Simulación de sistema de control a distancia de robots exploratorios. Prueba de concepto en realidad virtual con control gestual y por hardware.

Autores: Pavéz Fabriani Eduardo, Tous Martín

Mayo de 2023

Resumen

En el presente trabajo se realiza la simulación del control de un robot móvil con orugas mecánicas dotado de un brazo robótico y una cámara, el cual tiene como finalidad ser utilizado para la inspección de zonas de difícil acceso o incluso peligrosas para un ser humano.

La finalidad de dicho proyecto es desarrollar un sistema de control intuitivo y fácil de utilizar para el operario. El mismo consiste en una simulación desarrollada en Unity en donde se navega a través de ciertos menús que permiten el registro del operario y finalmente la representación de un posible entorno a través del cual operaría dicho robot.

Se cuenta con un sistema de control que, haciendo uso de la cámara del dispositivo en el cual se corre la simulación, reconoce el movimiento de las manos mediante "*handtracking",* y de esta forma se controla el desplazamiento del robot. A su vez, se cuenta con un dispositivo hardware que, mediante conexión bluetooth, se utiliza tanto como escáner de la credencial de registro del operario como así también para controlar la cámara del robot en cuestión. Se desarrolla un método de registro alternativo con realidad aumentada, reconocimiento de credencial por imagen y el uso de una contraseña personal.

Se ofrece una alternativa de procesamiento del handtracking utilizando cloud computing para alivianar la carga computacional del dispositivo que corre la simulación.

Índice

1. Introducción

La exploración y la inspección de zonas de difícil acceso o peligrosas son tareas esenciales en varios campos, desde la exploración espacial hasta la inspección de infraestructuras críticas. Sin embargo, estas actividades a menudo implican riesgos para los seres humanos debido a las condiciones extremas o peligrosas en las que se realizan. Para abordar este problema, se han desarrollado robots móviles capaces de realizar estas tareas de manera segura y efectiva.

El objetivo principal de este proyecto es desarrollar un sistema de control intuitivo y fácil de utilizar para comandar un robot móvil con orugas mecánicas, dotado de un brazo robótico y una cámara, el cual es utilizado para la inspección de zonas de difícil acceso o peligrosas.

Para lograr esto, se ha desarrollado una simulación en Unity que permite controlar una versión virtual del robot móvil mediante gestos manuales, haciendo uso de "handtracking", y un dispositivo hardware que se comunica con la simulación mediante Bluetooth. Este sistema de control intuitivo aumenta la seguridad y la eficiencia del trabajo, y reduce los riesgos asociados con la exposición humana a ambientes peligrosos.

Además, se ha incluido un dispositivo hardware que permite escanear la credencial del operario y validar su identidad y permiso. Esto hace que el sistema sea seguro y fácil de utilizar para cualquier operario.

En general, esta simulación transita a través de tres escenas distintas, en las cuales, por un lado, se le da la bienvenida al operario/usuario, luego se le pide que se registe y en la última escena se ingresa en el entorno virtual en el cual se comanda al robot. Dicho entorno virtual es una representación de un posible entorno a través del cual operaría dicho robot, en este caso, se optó por un entorno industrial.

Este proyecto cuenta con dos variantes, una en la cual se corre la simulación desde una PC, y otra en la que corre en un dispositivo Android. La versión en PC cuenta con la posibilidad de utilizar el dispositivo hardware, tanto para el registro como así también para el control de la cámara en la última escena, contrastándolo con el control del robot mediante gestos. La versión para Android no utiliza el controlador Hardware, sino que es totalmente un control gestual tanto para el robot como para la cámara en la última escena. La finalidad de contrastar ambas versiones es analizar sus ventajas y desventajas para así brindar alternativas que se adecúen al uso final.

En un desarrollo futuro en el que se contara con el robot físico, el mismo transmitiría hacia la PC o hacia el celular un modelo de su entorno obtenido a través de sus sensores. En tal caso se trabajaría con una representación virtual de ese entorno en tiempo real.

El mundo virtual desarrollado durante este proyecto serviría también como entrenamiento para el operario, evitando los riesgos del envío del comando equivocado en una situación real que podría producir daño en el robot o en las instalaciones, y por ende, pérdidas económicas.

2. Diseño y desarrollo

En esta sección se proporciona una descripción técnica de las herramientas utilizadas durante el desarrollo del proyecto. Se detallan tanto el hardware como el software utilizado en su construcción, además de brindar una breve introducción sobre cómo utilizarlas y definir los conceptos básicos necesarios para su comprensión.

2.1. Hardware

Se desarrolló un dispositivo hardware con el cual, a través de comunicación bluetooth, es posible conectarse a la simulación que corre en la PC y realizar distintas tareas tales como el registro e identificación del operario y el manejo de la cámara del robot. A continuación, se detalla dicho sistema.

2.1.1. Componentes

- **Arduino UNO**

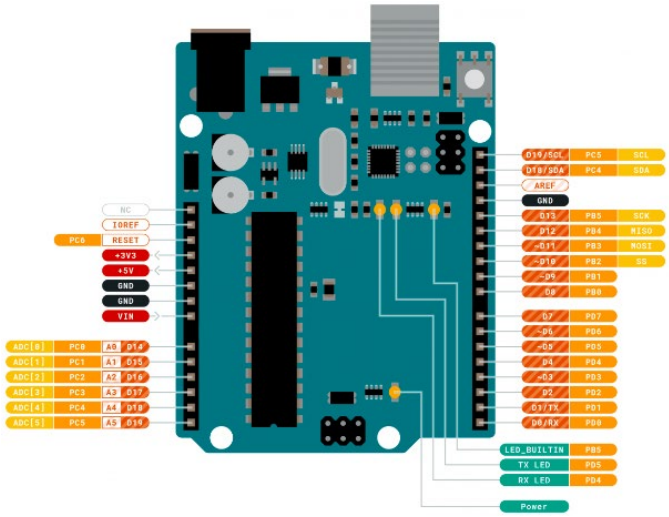


Figura 2.1.1: Esquema Arduino UNO

Arduino UNO es una placa de microcontrolador basada en el ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden usarse como salidas PWM), 6 entradas analógicas, un resonador de cerámica de 16 MHz, una conexión USB, un conector de alimentación, un cabezal ICSP y un botón de reinicio.

Este Arduino se encuentra integrado en el controlador y se encarga de recibir datos de entrada provenientes de un módulo analógico joystick y un escáner de identificación RFID RC522, luego, procesa dichos datos y los envía a través del pin de comunicación TX, información que posteriormente es recibida por un módulo bluetooth HC-05. Además, enciende o apaga unos LEDs en función de ciertos datos leídos.

