Universidad Nacional de Cuyo

Facultad de Ingeniería

Ingeniería en Mecatrónica

Teleoperación de robot serie de tres grados de libertad

Realidad Virtual

Profesor:

Lic. Javier Rosenstein

Alumnos:

Gonzalo Romero Román

Resumen

La telerobótica es el área de la robótica concerniente al control de robot a distancia. Se utilizan conexiones inalámbricas para lograr la comunicación entre el operador y el sistema mecánico, permitiendo que las decisiones acerca de los movimientos del robot sean realizadas, en su mayoría, por el usuario. En este informe se implementa la teleoperación de un robot serie de tres grados de libertad a partir de acelerómetros colocados en el brazo del operador y a través de una interfaz gráfica desarrollada en Unity.

Índice

1. Introducción

La teleoperación de robots o telerobótica se refiere al control de robots a distancia. La telerobótica implica una barrera entre el usuario y el robot [1]. Esta barrera puede ser solucionada mediante control de forma remota. Este tipo de sistemas generalmente está dividido en dos sitios, un sitio local donde se encuentra el operador humano y todos los elementos necesarios para realizar el mando y las comunicaciones, y un sitio remoto donde se encuentra el robot, los sensores y los elementos de control necesarios para realizar los movimientos deseados. Las razones por las cuales en ocasiones es preferible optar por este tipo de sistemas pueden ser de distinta naturaleza. Las razones pueden ser ya sea financieras, humanas o técnicas [2] y muchas veces las aplicaciones de teleoperación pueden ser la única solución apropiada a los problemas propuestos. La presencia de un operador humano hace que la teleoperación de robots sea utilizada en entornos desconocidos y no estructurados. Las aplicaciones relacionadas con la telerobótica van desde aplicaciones espaciales, operación en ambientes peligrosos, búsqueda y rescate, aplicaciones médicas, entre muchas otras.

En el presente informe se realizará el control de un robot de tres grados de libertad mediante acelerómetros ubicados en el brazo del usuario. Se comunicarán los datos de dichos acelerómetros mediante una comunicación Bluetooth a una interfaz en Unity que recreará dichos movimientos a partir de un robot virtual con la mismos grados de libertad, configuración y dimensiones que el robot físico a controlar. Los valores cinemáticos obtenidos se comunicarán por Wifi al robot a través de un módulo Wifi ESP8266 que, utilizando un controlador PC9685, mueve los servomotores del robot a controlar. Primero se hará un análisis del robot, posteriormente se detallará el hardware utilizado para la aplicación, luego se describirán las comunicaciones realizadas, las conexiones utilizadas y se mostrarán los resultados obtenidos. Por último, se expondrán las conclusiones obtenidas a partir de dichos resultados.

2. Análisis de la aplicación

Como se mencionó anteriormente, se desea operar de forma inalámbrica un robot de tres grados de libertad mediante los movimientos de brazo de un operador. Los acelerómetros de colocarán en el brazo y antebrazo respectivamente. Esto puede observarse en la Figura 1. Éstos medirán los ángulos los tres ángulos necesarios para definir completamente la configuración del robot. El acelerómetro colocado en el antebrazo medirá dos ángulos, que permitirá mover las dos primeras articulaciones del robot. Mientras que el acelerómetro colocado en el antebrazo medirá solo un ángulo y permitirá mover la tercera articulación del robot.

Los datos posteriormente son enviados a una interfaz gráfica en Unity para luego ser llevados al robot a controlar. La comunicación entre los acelerómetros y la interfaz en Unity se realiza mediante Bluetooth y la comunicación entre

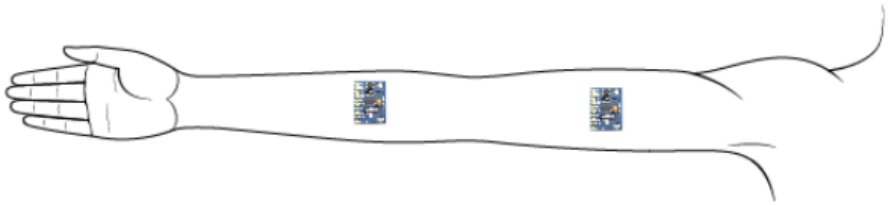


Figura 1: Disposición de los sensores en el brazo del operador

dicha interfaz y el robot se realiza mediante Wifi. Esto puede observarse en la Figura 2.

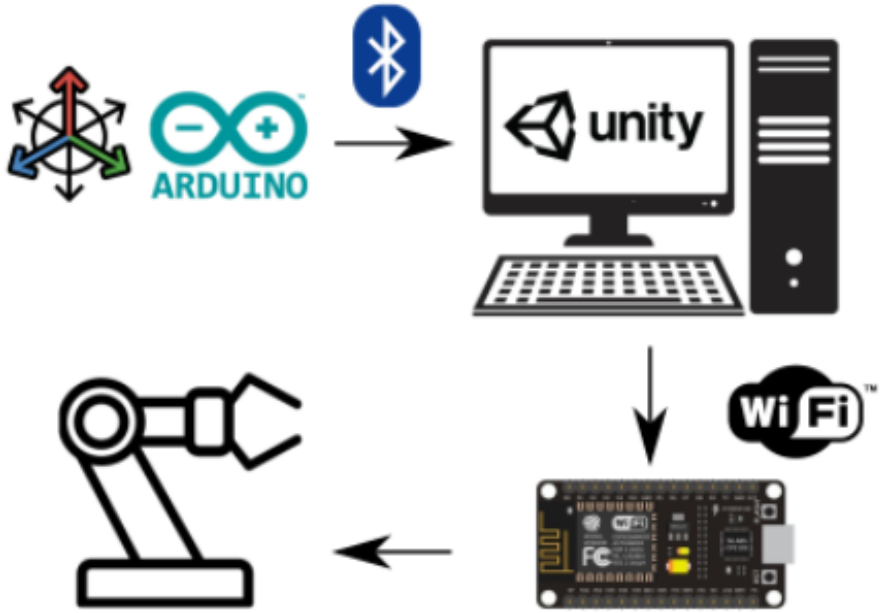


Figura 2: Arquitectura del sistema a desarrollar

Como puede observarse, se tendrán dos sitios, un sitio local en donde se encontrará el operador con los sensores y el ordenador, y un sitio remoto donde estará el robot. La comunicación entre ambos se hará por medio de una conexión Wifi.

