁNDEZ, Y. (s.f.). *Qué es arduino, cómo funciona y qué puedes hacer con uno.* https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno. Descargado de https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

Galindo, P. M. (s.f.). *Unidad de medición inercial: estudio de fiabilidad y precisión.* Descargado de https://zaguan.unizar.es/record/59188/files/TAZ-TFG-2016-1026.pdf

Haviland, J. (2023). *Robotics toolbox for python.* https://github.com/petercorke/robotics-toolbox-python. Descargado de https://github.com/petercorke/robotics-toolbox-python

Installing gazebo_ros_pkgs (ros 1). (s.f.). https://classic.gazebosim.org/tutorials?tut=ros_installingcat=connect_ros. Descargado de https://classic.gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros

InvenSense. (2013). *Mpu-6000 and mpu-6050 register map and descriptions.* https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf. Descargado de https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

IOTProyectsIdeas. (s.f.). *Measure pitch roll and yaw angles using mpu6050 and arduino.* https://iotprojectsideas.com/measure-pitch-roll-and-yaw-angles-using-mpu6050-and-arduino/. Descargado de https://iotprojectsideas.com/measure-pitch-roll-and-yaw-angles-using-mpu6050-and-arduino/

Joober. (s.f.). *Ángulos de rotación con el imu de arduino nano 33 iot.* https://www.joober.eu/angulos-de-rotacion-con-el-imu-de-arduino-nano-33-iot/. Descargado de https://www.joober.eu/angulos-de-rotacion-con-el-imu-de-arduino-nano-33-iot/

Kumar, A. (2020). *How to use arduino with robot operating system (ros).* https://maker.pro/arduino/tutorial/how-to-use-arduino-with-robot-operating-system-ros. Descargado de https://maker.pro/arduino/tutorial/how-to-use-arduino-with-robot-operating-system-ros

Llamas, L. (2016a). *Controla tus proyectos con arduino y joystick analógico.* https://www.luisllamas.es/arduino-joystick/. Descargado de https://www.luisllamas.es/arduino-joystick/

Llamas, L. (2016b). *Determinar la orientación con arduino y el imu mpu-6050.* https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/. Descargado de https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/

Naylampmechatronics. (s.f.). *Tutorial mpu6050, acelerómetro y giroscopio.* https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html. Descargado de https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html

PELEGRÍ, J. (2022). *Universal robots ur5: CaracterÍsticas y aplicaciones.* https://www.universal-robots.com/es/blog/universal-robots-ur5/. Descargado de https://www.universal-robots.com/es/blog/universal-robots-ur5/

Prometec. (s.f.). *Usando el mpu6050.* https://www.prometec.net/usando-el-mpu6050/. Descargado de https://www.prometec.net/usando-el-mpu6050/

RoboticsCasual. (2021). *Ros tutorial: Simulate the ur5 robot in gazebo – urdf explained.* https://roboticscasual.com/ros-tutorial-simulate-ur5-robot-in-gazebo-urdf-explained/. Descargado de https://roboticscasual.com/ros-tutorial-simulate-ur5-robot-in-gazebo-urdf-explained/

ROS.org. (s.f.-a). *Creating a ros package.* http://wiki.ros.org/catkin/Tutorials/CreatingPackage. Descargado de http://wiki.ros.org/catkin/Tutorials/CreatingPackage

ROS.org. (s.f.-b). *Creating a workspace for catkin.* http://wiki.ros.org/catkin/Tutorials/create_a_workspace. Descargado de http://wiki.ros.org/catkin/Tutorials/create_a_workspace

ROS.org. (s.f.-c). *Ubuntu install of ros noetic.* http://wiki.ros.org/noetic/Installation/Ubuntu. Descargado de http://wiki.ros.org/noetic/Installation/Ubuntu

ROS.org. (s.f.-d). *ur_gazebo package summary.* http://wiki.ros.org/ur_gazebo. Descargado de http://wiki.ros.org/ur_gazebo

Said, A. (2019). *Ros, imu and an arduino: How to read imu sensor output and send it to ros.* https://atadiat.com/en/e-ros-imu-and-arduino-how-to-send-to-ros/. Descargado de https://atadiat.com/en/e-ros-imu-and-arduino-how-to-send-to-ros/

UniversalRobots. (2023). *Ros-industrial universal robot meta-package.* https://github.com/ros-industrial/universal_robot. Descargado de https://github.com/ros-industrial/universal_robot

Lista de Acrónimos y Abreviaturas

CPU Unidad Central de Procesamiento.
IDE Integrated Development Environment.
IMU Unidad de Medición Inercial.
ROS Robot Operating System.
TFG Trabajo Final de Grado.