- **RFID RC522**

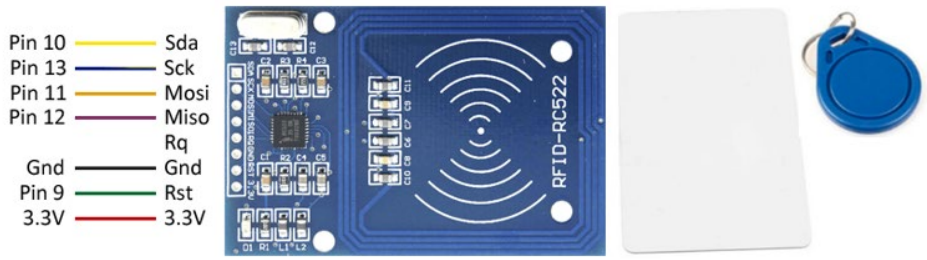


Figura 2.1.2: Módulo RFID RC522 y tarjetas de identificación RFID

El módulo RFID RC522 es un dispositivo de lectura y escritura que utiliza tecnología de radiofrecuencia para comunicarse con etiquetas/tarjetas de identificación RFID. Este dispositivo está compuesto por una antena, un microcontrolador y circuitos electrónicos adicionales. La antena es la encargada de enviar y recibir señales de radiofrecuencia a las etiquetas, mientras que el microcontrolador procesa la información recibida y la transmite a través de una interfaz de comunicación. El RC522 es compatible con varios protocolos de comunicación y puede ser utilizado en diferentes aplicaciones, como sistemas de control de acceso, inventario y seguimiento de objetos, entre otros.

En este proyecto se tomará como correcto sólo el código correspondiente al llavero mientras que el de la tarjeta es incorrecto, esto, con el fin de contrastar ambas situaciones.

- **Módulo bluetooth HC-05**

Figura 2.1.3: Módulo bluetooth HC-05

El módulo Bluetooth HC-05 es un componente de hardware que permite la comunicación inalámbrica a corta distancia entre dispositivos, como computadoras, teléfonos inteligentes, microcontroladores y otros dispositivos electrónicos. El HC-05 es un módulo Bluetooth versión 2.0+EDR (Enhanced Data Rate) que utiliza el protocolo de comunicación serial UART (Universal Asynchronous Receiver-Transmitter) para transmitir datos de forma inalámbrica a una velocidad de hasta 2.1 Mbps. El módulo HC-05 se puede configurar para funcionar en modo maestro o esclavo, lo que permite una amplia gama de aplicaciones para su uso en sistemas electrónicos. Además, el HC-05 es compatible con la mayoría de los microcontroladores, incluyendo Arduino, y se puede utilizar para controlar dispositivos electrónicos mediante la comunicación inalámbrica Bluetooth.

En este caso el módulo se configura en modo esclavo, ya que solo recibe datos del Arduino y los envía a la PC, pero no envía ningún dato al Arduino. Por ende, solo se hace uso del PIN de recepción de dicho módulo.

- **Módulo analógico joystick**

Figura 2.1.4: Módulo analógico joystick

El módulo analógico joystick es un componente comúnmente utilizado en proyectos de robótica y control de drones, ya que permite controlar el movimiento de un objeto de manera intuitiva mediante la manipulación de una palanca en dos ejes. Este módulo consta de dos potenciómetros que detectan la posición en el eje X e Y, así como un botón de acción.

Para su funcionamiento, se conecta el módulo a un microcontrolador, como Arduino, y se utilizan los valores de voltaje de los potenciómetros para controlar los movimientos del objeto. Además, el botón de acción se puede utilizar para activar o desactivar ciertas funciones del objeto controlado.

En este proyecto, se ha utilizado el módulo joystick para controlar el movimiento de la cámara del robot móvil.

2.1.2. Circuito electrónico

El circuito y conexionado se puede observar con detalle en la Figura 2.1.5

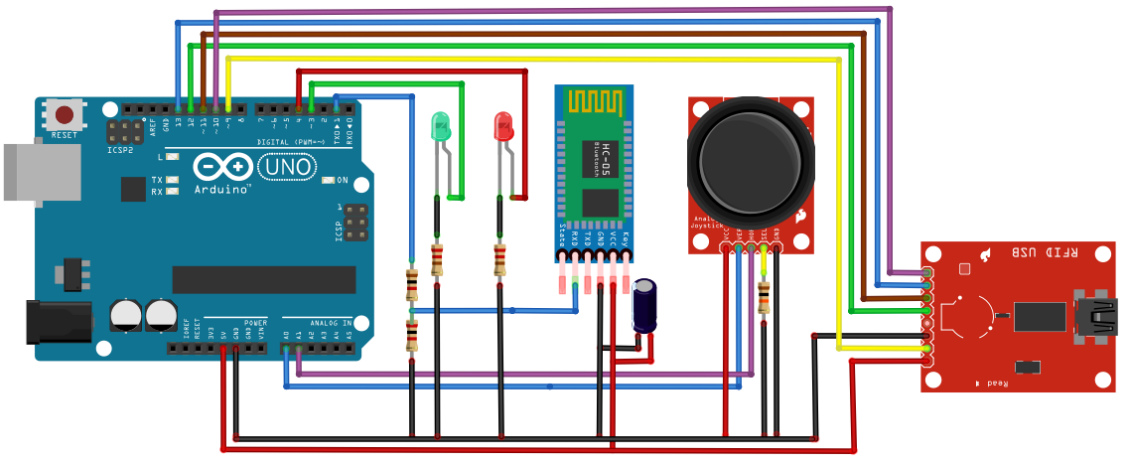


Figura 2.1.6: Componentes y conexionado correspondiente al controlador

Tal y como se mencionó, se utiliza un módulo de radiofrecuencia RFID el cual tiene como función detectar una credencial que se apoya sobre el mismo. En caso de ser una credencial no autorizada, el sistema enciende el led rojo y envía dicha señal a la simulación. En caso de ser la credencial correcta se enciende el led verde y nuevamente se envía dicha señal a la simulación.

A su vez se cuenta con un módulo analógico joystick, el cual se utiliza para comandar los movimientos de la cámara del robot, pudiendo rotar la misma en torno a dos ejes.

Estas señales se envían a la simulación que corre en una PC mediante bluetooth haciendo uso de un módulo HC-05 configurado como esclavo, ya que solo se encarga de recibir datos del Arduino y enviarlos a la PC, pero no recibe ningún dato desde la PC. El mismo tiene conectado en la entrada de alimentación un capacitor de 10 uf, con el fin de suavizar variaciones en la tensión de alimentación y filtrar ruido.

2.1.3. Carcasa

Para comandar dicho sistema de una forma fácil y cómoda se diseñó una carcasa dentro de la cual se posicionan de forma fija y segura cada componente del circuito. Dicho diseño se realizó haciendo uso del programa de diseño y modelado 3D Fusion360. El mismo se muestra en la Figura 2.1.7.