Los datos son usados por la interfaz en Unity para poder simular el comportamiento del robot antes de que éstos sean realizados por el robot físico. De esta forma el usuario podrá ver cómo se comportará el robot sin tener que estar en contacto físico con éste.

2.1. Robot a controlar

Se pretende controlar un robot serie de tres grados de libertad. Éste puede observarse en la Figura 3. Este robot está compuesto por tres eslabones unidos entre sí por articulaciones rotacionales permitiendo un movimiento en el espacio del efector.

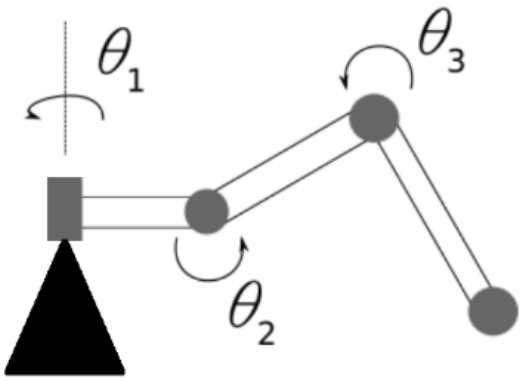


Figura 3: Diagrama descriptivo del robot

La posición y orientación del efector final con respecto a las variables articulares puede ser calculada a partir de la cinemática directa. Para ello se obtienen los parámetros de Denavit-Hartenberg. Los parámetros obtenidos para este robot se muestran en la Figura 4

	θ_i	d_i	a_i	α_i
1	θ_1	0	l_1	$\pi/2$
2	θ_2	0	l_2	0
3	θ_3	0	l_3	0

Figura 4: Parémetros de Denavit-Hartenberg del robot

La posición y orientación del efector final puede obtenerse fácilmente a partir de transformaciones homogéneas utilizando estos valores.

La cinemática inversa se utiliza para obtener el valor de las variables articulares en función de la posición y orientación del efector final.

A partir de la Figura 5 podemos obtener las siguientes ecuaciones de las coordenadas cartesianas del extremo de la pata referidas a la segunda articulación.

$$x_0 = x - l_1 \cos\theta_1$$

$$y_0 = y - l_1 \sin\theta_1$$

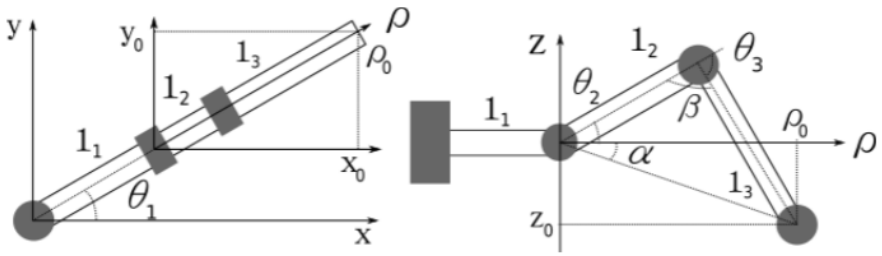


Figura 5: Cálculo de la cinemática inversa de la pata mediante el método geométrico. Fuente propia

Siendo:

$$\rho_0 = \sqrt{x_0^2 + y_0^2}$$

La primera variable articular puede obtenerse de la siguiente forma:

$$\theta_1 = \arctan(\frac{y_0}{x_0})$$

Usando la ley del coseno pueden obtenerse las demás variables articulares:

$$\alpha = \arctan\left(\frac{z_0}{\sqrt{x_0^2 + y_0^2}}\right)$$

$$\beta = \pm \arccos\left(\frac{x_0^2 + y_0^2 + z_0^2 - l_2^2 - l_3^2}{-2l_2 l_3}\right)$$

$$\theta_2 + \alpha = \pm \arccos\left(\frac{-l_3^2 + l_2^2 + x_0^2 + y_0^2 + z_0^2}{2l_2\sqrt{x_0^2 + y_0^2 + z_0^2}}\right)$$

$$\theta_2 = \pm \arccos\left(\frac{-l_3^2 + l_2^2 + x_0^2 + y_0^2 + z_0^2}{2l_2\sqrt{x_0^2 + y_0^2 + z_0^2}}\right) - \arctan\left(\frac{z_0}{\sqrt{x_0^2 + y_0^2}}\right)$$

La tercera variable articular está dada por:

$$\theta_3 = -(\pi - \beta)$$

$$\theta_3 = \pm \arccos\left(\frac{x_0^2 + y_0^2 + z_0^2 - l_2^2 - l_3^2}{2l_2 l_3}\right)$$

Teniendo de esta forma dos soluciones: codo arriba y codo abajo. Dado que se pretende que en todo momento sea la misma cara de la base la que enfrente el suelo se considera como única solución el codo arriba. Por lo tanto las ecuaciones quedarían de la siguiente forma:

$$\theta_1 = \arctan(\frac{y_0}{x_0})$$

$$\theta_2 = \arccos\left(\frac{-l_3^2 + l_2^2 + x_0^2 + y_0^2 + z_0^2}{2l_2\sqrt{x_0^2 + y_0^2 + z_0^2}}\right) - \arctan\left(\frac{z_0}{\sqrt{x_0^2 + y_0^2}}\right)$$

$$\theta_3 = -\arccos\left(\frac{x_0^2 + y_0^2 + z_0^2 - l_2^2 - l_3^2}{2l_2 l_3}\right)$$

Siendo entonces θ_1, θ_2 y θ_3 son los valores de las variables articulares del robot y x_0, y_0 y z_0 son las componentes de la posición del efector final con respecto al sistema de referencia fijo al suelo.

3. Desarrollo de la aplicación

En la siguiente sección se detallarán los pasos necesarios para llevar a cabo la aplicación propuesta. Primero se hará un análisis del hardware a utilizar y posteriormente se explicará de forma específica cada una de las partes que conforman el proyecto.

3.1. Selección del Hardware

Para poder llevar a cabo la aplicación es necesario utilizar distintos tipos de sensores, placas controladoras y módulos de comunicación. En esta sección se detallarán las características principales de cada una de ellas.

3.1.1. Acelerómetro y giróscopo MPU 6500

Se utiliza este sensor para medir los ángulos del antebrazo del operador. El MPU6050 combina un giróscopo y un acelerómetro de tres ejes junto con un procesador de movimiento digital. Los valores obtenidos pueden ser tratados para obtener los valores roll, pitch y yaw. El sensor descripto puede observarse en la Figura 6

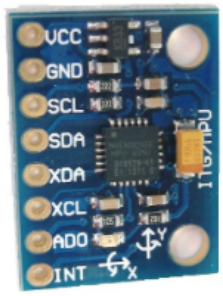


Figura 6: Acelerómetro y giróscopo MPU 6500

3.1.2. Acelerómetro, giróscopo y magnetómetro MPU 9250

Se utiliza un sensor MPU 9250 para medir los ángulos del brazo del operador que serán utilizados para darle consigna de posición angular a los actuadores del robot, éste se muestra en la Figura 7. El MPU-9250 es un sensor de 9 ejes de bajo consumo. El sistema posee 2 chips, un MPU-6500 que contiene un giróscopo de 3 ejes y un acelerómetro de 3 ejes. Además posee un magnetómetro de 3 ejes AK8963. La información provista por el sensor permite calcular los ángulos de roll, pitch y yaw necesario para darle consignas de posición al robot. La información recolectada por el sensor es comunicada por I2C a la placa controladora de destino.

Figura 7: Acelerómetro, giróscopo y magnetómetro MPU 9250

3.1.3. Arduino Uno / Pro Mini

Para poder obtener y procesar los valores medidos por los acelerómetros, obteniendo los ángulos roll, pitch y yaw de cada uno de ellos y gestionar la transferencia de datos a través del módulo bluetooth se utiliza un Arduino Uno. Un Arduino Uno es una placa controladora basada en el Atmega328P. Éste posee 14 pines digitales y 6 analógicos, suficientes para la aplicación en cuestión. Para reducir el tamaño del sistema, se puede utilizar un Arduino Pro Mini, que está provisto con el mismo controlador Atmega328P, por lo que comparte los mismos registros y los programas pueden ser fácilmente exportados entre una placa y otra. Ambos Arduino pueden observarse en la Figura 8.

3.1.4. Módulo Bluetooth HC-05

Para poder comunicar los datos calculados por el Arduino a la interfaz gráfica en Unity se utiliza una comunicación Bluetooth. Para ello se utiliza un módulo HC-05 que es un módulo Bluetooth SPP diseñado configuraciones de conexiones seriales transparentes. El módulo puede verse en la Figura 9.

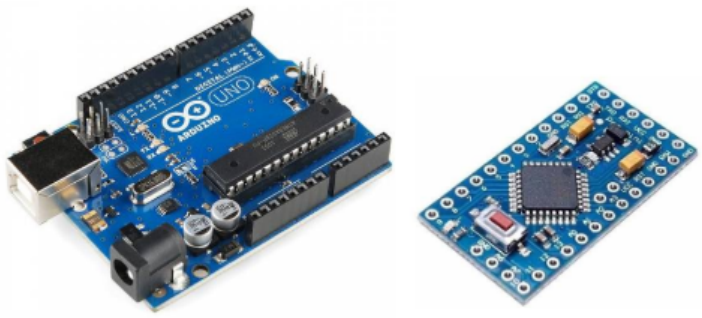


Figura 8: Arduino Uno (izquierda) y Arduino Pro Mini (derecha)

Figura 9: Módulo Bluetooth HC-05

3.1.5. Módulo Wifi NodeMCU

Con el objetivo de recibir la información del servidor en Unity y transferir los datos a la placa controladora de los servos se utiliza un módulo Wifi NodeMCU que se muestra en la Figura 10. NodeMCU es una plataforma Open Source de bajo costo que utiliza un módulo Wifi ESP8266 con un stack TCP/IP completo y un microcontrolador. Dicha placa puede ser programada a partir de Arduino IDE y provee una interfaz I2C implementada por software que permite la comunicación con la placa controladora de servomotores que se explicará en la siguiente sección.

3.1.6. Placa controladora PCA9685

El robot a controlar está provisto por tres articulaciones rotacionales que son movidas por servomotores. Para poder controlar dichos motores se utiliza una placa PCA9685, ésta se muestra en la Figura 11. La placa permite controlar hasta 16 servomotores a través de I2C. Esto permite reducir la cantidad de pines necesarios para mover los motores y simplificar la programación requerida.

Figura 10: Módulo Wifi NodeMCU

Figura 11: Placa controladora PCA9685

3.1.7. Servos Futaba S3003

Para poder movilizar la estructura del robot se utilizan servomotores Futaba S3003 que permiten movimientos de entre 0 y 180° en cada una de las articulaciones del brazo robótico. Estos son controlados a partir de la placa controladora previamente descrita y proveen un torque de 3,17 kg-cm a 5V, más que suficiente para mover la estructura del robot. Uno de los motores utilizados puede observarse en la Figura 12.

Figura 12: Servomotor Futaba S3003

3.2. Obtención de los datos

Para la obtención de los ángulos se utilizan acelerómetros dispuestos en el brazo y el antebrazo del operador. Éstos miden los ángulos necesarios para poder movilizar las articulaciones, tanto del robot virtual como del robot físico.

Para las primeras dos articulaciones del robot se utiliza un MPU9250. Éste está provisto con un magnetómetro y permite medir con mayor precisión el ángulo yaw. El MPU6050, que carece de magnetómetro, se utiliza para medir el ángulo del antebrazo y movilizar la última articulación del robot.

Además se tiene un joystick que permite variar la posición y orientación de la cámara de la simulación permitiendo al operador variar la perspectiva de visualización para tener un mejor conocimiento del espacio de trabajo y la posición actual del robot. Los datos obtenidos se muestran en la Figura 13. En ella podemos visualizar los tres valores obtenidos del MPU9250, los valores de X, Y y el botón del joystick y el ángulo de roll del MPU6050.