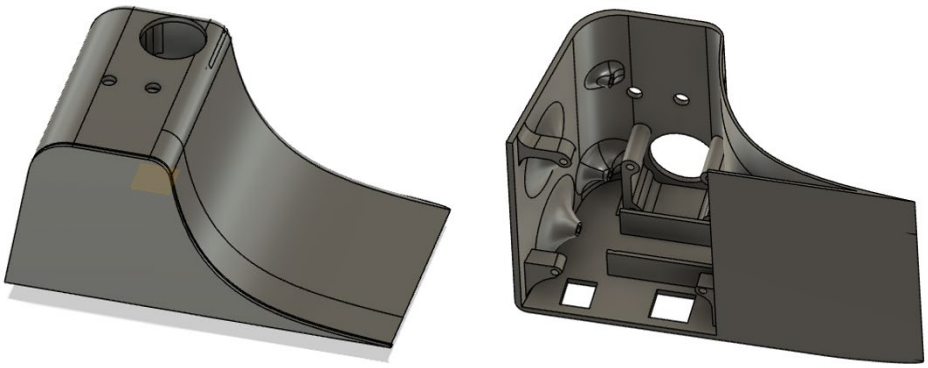


Figura 2.1.8: Modelo 3D de la carcasa del controlador

Una vez diseñado el modelo correspondiente se prosiguió con la impresión 3D del mismo y el ensamblado de cada componente. El resultado final se observa en Figura 2.1

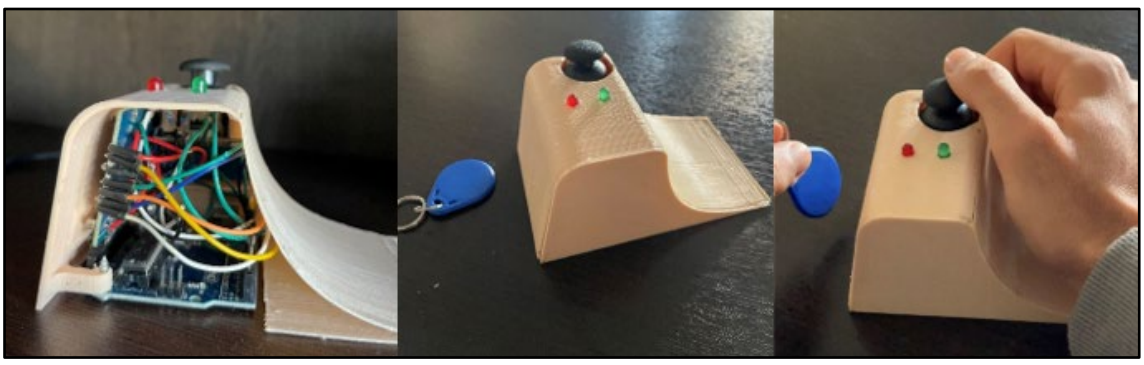


Figura 2.1.9: Resultado final del controlador. Impresión y ensamblado del mismo

2.2. Software

2.2.1. Arduino

El software correspondiente al controlador descripto en la sección de hardware fue desarrollado en el entorno Arduino IDE. El mismo se utiliza para escribir y cargar programas en placas compatibles con Arduino, pero también, con la ayuda de núcleos de terceros, se puede usar con placas de desarrollo de otros proveedores. El IDE de Arduino admite los lenguajes C y C++.

En general, dicho software se encarga de leer el valor de los potenciómetros X e Y del módulo analógico joystick, el pulsador del mismo, el valor leído por el escáner de identificación y escribir dichos valores a través del puerto serial para ser enviados a través del módulo bluetooth a la computadora. La cadena de datos enviada tiene la siguiente forma:

$$[potenciómetro_x \ , \ potenciómetro_y \ , \ pulsador \ , \ código\_indentificación \ ]$$

Ejemplo 1: $[511 \ , \ 511 \ , \ 1 \ , \ nada]$

Ejemplo 2: $[1023 \ , 0 \ , \ 0 \ , \ \ 47 \ \ 46 \ \ 70 \ \ B4]$

En el ejemplo 1 el módulo joystick se encuentra en su posición central (quieto) ya que los valores varían desde 0 hasta 1023, el pulsador se encuentra sin ser presionado y el escáner no ha leído nada.

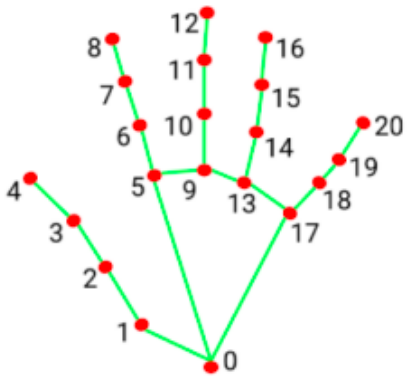
En el ejemplo 2, el potenciómetro en X se encuentra en su máximo valor, en Y en el mínimo valor, el pulsador presionado y el escáner leyendo una identificación.

El código que fue utilizado en este controlador se encuentra en Anexos [7].

2.2.2. Python - OpenCV – Mediapipe

Todo el código del lado servidor se desarrolla en Python. Para el reconocimiento de manos se utiliza Mediapipe, una librería de código abierto desarrollada por Google que proporciona una colección de herramientas para la construcción de aplicaciones de visión por computadora en tiempo real. Para la decodificación y manipulación de los frames de webcam recibidos se utiliza OpenCV, una librería de software de Machine Learning y Computer Vision de código abierto y multiplataforma.

Basado en el módulo HandTrackingModule.py desarrollado por Computer Vision Zone se realizan adaptaciones para el presente proyecto. Este módulo modificado es capaz de recibir un frame y procesar la imagen devolviendo 21 landmarks con sus coordenadas x,y,z de cada mano reconocida (las mismas están ubicadas como se muestra en la Figura 2.2.1), calcular la distancia en pixeles entre 2 landmarks, calcular la cantidad de dedos extendidos en cada mano, y obtener listas con esta información en un formato conveniente, incluyendo una diferenciación entre mano izquierda y derecha. Además, nos permite visualizar las manos reconocidas junto con un enmarcado de cada una como se muestra en la Figura 2.2.2.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figura 2.2.1: Marcas de las articulaciones de la mano

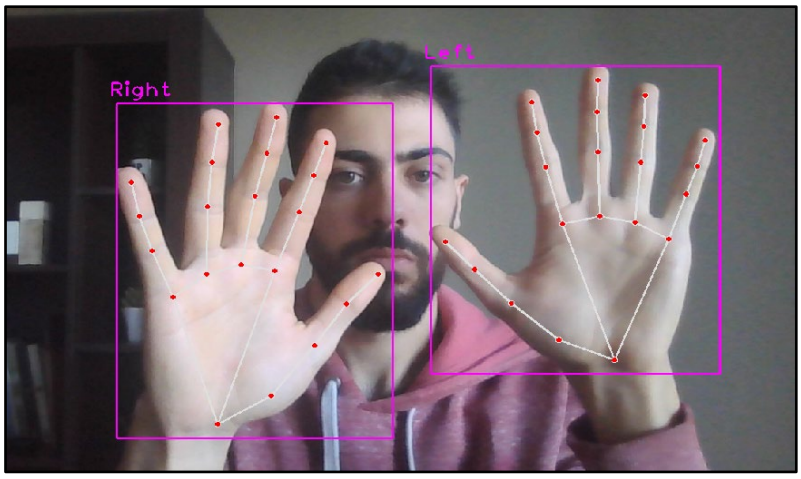


Figura 2.2.2: Reconocimiento de mano izquierda y mano derecha.