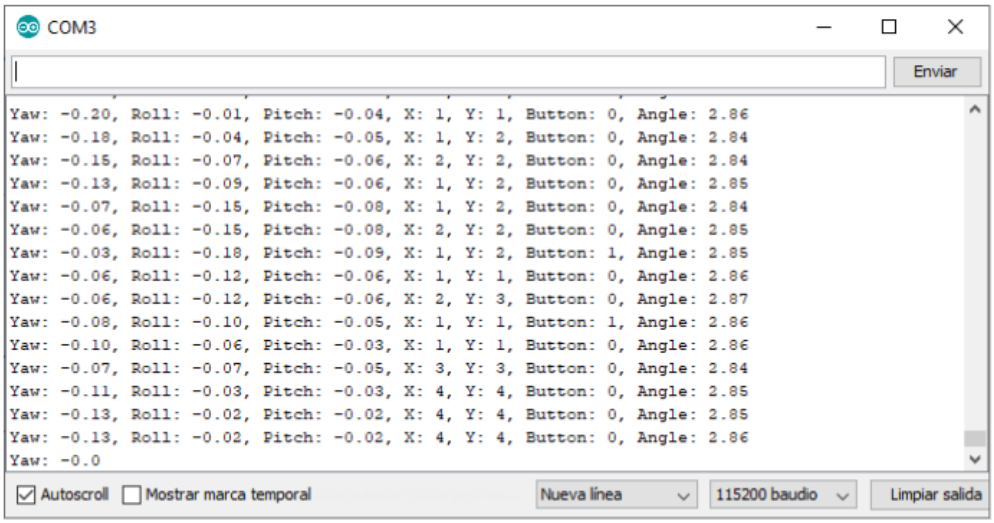


Figura 13: Datos obtenidos de los sensores y mostrados en el monitor serial

Los datos obtenidos son enviados a través del módulo bluetooth HC05 a la aplicación.

Una vez se inicializa el programa, se inicia la comunicación I2C y se establece la comunicación serial a 115200 baudios. Se crea una interrupción en el timer 1 del arduino cada 50 milisegundos para el envío de información y se le asocia la rutina encargada de ello. Se establece la velocidad de comunicación del módulo bluetooth, que debe ser previamente modificada utilizando comandos AT. Se inicializan los acelerómetros en las direcciones 0x68 y 0x69 (se debe poner el pin AD0 de este último acelerómetro en alto) y por último se habilitan las interrupciones.

Se espera un tiempo de 15 segundos hasta que se estabilizan los datos obtenidos por los acelerómetros y una vez pasado este tiempo se ponen en cero los valores medidos mediante un offset. A partir de ese momento se comienza a enviar los datos a la simulación,

En bucle loop, es decir, en cada iteración, se calculan los valores de los acelerómetros, se leen los valores del joystick, se asignan a sus respectivas variables para luego ser enviadas y se imprimen los valores por pantalla.

Para la medición de los datos provistos por el acelerómetro se utiliza la librería MPU9250 desarrollada en [3]. Mientras que para las medidas obtenidas por el MPU6050, se obtienen los valores crudos, se procesan y se utiliza un filtro complementario utilizando los valores del acelerómetro y del giróscopo para obtener valores de ángulos más precisos.

En la Figura 14 puede verse el diagrama de conexiones utilizado para la obtención de datos y la transmisión de estos a través del módulo bluetooth.

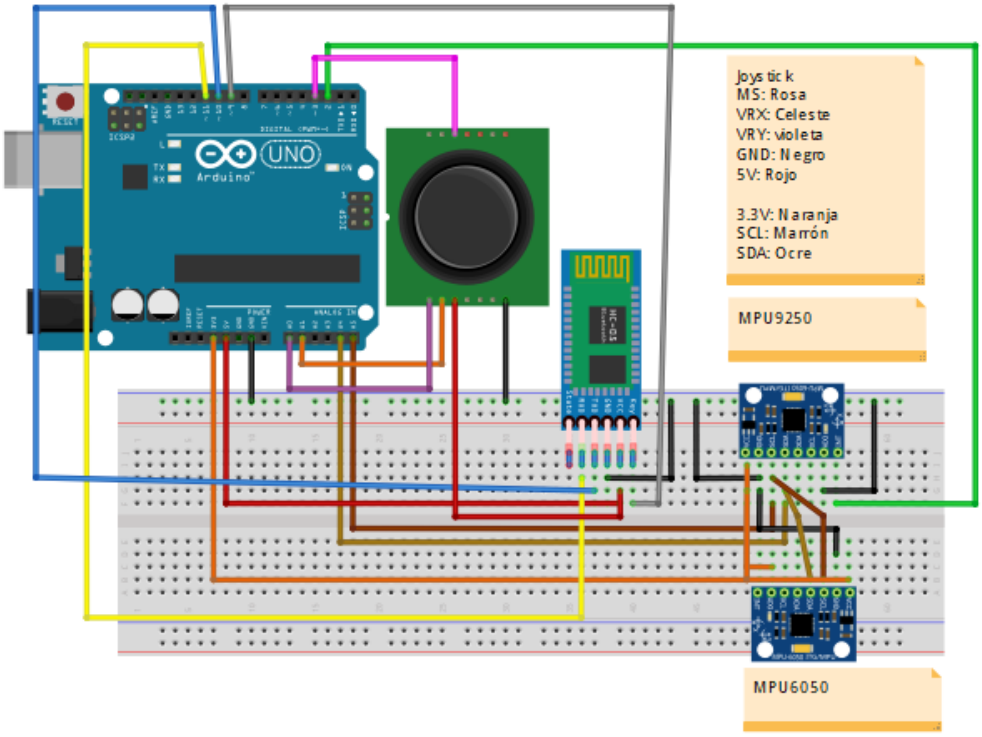


Figura 14: Diagrama de conexiones

3.3. Comunicación Bluetooth

Para realizar la comunicación entre el Arduino y la aplicación en Unity se utiliza una conexión Bluetooth. Ésta es una especificación para redes inalámbricas de área personal que posibilita la transmisión de datos entre dispositivos mediante un enlace por radiofrecuencia en la banda de los 2,4 GHz. Permite una comunicación veloz y sencilla entre dispositivos. Si bien el alcance de este tipo de comunicación es una de sus desventajas, la aplicación en cuestión no requiere transferir información a distancias largas entre el dispositivo de medición de ángulos y la PC. Para transferir información desde la PC al Robot se requieren distancias más elevadas y por esto se utiliza en ese caso una comunicación WiFi y no una Bluetooth.

Los datos provistos por los acelerómetros y por el joystick son enviados utilizando el módulo HC-05 previamente mencionado utilizando la librería de Arduino SoftwareSerial.h. Una vez realizada la aplicación en Arduino se verifica la correcta transmisión de datos a partir de una terminal serial. Los datos transmitidos pueden observarse en la Figura 15.