Por otro lado, el módulo principal del lado servidor es HandTracking.py

El mismo hace uso del módulo modificado mencionado anteriormente. Se encarga de establecer una conexión TCP con el cliente, recibir un frame del mismo, procesarlo para obtener la información de coordenadas de las manos junto con el comando correspondiente, y enviar esta información de vuelta al cliente.

Entre otras funciones implementa también el cálculo de los fps de envío de información, promediando los últimos 10 envíos para reducir la sensibilidad del cálculo.

2.2.3. Docker

Con la finalidad de realizar el procesamiento del servidor en el Cloud, se construye una imagen de Docker.

Docker es una plataforma de contenedores que permite a los desarrolladores empaquetar y distribuir aplicaciones en un entorno portátil y seguro. Los contenedores Docker permiten a los desarrolladores encapsular todas las dependencias y configuraciones de una aplicación en un único paquete, lo que facilita su distribución y despliegue en diferentes entornos, que en este caso será una instancia en el Cloud.

Uno de los componentes fundamentales de Docker es el Dockerfile, que es un archivo de texto que contiene las instrucciones necesarias para construir una imagen de contenedor Docker. Un Dockerfile se utiliza para definir la configuración y las dependencias de una aplicación, así como para especificar cómo se debe ejecutar la aplicación dentro del contenedor.

El Dockerfile desarrollado incluye las librerías del módulo de python y está basada en una imagen de python3. Se anexa el mismo junto con las instrucciones para utilizarlo en la nube.

2.2.4. Cloud Computing

El Cloud Computing permite a los usuarios acceder a una amplia variedad de recursos informáticos, como servidores, almacenamiento, bases de datos, herramientas de desarrollo y aplicaciones, desde cualquier lugar del mundo y en cualquier momento.

Amazon Web Services (AWS) es uno de los principales proveedores de servicios en la nube, y ofrece una amplia gama de servicios de infraestructura y plataforma como servicio, que permiten a los desarrolladores y empresas escalar sus aplicaciones y servicios de manera rápida y fácil.

En particular para este proyecto es de interés el servicio EC2 (Elastic Compute Cloud), el cual permite a los usuarios lanzar y escalar instancias de servidores virtuales en la nube. En la versión de control por celular se implementa el servidor en la nube, para aliviar el procesamiento en el dispositivo móvil. Se anexan los requisitos técnicos de la instancia a lanzar en la nube, junto con los programas a instalar y el proceso para correr el script del servidor desde la misma.

2.2.5. Unity

La simulación fue desarrollada utilizando el entorno de desarrollo Unity 2021.3.9f1 y 2021.3.17f1. Se realizan dos versiones. Una para control híbrido gestual+hardware, y otra para control gestual puro.

Es importante mencionar que todos los scripts utilizados desde Unity están escritos en C#.

2.2.5.1. Versión Control Híbrido

Entre las particularidades de esta solución se encuentra el uso del dispositivo de Hardware mencionado previamente y una de las manos del usuario. El registro se realizará por RFID y el control del robot por joystick combinado con una de las manos.

Requisitos:

-Notebook.

-Dispositivo de Hardware.

-Credencial personal

El servidor de python se encontrará en la máquina local, al igual que el cliente de Unity.

Esta versión cuenta con 3 escenas:

1- Escena “Welcome”

Escena en la cual se le da la bienvenida al usuario y se le pide que ingrese al menú donde deberá registrarse. Esta interfaz es manejada mediante los gestos de la mano mediante handtracking, como se mencionó anteriormente. Es decir, que se debe “presionar” un botón moviendo la mano en el aire. Una vez que se presiona el botón, comienza a cargar la próxima escena. Se visualiza el progreso de carga de la próxima escena mediante una barra de carga. La siguiente imagen representa dicha escena.

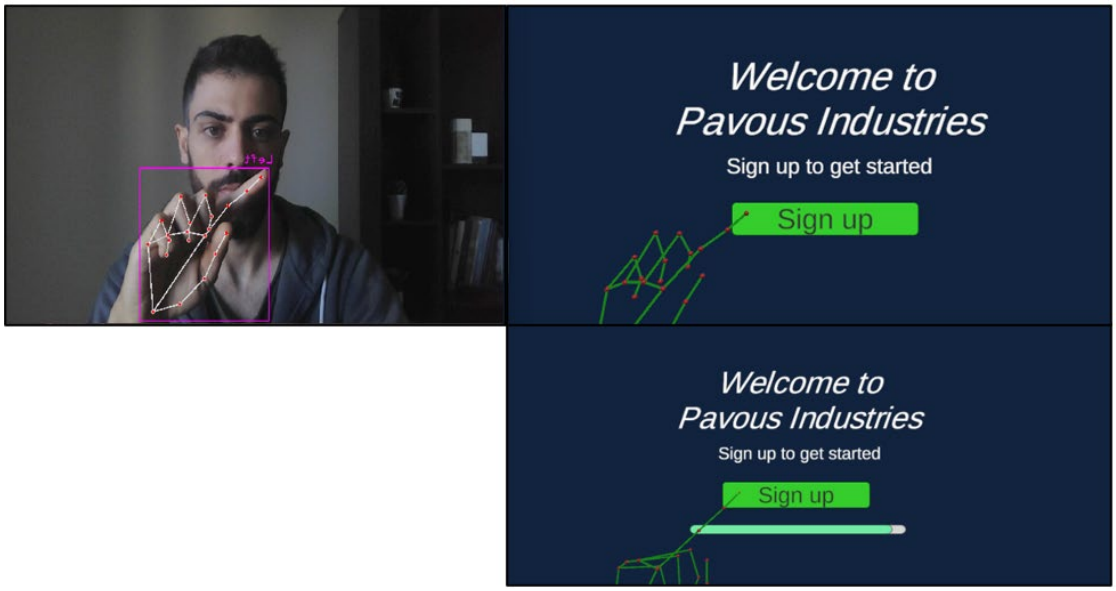


Figura 2.2.3: Escena 1 "Welcome"

Los objetos principales de esta escena son:

$Canvas$, que está formado por el texto correspondiente mostrado en la escena y el botón de *sign up*. Dicho botón, posee un *Cube* el cual actúa como colisionador y permite que un objeto *Point* perteneciente a *Hand* pueda activar el mismo al colisionar.