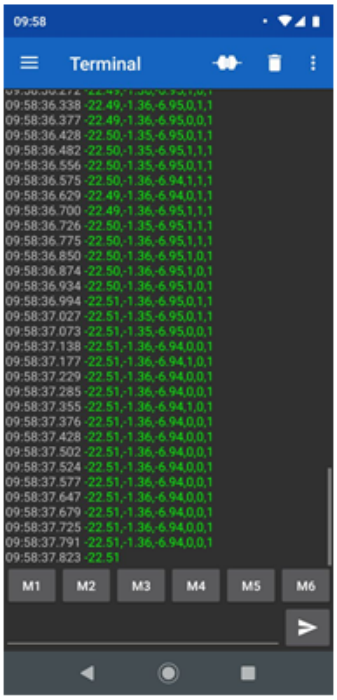


Figura 15: Verificación de la transmisión de datos

Una vez que se corrobora que los valores son transferidos con éxito, se procede a realizar la recepción de datos desde la aplicación en Unity. Para ello se crea un GameObject y se le asigna un Script que será el encargado de la recepción

de los datos.

Cuando inicializa el programa, la función Start llama al método Connect-Bluetooth. Dicho método puede observarse en la Figura 16.

```
void ConnectBluetooth()
{
    sp = new SerialPort("\\\\.\\COM5", 115200);
    if (!sp.IsOpen)
    {
        sp.Open();
        sp.ReadTimeout = 100;
        sp.Handshake = Handshake.None;
    }
}
```

Figura 16: Configuración de la conexión Bluetooth en Unity

En él se define el puerto de comunicación y la velocidad de transmisión de datos, para luego abrir el puerto e iniciar la transferencia de información. Una vez conectado se comienza a leer los datos utilizando la función Update, como puede verse en la Figura 17.

```
string data;
if (sp.IsOpen)
{
    try
    {
        data = sp.ReadLine();
        sp.BaseStream.Flush();
        text = data.ToString();
        string[] values = text.Split(',');
        Global.Instance.eulerAngles = new float[]
        { float.Parse(values[0]), float.Parse(values[1]), float.Parse(values[2]) };
        Global.Instance.x_joystick = float.Parse(values[3]);
        Global.Instance.y_joystick = float.Parse(values[4]);
        Global.Instance.button_joystick = float.Parse(values[5]);
        Global.Instance.angleLast = float.Parse(values[6]);
    }
    catch
    {

    }
}
```

Figura 17: Recepción y guardado de datos

Se lee una línea y se le asigna a una variable, se divide el String obtenido y se le asigna a cada una de las variables globales los datos recibidos. Previo a

esto, se castean los datos para que estos sean del tipo flotante. Esto es realizado nuevamente en cada iteración.

La recepción de los datos puede verse en la Figura 18. Se lee la información recibida en el puerto serial y se muestra por consola con el fin de verificar la correcta recepción de los datos.

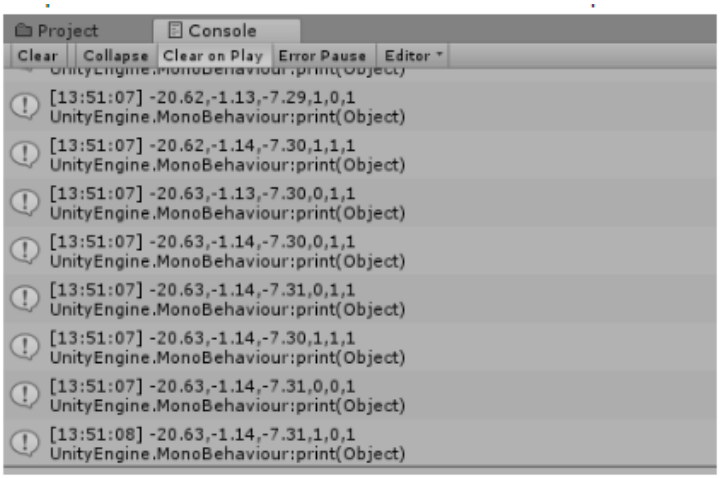


Figura 18: Obtención de los datos en Unity

Se utiliza un singleton para guardar los datos obtenidos de forma global. La intención del singleton es garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global.

3.4. Interfaz en Unity

Para realizar la interfaz gráfica y poder modelar el comportamiento del robot se utiliza el motor de videojuegos multiplataforma Unity. Este es uno de los motores de videojuegos más utilizados actualmente y provee herramientas para el desarrollo de juegos en 2D, 3D, realidad virtual, entre otros. Las facilidades provistas permiten que esta sea una herramienta apropiada para el desarrollo de la aplicación.

Para corroborar la correcta recepción de datos primero se crean los GameObject. Se crea, en un comienzo, un eslabón y una articulación de dos grados de libertad para las primeras dos articulaciones y eslabones del robot, y un eslabón y una articulación de un grado de libertad para los restantes. Se le asigna un prisma a cada eslabón y un pequeño cubo a las articulaciones, esto permite que se pueda producir una rotación por un eje que no sea el centro geométrico del eslabón. Los GameObject utilizados pueden observarse en la Figura 19.

Los orígenes de los sistemas de referencia del eslabón y articulación pueden observarse en la Figura 20. Éstos corresponden los sistemas de referencia relacionados con el segundo eslabón.

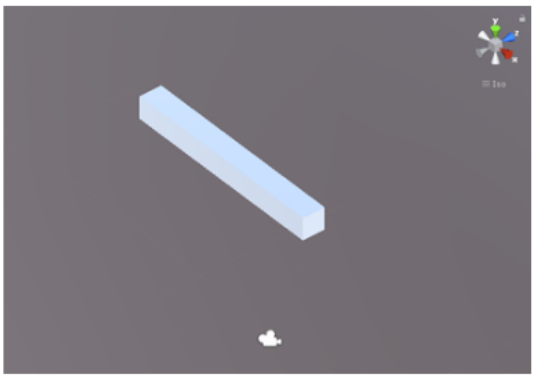
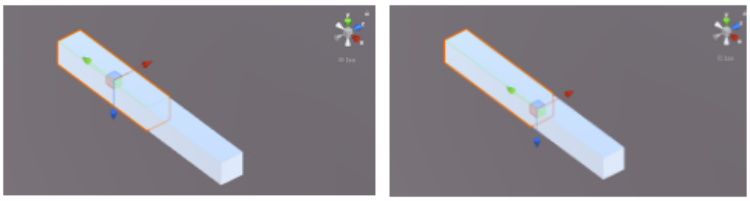


Figura 19: GameObject utilizados



(a) Sistema de referencia del eslabón 2

(b) Sistema de referencia de la articulación 2

Figura 20: Sistemas de referencias

Por lo tanto, en un principio, la escena es de la forma mostrada en la Figura 21. Se tendrá el programa encargado de la gestión de las comunicaciones Bluetooth, la cámara principal, la iluminación, el suelo y las articulaciones y eslabones mencionados anteriormente. Siendo el eslabón uno y la articulación dos nodos hijos de la articulación uno y el eslabón dos un nodo hijo de la segunda articulación.