$Manager$, posee los scripts TCPClient y HT. TCPClient gestiona la comunicación TCP con el servidor. Toma un frame de la webcam, lo envía al servidor y luego recibe los datos de comandos y landmarks obtenidos por el script de python HandTracking.py descripto anteriormente, para finalmente procesarlos y ponerlos a disponibilidad de otros scripts. HT actualiza la posición de los objetos $Points$ pertenecientes a $Hands$.

En resumen, se toma la imagen desde Unity, se envía a Python, el misma procesa la imagen para reconocer la mano, y devuelve los datos a Unity para actualizar la posición de los Points.

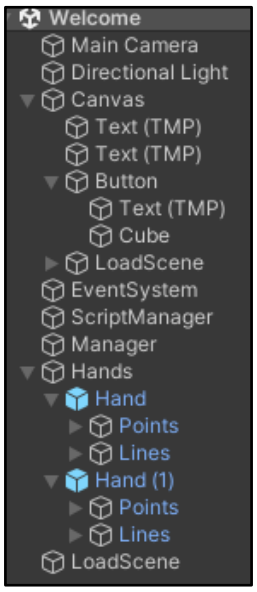


Figura 2.2.4: jerarquía de objetos escena Welcome

Los códigos detallados utilizados se encuentran en Anexos [2].

2- Escena “Register”

Luego de presionar el botón “sign up” en la escena “Welcome”, la simulación avanza hacia la escena “Register”, escena en la cual el operario debe escanear su credencial para verificar si el mismo posee permiso para controlar el robot o no.

El registro se realiza mediante el controlador hardware con RFID detallado en la sección 2.1. Dicho controlador permite escanear la credencial, e identificar si la misma posee o no el permiso adecuado, y le envía dicha información a la simulación en la PC.

A continuación, en la

Figura 2.2.3 se pueden observar las distintas instancias que forman parte de la escena “Register”. Al comienzo, una instancia que solicita al usuario escanear la credencial. Luego, una instancia en la cual si la credencial en cuestión no se encuentra autorizada para comandar el robot muestra que el acceso es denegado, y el controlador hardware enciende un led rojo. En caso en que la credencial escaneada se encuentre autorizada, entonces se ejecuta la instancia que muestra un registro exitoso, se enciende un led verde en el controlador y comienza la carga de la tercer y última escena.

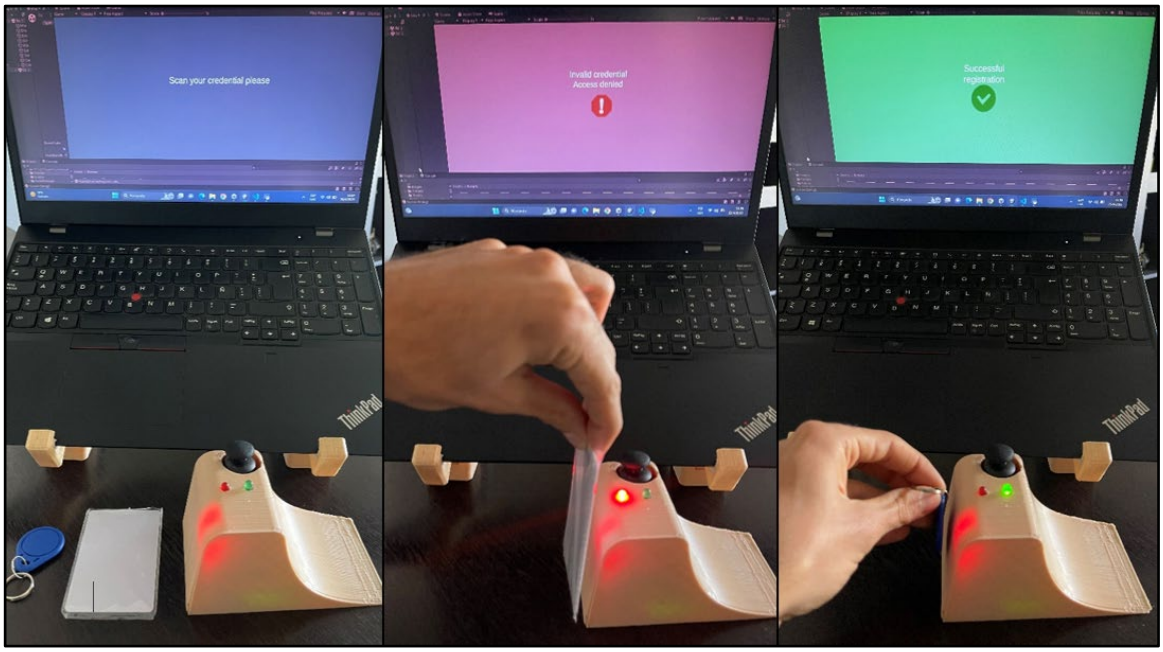


Figura 2.2.5: Instancias de la escena "Register". Instancia que solicita escanear la credencial - Instancia de acceso denegado - Instancia acceso permitido

En la Figura 2.2.6 se observa el árbol jerárquico de objetos que forman parte de la escena en cuestión.

Los elementos principales que se destacan de esta escena son:

$CANVAS$, el cual posee como hijos las tres instancias ($Canvas1, Canvas2\ y\ Canvas3$) que se recorren dentro de la escena (solicitar credencial, credencial denegada y registro exitoso).

$SerialManager$ se encarga de establecer la comunicación bluetooth con el controlador y recibir los datos del mismo.

$SerialBTManager$, se encargan de procesar los datos recibidos por bluetooth.

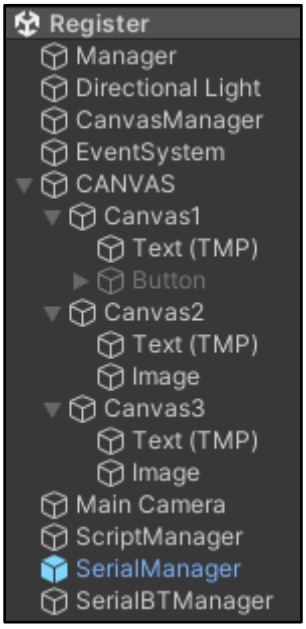


Figura 2.2.6: Jerarquía de objetos - escena Register

3- Escena "Town"

Finalmente, en caso de que se permita el acceso en la escena "Register", se accede a la escena "Town". En dicha escena se encuentra el robot que queremos comandar, se trata de un robot móvil dotado de un sistema de orugas mecánicas. En la parte superior posee un brazo robótico tipo serie con una cámara de video en el extremo. Dicho robot se encuentra en un entorno del tipo industrial representativo de las posibles tareas que realizaría dicho robot. En la aplicación real el robot debe ir construyendo un modelo de su entorno con mayor o menor detalle en función de los sensores disponibles. Este sería transmitido a la consola del operario que lo controla de forma remota.

La traslación y rotación del robot es comandada por los movimientos de la mano, haciendo uso de *HandTracking,* mientras que el movimiento de la cámara es comandado por el controlador bluetooth mencionado anteriormente y el cual se detalla en la sección 2.1. A su vez, se cuenta con la posibilidad de cambiar la perspectiva de la escena presionando la barra espaciadora.

Figura 2.2.7: Vistas de las distintas cámaras en la escena "Town"

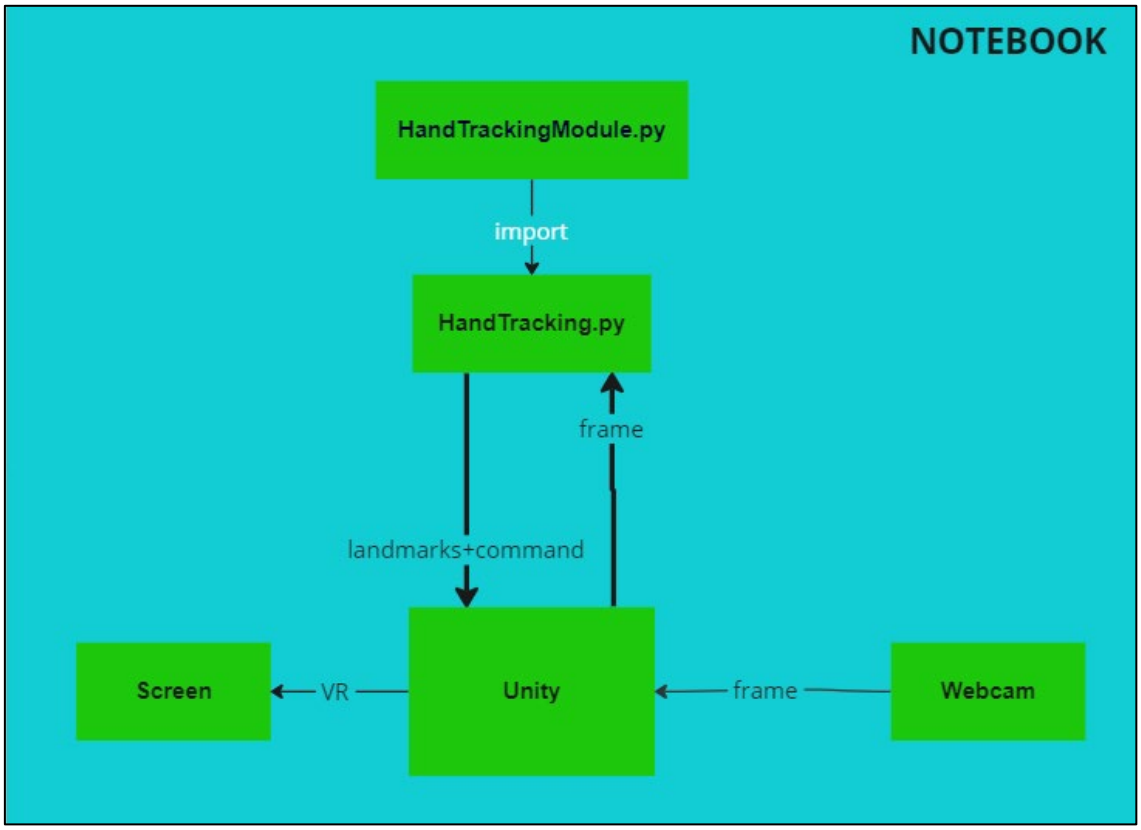


Figura 2.2.8: Esquema de comunicaciones escena "Town"

En la Figura 2.2.9 se observa el árbol jerárquico de objetos que forman parte de la escena en cuestión.

Los elementos principales que se destacan de esta escena son:

Map_v2, es el mapa de la escena, es decir, las calles, edificios, contenedores, etc. El mismo fue obtenido del *"Asset store"* y es un entorno industrial representativo de los posibles entornos en los cuales este robot trabajaría en el mundo real.

$androrobot$, modelo 3D del robot móvil utilizado. El mismo fue obtenido de Turbosquid, plataforma de modelos 3D gratuitos y de libre acceso. Dentro de *Body* se encuentran todos los componentes que forman parte del mismo. Como se observa, el objeto *androrobot* también posee como hijos a *Main Camera* y *Hands*, esto, con el objetivo de que los mismos se muevan junto con el robot cuanto éste se desplaza, y así, permanezcan en el mismo punto del espacio relativo al robot.

$SerialManager$ y $SerialBTManager$, cumplen la misma función que en la escena *"Register"* pero en este caso con la finalidad de controlar los movimientos de la cámara del robot mediante el módulo joystick.

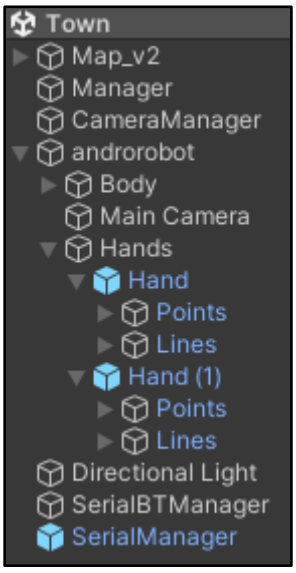


Figura 2.2.9: Jerarquía de objetos - escena Town

2.2.5.2. Versión Control gestual completo

Esta solución hace uso de control gestual simultáneo con ambas manos. El registro se realiza por escaneo de una credencial con realidad aumentada. La principal diferencia en la interfaz es que se utiliza un celular en lugar de una Notebook.

Requisitos:

- Celular
- VR Box
- Conexión a Internet
- Credencial personal

El servidor de python se encontrará funcionando en el Cloud, y el cliente de Unity en el celular.

Esta versión también cuenta con 3 escenas:

1- Escena "Welcome"

Se da la bienvenida al usuario y se le pide que ingrese al menú donde deberá registrarse. A diferencia de la escena 1 del sistema Híbrido, en esta no hay control gestual y se maneja desde el celular. En la misma debe indicarse también la IP del servidor Cloud que procesará el HandTracking. La siguiente imagen representa dicha escena.

Figura 2.2.10: Escena 1 "Welcome"

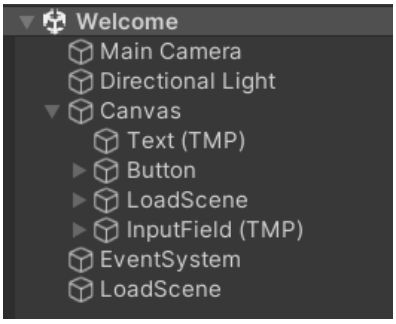


Figura 2.2.11: Jerarquía de objetos - Escena "Welcome"

Los objetos principales de esta escena son similares a la versión Híbrida, con excepción de que se removió el control por HandTracking y se agregó un objeto InputField que almacene la IP ingresada. De esta manera no es necesaria la comunicación con el servidor en esta etapa.