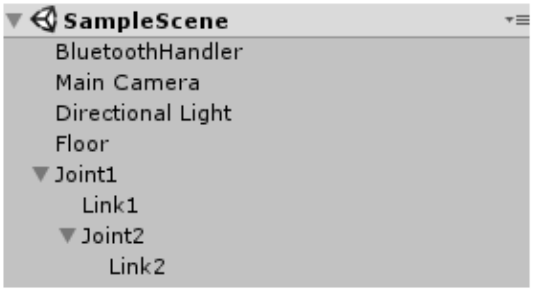


Figura 21: Escena en unity

Se les asigna a cada articulación un Script que realizará una rotación local en función de los datos obtenidos por Bluetooth. Es necesario redefinir el orden en que deben realizarse los ejes para adaptarse a los ángulos de Euler previamente calculados. En la Figura 22 pueden observarse las transformaciones realizadas en la última articulación del robot. Primero se realiza una rotación con respecto al eje X del sistema de referencia fijo a la pieza (Vector3.right) para alinearla con el eje de rotación. Posteriormente se realiza una rotación en Y (Vector3.up) con un valor igual al ángulo provisto por el operador. No se realizan rotaciones con respecto al eje Z (Vector3.forward).

```
transform.localRotation =
   Quaternion.AngleAxis(90, Vector3.right) *
   Quaternion.AngleAxis(Global.Instance.angleLast, Vector3.up) *
   Quaternion.AngleAxis(0, Vector3.forward);
```

Figura 22: Transformaciones realizadas en la última articulación

Se ejecuta el programa y se verifican los movimientos de las articulaciones en función de la variación de los ángulos en los acelerómetros. Los movimientos obtenidos pueden observarse en la Figura 23.

(a) Simulación en el instante 1 (b) Simulación en el instante 2

Figura 23: Ejecución del programa

Una vez verificado el funcionamiento, se crea un objeto por cada eslabón y uno por cada articulación. La simulación puede observarse en la Figura 24. Se agrega un botón para recargar la escena en caso de una pérdida de conexión.

Una vez que se agrega el Singleton, el gestor para las comunicaciones Wifi y el botón, la escena será de la forma mostrada en la Figura 25.

3.4.1. Movimiento de la cámara

Como se mencionó en un principio, la información provista por el joystick sirve para que el operador pueda variar la perspectiva de donde visualiza al robot. Para lograr esto se agrega un Script a la cámara que, en cada iteración, lee

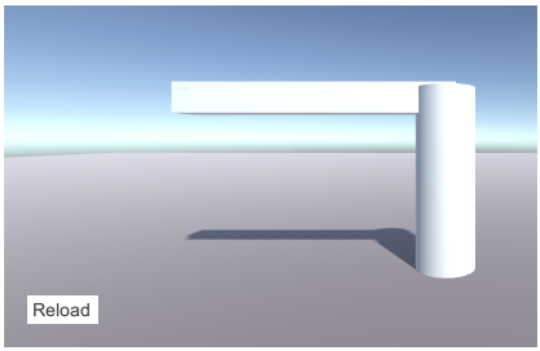


Figura 24: Ejecución del programa luego de los cambios

Figura 25: Escena actualizada

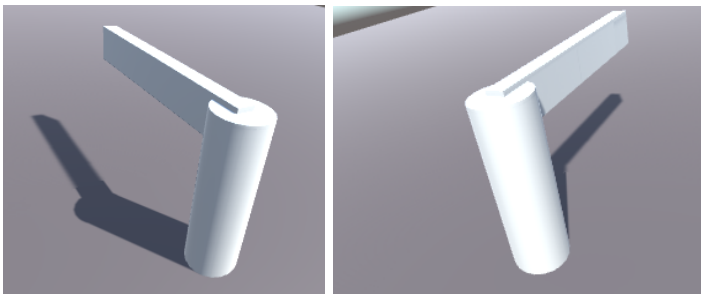
los valores del Joystick. En función de dichos valores se realiza una transformación alrededor del robot usando la función RotateAround. Dado que se tienen dos ejes, los datos provistos por el joystick permiten al operador visualizar al robot desde la perspectiva deseada, teniendo un mayor conocimiento sobre la posición y el espacio de trabajo de éste. Distintas perspectivas para la misma configuración pueden observarse en la Figura 26.

3.4.2. Utilización de las partes del robot

Para que la aplicación sea más realista y el operador pueda visualizar con mejor detalle la posición del efector final y las dimensiones reales de los actuadores y de los eslabones, es necesario reemplazar los eslabones utilizados hasta ahora por las piezas del robot real.

Las piezas fueron diseñadas utilizando el software de diseño 3D Solid Edge, previamente a ser impresas. El modelo del robot en Solid Edge puede observarse en la Figura 27.

Los archivos en Solid Edge tienen formato .par para el caso de las partes y .asm para el caso de los conjuntos. Ninguno de los formatos anteriores es



(a) Perspectiva 1 (b) Perspectiva 2

Figura 26: Distintas perspectivas de la cámara

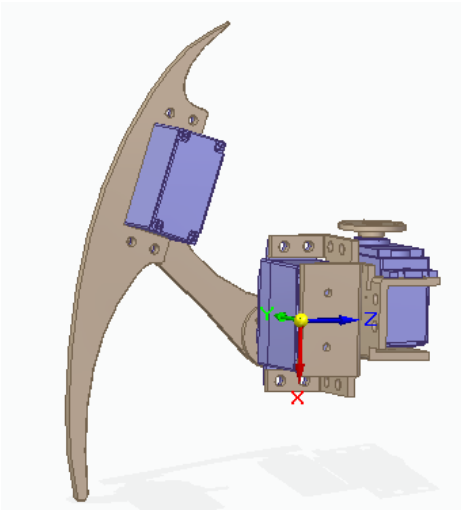


Figura 27: Modelo del robot en Solid Edge

soportado por Unity. Por ello es necesario cambiar el formato a .obj para que puedan ser subidos. Para ello es necesario guardar los archivos en Solid Edge en el formato VRLM para luego ser transformados a .obj.

Debido a que al ser guardados los archivos en formato VRLM se obtendrán piezas rígidas y no se permitirá el desplazamiento relativo de las partes, es necesario crear tres archivos diferentes con cada uno de los eslabones del robot. Esto puede verse en la Figura 28.

Una vez guardadas las piezas en el formato mencionado, es necesario transformarlas a formato .obj. Para ello se utiliza es software MeshLab. Éste es un programa Open Source para el editado y procesado de mallas en 3D. Se cargan las piezas de forma individual. Esto puede verse en la Figura 29.

Se guardan las tres piezas en formato .obj. Una vez realizado esto, las piezas pueden ser fácilmente usadas en Unity, solo es necesario arrastrarlas a su

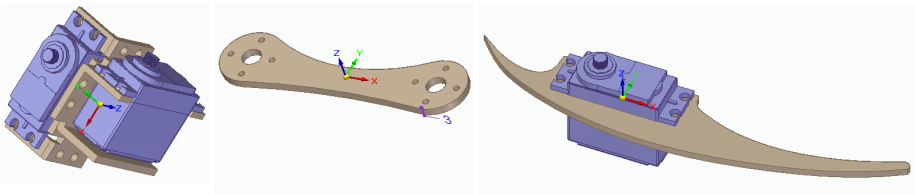


Figura 28: Piezas rígidas que conforman al robot

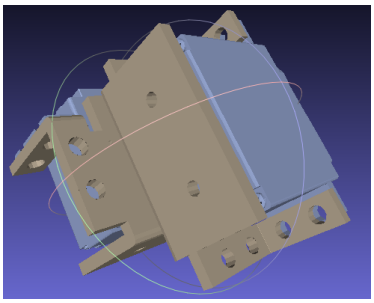


Figura 29: Pieza en MeshLab

interfaz. Estando las piezas cargadas, se hace el mismo desarrollo que el realizado anteriormente. Se definen las articulaciones y se utilizan las piezas del robot como eslabones. Se definen los ejes de giro y se agregan los scripts que posibilitarán los movimientos del robot. Una vez realizado todo esto, se habrán remplazado los eslabones prismáticos por los modelos de las piezas reales del robot. Esto puede observarse en la Figura 30.

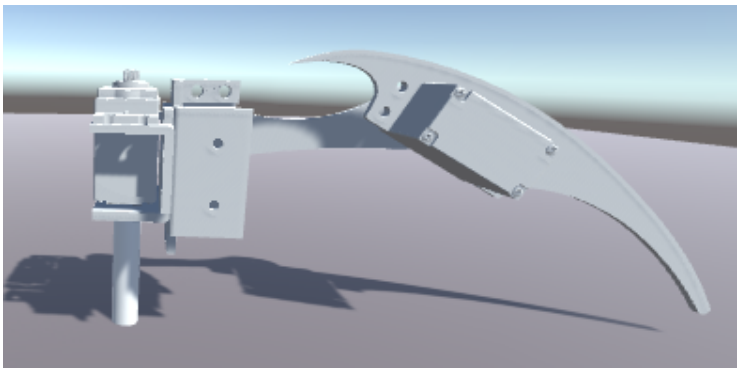


Figura 30: Brazo robótico en Unity

Las nuevas piezas en la simulación permiten al operador tener una descripción más completa del estado del robot y su posición en el espacio.

En caso que se desee, se puede optar por utilizar elementos para simular

el entorno específico donde estará operando robot. Esto permitirá al usuario tener una interfaz gráfica más amigable, permitiéndole tener más información del lugar donde el robot se está moviendo. Para ello se descarga de la Asset Store el paquete "3D Kit Starter Kit", un entorno similar a una fábrica para que el robot opere. Dicho entorno se agrega a la simulación. Esto puede verse en la Figura 31.

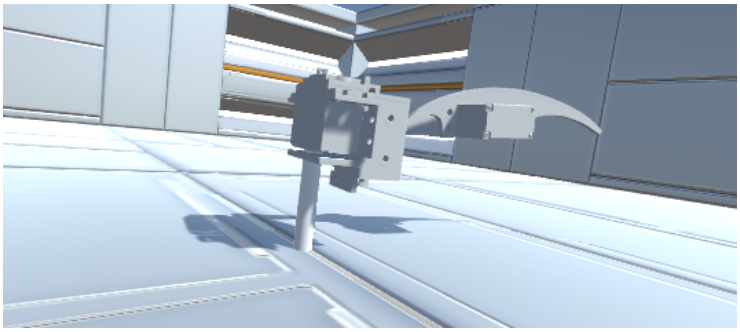


Figura 31: Entorno similar a una fábrica

Si bien en el programa no se tienen los elementos necesarios para que la fábrica donde se opera el robot sea similar a una fábrica real. El diseño y generación de un entorno más complejo excede las capacidades del proyecto y por lo tanto no se tratará en el presente informe.

3.5. Comunicación Wifi

La conexión entre la aplicación y el robot se realiza a través de una conexión WiFi. La aplicación en Unity sirve de servidor mientras que el módulo ESP8266 sirve de cliente. Se eligió este tipo de conexión debido a que permite una transmisión de información a mayores distancias.

Una vez que se inicializa la escena en Unity, se crea un hilo que inicia el servidor y espera al cliente a que se conecte. Esto puede observarse en la Figura 32. La utilización del hilo permite que la simulación no tenga que esperar a que se establezca la comunicación para iniciarse.

```
void Start () {

    m_Thread = new Thread(ConnectWifi);
    m_Thread.Start();

}
```

Figura 32: Creación del hilo que inicializa el servidor

En el hilo se ejecuta la función ConnectWifi, esta se muestra en la Figura 33.

Se inicia el servidor en el puerto 8080 y se espera la conexión por parte del cliente. Una vez conectado el cliente comienza la transferencia de datos.

```
void ConnectWifi()
{
    IPAddress ipaddress = IPAddress.Parse("0.0.0.0");
    listener = new TcpListener(ipaddress, 8080);
    listener.Start();
    print("Server mounted,  listening to port 8080");
    soc = listener.AcceptSocket();
    print("Connected");

    s = new NetworkStream(soc);
    sw = new StreamWriter(s);
    sw.AutoFlush = true; // enable automatic flushing
}
```

Figura 33: Creación del servidor

En cada Update se envían los datos al robot. Se utiliza para ello el método SendData mostrado en la Figura 34. Se crean cadenas de caracteres para posteriormente ser transmitidas.

```
void SendData()
{
    valueTag[0] = (int)Global.Instance.eulerAngles[0];
    valueTag[1] = (int)Global.Instance.eulerAngles[1];
    valueTag[2] = -(int)Global.Instance.angleLast;
    valueTag[3] = (int)Global.Instance.x_joystick;
    valueTag[4] = (int)Global.Instance.y_joystick;
    valueTag[5] = (int)Global.Instance.button_joystick;


    for (int i = 0; i < valueTag.Length; i++)
    {
        chain = CreateChain(dataTag[i], valueTag[i]);
        sw.WriteLine(chain);
    }


}
```

Figura 34: Envío de los datos

Se envían los ángulos leídos por los giróscopos para luego ser tratados de

manera de obtener los ángulos de consigna para los actuadores.