2- Escena “Register”

Luego de presionar el botón “Set IP” en la escena “Welcome”, la simulación avanza hacia la escena “Register”, escena en la cual el operario debe escanear su credencial para verificar si el mismo posee permiso para controlar el robot o no.

El registro se realiza mediante realidad aumentada, que coloca una cruz sobre la credencial de ser escaneada del lado incorrecto o tratarse de un ingeniero no autorizado. Si en cambio escanea la credencial correcta coloca una tilde verde y consulta la contraseña para acceder al control. El botón torna de color rojo o verde en función del uso de la contraseña correcta, y si torna verde comienza a cargar la próxima escena. Esta etapa de Realidad Aumentada es posible gracias al uso de Vuforia.

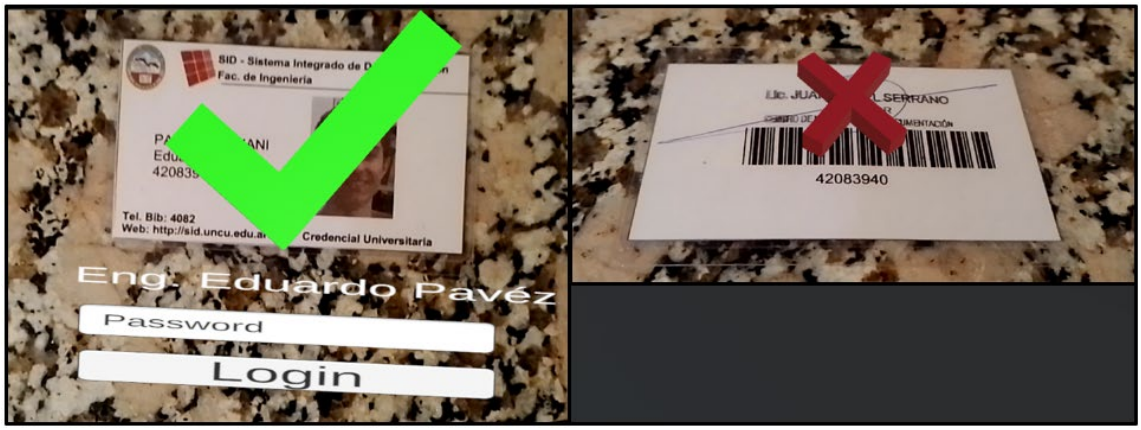


Figura 2.2.12: Escena “Register” mediante realidad aumentada. Caso de identificación exitosa vs caso denegado

3- Escena “Town”

Similar al caso sistema Híbrido, se permite ingresar a esta escena luego de un registro correcto. El robot y entorno de trabajo simulados son los mismos. El entorno en la aplicación real se obtendría de un modelo construido por el robot según los sensores que tenga disponibles, y sería transmitido al operario que lo controla de forma remota.

Las principales diferencias de esta escena con la versión anterior son el control gestual completo y la vista adaptada para el uso de VR Box. Tanto los movimientos del robot como los de su brazo robótico son controlados por gestos. Llegados a esta escena el celular debe ser colocado en el VR Box, de forma que el control sea inmersivo. Se permite al operario visualizar el entorno construido por el robot sin mover este último, colocando el punto de vista por encima del mismo, similar a la vista original de la versión anterior. Además, se cuenta con la posibilidad de mover la cámara al girar tu propio cuerpo.

Esta escena se construye gracias al Google Cardboard XR Plugin, que posibilita la vista de VR Box.

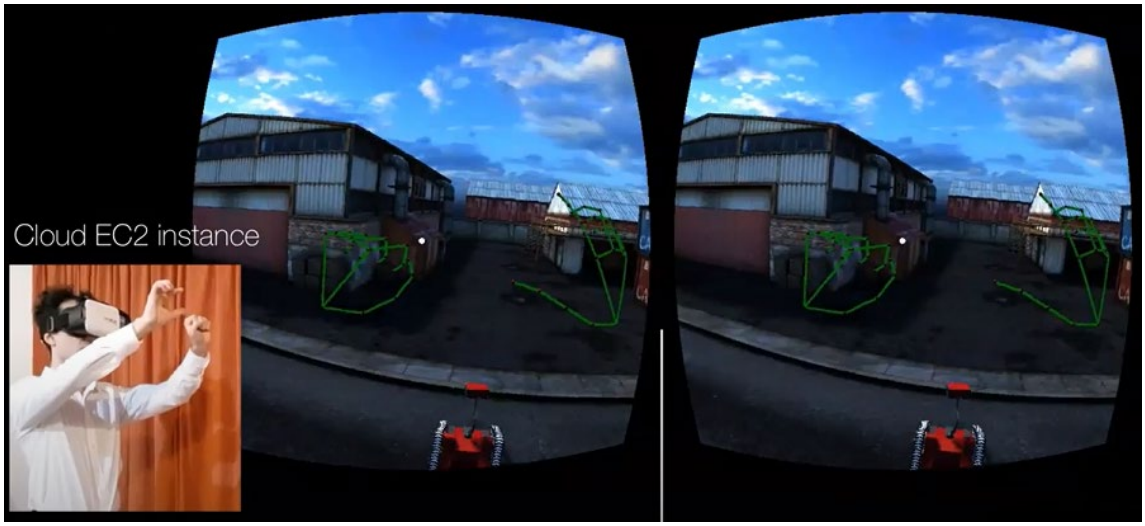


Figura 2.2.13: Vista en VRBox de la escena "Town"