Los datos son enviados en cadenas de caracteres de 8 bits de la forma :Xvvv– donde : señala el inicio de la cadena, X determina el ángulo y valor del joystick vvv representa el valor en entero y - representa el fin de la cadena. El último bit corresponde al salto de línea. Posteriormente el cliente lee dichos valores y estos pasan por un intérprete de comandos que interpreta las cadenas y guarda los valores en sus respectivas variables.

Del lado del cliente se utiliza la librería ESP8266WiFi que permite configurar el cliente de forma simple.

Los datos obtenidos del servidor se muestran a través del monitor serie como se muestra en la Figura 35.

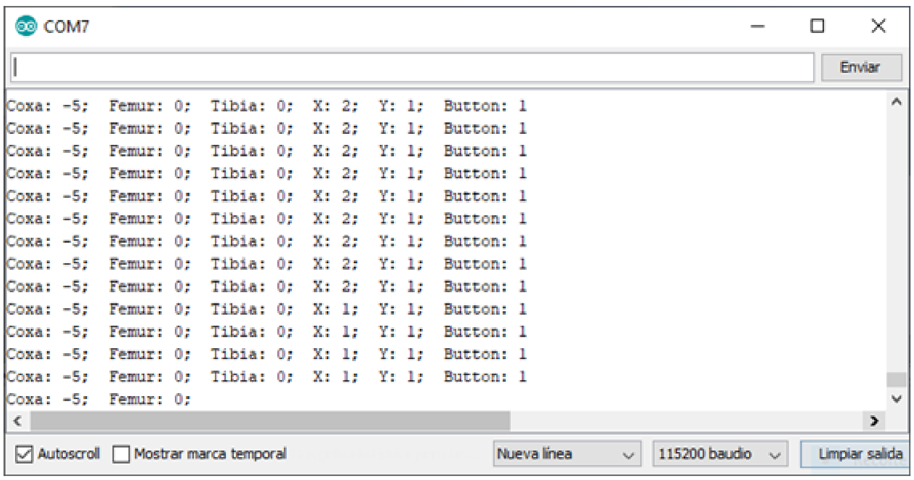


Figura 35: Datos obtenidos por el módulo NodeMCU

3.6. Control del robot

Se utiliza un controlador PCA9685 para generar los PWM necesarios para movilizar los motores del robot a sus posiciones deseadas. Las conexiones realizadas se muestran en la Figura 36. La placa es alimentada por el módulo Wifi con una tensión de 5V. La comunicación se realiza por I2C, esto se implementado por software mediante los pines D1 y D2 del NodeMCU. Se conectan los motores a los respectivos pines.

La placa es alimentada por una fuente externa debido a la gran cantidad de corriente que requieren los motores para movilizarse. Se alimenta con 5V. Se utilizó una fuente de PC y un regulador LM317T para mantener la tensión en un valor lo más constante posible. Las conexiones necesarias para regular la tensión usando el LM317T pueden ser vistas en la datasheet provista por el fabricante.

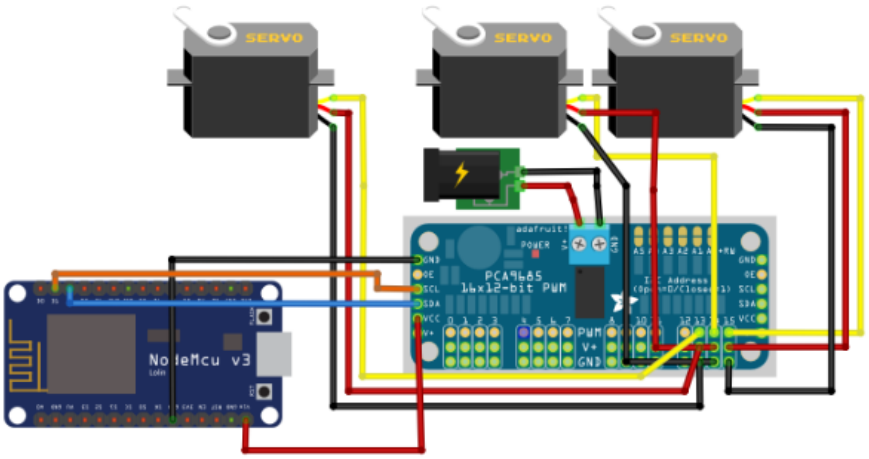


Figura 36: Diagrama de conexiones para la movilización de los motores

Los motores son movidos utilizando la librería AdafruitPWMServoDriver que permite establecer fácilmente los anchos de pulso necesarios para mover el motor a un ángulo específico. Para ello es necesario obtener, mediante ensayos, los anchos de pulso que corresponden a los ángulos cero de cada motor y los correspondientes a los valores máximos y mínimos.

Una vez obtenidos dichos valores, se procede a mapear los anchos de pulso con el fin de obtener cada ancho de pulso en función de cada ángulo específico.

4. Conclusión

El presente informe analizó la teleoperación de un robot serie de tres grados de libertad. Se analizó el robot, se detalló el hardware utilizado, se explicó las comunicaciones realizadas y la interfaz gráfica en Unity. Una vez puesta en marcha la aplicación, las comunicaciones y funcionamiento de cada parte fueron exitosos. Si bien el movimiento del robot no fue exactamente el deseado por parte del usuario debido a los retardos de comunicación, la precisión de los actuadores y la rigidez de los eslabones, estos problemas pueden ser solucionados utilizando otros actuadores y cambiando el material con el que está hecho el robot. Dejando de lado estas falencias, la visualización en Unity permitió al usuario conocer la posición del robot y éste respondió correctamente a las consignas del operador.

Referencias

[1] G. Niemeyer, C. Preusche, and G. Hirzinger, "Handbook of robotics," *Springer*, 2008.

[2] A. Leleve, "Contribution a la téléoperation de robots en présence de délais de transmission variables," *Sciences et Techniques du Languedoc*, 2000.

[3] Hideakitai, "Mpu9250," *Github*, 2020.