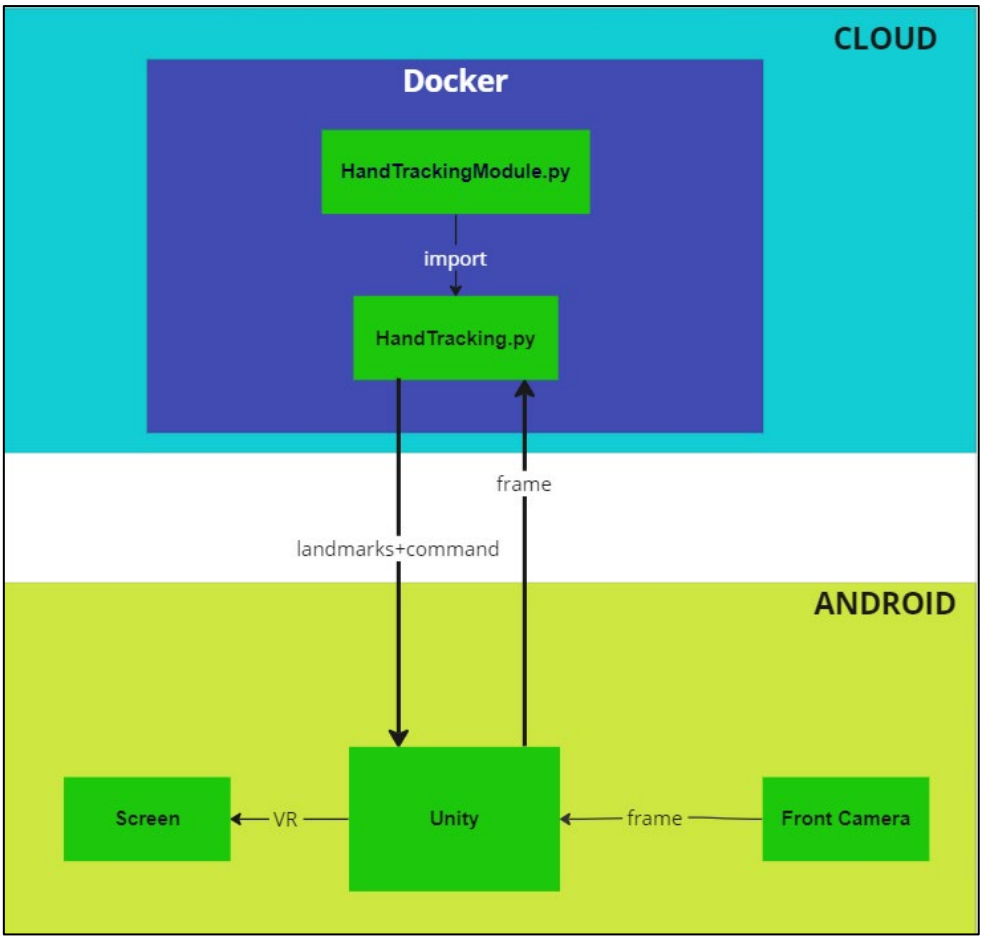


Figura 2.2.14: Esquema de comunicaciones escena "Town"

En la Figura 2.2.15 se observa el árbol jerárquico de objetos que forman parte de la escena en cuestión.

Los elementos principales que se destacan de esta escena son:

Map_v2: es el mapa de la escena, es decir, las calles, edificios, contenedores, etc. El mismo fue obtenido del *"Asset Store"* y es un entorno industrial representativo de los posibles entornos en los cuales este robot trabajaría en el mundo real. modelo 3D del robot móvil utilizado.

Androrobot: El mismo fue obtenido de Turbosquid, plataforma de modelos 3D gratuitos y de libre acceso. Dentro de *Body* se encuentran todos los componentes que forman parte del mismo. Como se observa, el objeto *androrobot* también posee como hijos a *Main Camera* y *Hands*, esto, con el objetivo de que los mismos se muevan junto con el robot cuanto éste se desplaza, y así, permanezcan en el mismo punto del espacio relativo al robot.

Figura 2.2.15: Jerarquía de objetos - escena Town

Hands: Este posee todos los puntos y líneas que construyen la representación virtual de las manos reales, teniendo así Virtualidad Aumentada.

VR Camera: La cámara que permite la rotación del visor con las rotaciones del celular. De esta forma la experiencia es aún más inmersiva.

Manager_Hands: Se encarga de asignar las posiciones y comandos obtenidos via TCP a las manos y al desplazamiento del robot.

Manager_VR: Dado que las primeras dos escenas requieren el uso de la pantalla del celular estándar, la inicialización de la visión para VR Box debe realizarse en esta escena. Este es el rol de este Manager, que gestiona todas las configuraciones para permitir la nueva modalidad de la vista del celular.

3. Resultados

A continuación, se presenta un video de la simulación completa, en el que se observa en el primer parte la versión PC y en la segunda parte la versión móvil.

https://youtu.be/Dqqu_yh6Uw0

4. Conclusiones

Habiendo desarrollado dos simulaciones, una para PC y otra para el móvil, procedemos a contrastarlas analizando las ventajas y desventajas de cada una.

Simulación PC con hardware específico:

- Ventajas:
  - Permite una mayor seguridad al utilizar una credencial física con un identificador RFID interno.
  - Permite el control por joystick, el cual puede resultar más familiar para el operario. Además, el mismo permite trabajar en una posición cómoda durante largos períodos de tiempo.
- Desventajas:
  - Requiere tener disponible una Notebook y el hardware específico, que deben transportarse a donde desee trabajar el operario.

Simulación por Celular:

- Ventajas:
  - Es más económica y accesible. Un operario trabajando de forma remota solo necesita acceder a un VR Box de bajo precio y hacer uso de su propio celular para estar en condiciones de maniobrar el robot.
  - Permite apalancarse en el Cloud para el procesamiento, por lo que a bajo precio se puede acceder a un poder computacional altísimo para realizar el handtracking.
- Desventajas:
  - La credencial utilizada para registro podría falsificarse tan solo a partir de una imagen de la misma. Por eso se opta por solicitar también una contraseña.
  - Si el sistema de IA utilizado para el handtracking falla en clasificar las manos como derecha o izquierda, el envío de un comando erróneo es inevitable. Se depende de la robustez del mismo.

Más allá de las ventajas específicas de cada sistema, es importante conocer la preferencia del operario, que podrá encontrar más intuitivo el uso de uno o del otro.

5. Anexos

[1] Repositorio del proyecto

https://github.com/EduPav/VR_HandTracking_Archimedes

[2] Códigos en C#. Simulación versión PC:

https://drive.google.com/drive/folders/1Z-JB0rukRJdqOhAUKR9OBGvpA7V-HxT1?usp=sharing

[3] Proyectos en Unity. Versión PC y Celular

https://drive.google.com/drive/folders/1AGd0dnqLf6OTxdLysQvqMhknENuwqijQ?usp=sharing

[4] Instrucciones de construcción del Docker y deploy al Cloud

https://github.com/EduPav/VR_HandTracking_Archimedes/blob/main/Docker_and_Cloud_usage.md

[5] Vuforia SDK

https://developer.vuforia.com/downloads/sdk

[6] Google Cardboard XR Plugin Quickstart for Unity

https://developers.google.com/cardboard/develop/unity/quickstart

[7] Código Hardware Arduino IDE:

https://drive.google.com/file/d/1EyKmrSVjSo4Fabw4B2ZEq1mg4YGNzWEz/view?usp=sharing

6. Referencias

[1] Modelo 3D androrobot:

https://www.turbosquid.com/es/3d-models/androrobot-3ds-free/709206

[2] Mapa industrial virtual:

https://assetstore.unity.com/packages/3d/environments/industrial/rpg-fps-game-assets-for-pc-mobile-industrial-set-v3-0-101429

[3] 3D Hand Tracking with UDP communications in Unity

3d Hand Tracking in Virtual Environment | Computer Vision

[4] Hand Tracking Module

https://github.com/cvzone/cvzone