UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

Trabajo de Suficiencia Profesional

Aplicación móvil de realidad aumentada de un equipo mecatrónico para atención formativa en la Industria 4.0 de un Instituto Tecnológico Superior

Para obtener el Título Profesional de Ingeniero Mecatrónico

Elaborado por:
Roger Juan Paredes Alta

0009-0003-6203-6295

Asesor
Dr. Aurelio Marcelo Padilla Ríos

0009-0007-6270-6171

LIMA – PERÚ

2024

Citar/How to cite	Paredes Alta [1]
Referencia/Reference Estilo/Style: IEEE (2020)	[1] R. Paredes Alta, "*Aplicación móvil de realidad aumentada de un equipo mecatrónico para atención formativa en la Industria 4.0 de un Instituto Tecnológico Superior*" [Trabajo de Suficiencia Profesional]. Lima (Perú): Universidad Nacional de Ingeniería, 2024.

Citar/How to cite	(Paredes, 2024)
Referencia/Reference Estilo/Style: APA (7ma ed.)	Paredes, R. (2024). *Aplicación móvil de realidad aumentada de un equipo mecatrónico para atención formativa en la Industria 4.0 de un Instituto Tecnológico Superior*. [Trabajo de Suficiencia Profesional, Universidad Nacional de Ingeniería]. Repositorio institucional Cybertesis UNI.

Índice

Índice de Figuras

Índice de Tablas

Resumen

El presente trabajo es la propuesta de desarrollo de una aplicación de realidad aumentada aplicada a la formación técnica en un instituto de educación superior el cual aplica en la especialidad de Mecatrónica, la aplicación de los conceptos clave de la cuarta revolución industrial, dentro de estos conceptos clave se trata el tema de la conectividad, donde términos como el IoT (Internet de las cosas) cobra vital relevancia, asimismo, las aplicaciones de realidad aumentada apoyan términos de virtualización de procesos, lo que permite adicionar información a lo que los sentidos como la vista pueden percibir. La aplicación propuesta se desarrolla en ocho etapas, que parten desde la concepción de la arquitectura de desarrollo de la aplicación, seguido del desarrollo de las comunicaciones a través de protocolos como OPC-UA y MQTT, luego se desarrolla el objeto 3D que se utiliza luego en la aplicación, se prosigue con la generación de la licencia en Vuforia para establecer el enlace con una imagen objetivo (Image Target) cargada en el servicio de la misma, esto permite proseguir con el desarrollo de la aplicación, partiendo por la generación del objeto 3D a partir de la imagen objetivo, luego el desarrollo de la interfaz y navegación en la aplicación, para finalmente establecer la comunicación MQTT con la aplicación para conseguir mostrar la información proveniente del equipo en estudio. El resultado es una aplicación disponible para sistemas operativos Android, que permite mostrar en una sección el objeto 3D adicionado a la proyección que muestra la cámara principal. En otra sección se adiciona la información que se encuentra compartida por el equipo a través de la comunicación desarrollada. Y finalmente, un botón que permite el cerrar y salir de la aplicación. Con ello se cumple con los objetivos contemplados para este trabajo.

Palabras Clave: Realidad Aumentada, Aplicación móvil, Industria 4.0, sistema operativo Android.

Abstract

This project presents the proposal for the development of an augmented reality application applied to technical training in a higher education institute which applies in the specialty of Mechatronics, the application of the key concepts of the fourth industrial revolution, within these concepts. key is the issue of connectivity, where terms such as the IoT (Internet of Things) become vitally relevant, likewise, augmented reality applications support terms of process virtualization, which allows adding information to what the senses such as sight can perceive.

The proposed application is developed in eight stages, which start from the conception of the application development architecture, followed by the development of communications through protocols such as OPC-UA and MQTT, then the 3D object is developed that is then used In the application, the license generation continues in Vuforia to establish the link with a target image (Image Target) loaded in its service. This allows us to continue with the development of the application, starting with the generation of the object. 3D starting from the target image, then the development of the interface and navigation in the application, to finally establish MQTT communication with the application to display the information coming from the equipment under study. The result is an application available for Android operating systems, which allows showing in a section the 3D object added to the projection shown by the main camera. In another section, the information that is shared by the team through the communication developed is added. And finally, a button that allows you to close and exit the application. This meets the objectives contemplated for this work.

Keywords: Augmented Reality, Mobile application, Industry 4.0, Android operating system.

CAPÍTULO I. Planteamiento del Problema

1.1. Antecedentes Investigativos

Gracia (2020)[1], en su trabajo de investigación, desarrolla una aplicación de realidad aumentada adaptable y ajustable en diversos ambientes de industria, busca reducir la mala operación de equipamiento por parte de usuarios nuevos, como objetivo propone mejorar el tiempo de vida útil y aumentar los tiempos de reposición o modernización de los equipos. Para la implementación utiliza herramientas de software como Unity y Vuforia, con ellas genera una aplicación que se basa en 3 secciones de interacción: Información, Instrucciones, Mantenimiento; en la primera muestra la información técnica del equipamiento en análisis, en la segunda informa de manera concisa la instalación y operación para acciones frecuentes con el equipo, finalmente en la tercera incorpora información referencial de diferentes errores y posibles acciones de diagnóstico para el equipamiento. Debido a la coyuntura del año de elaboración del trabajo, realizó la evaluación de su aplicación en un entorno básico con electrodomésticos diversos consiguiendo resultados positivos, logrando identificar y mostrar la información respectiva de cada equipamiento. Concluye que con la aplicación se puede conseguir la reducción de accidentes y lesiones dado que el usuario evitará cometer actos que desconozca en la operación del equipamiento en la industria. Asimismo, en el contexto educativo, concluye que la aplicación propuesta, previene la exposición al riesgo de los alumnos y también se consigue reducir el gasto institucional en la reposición de unidades, acompañado de una reducción en el impacto medioambiental.

[1] Gracia, R. (2020) Aplicación de la realidad aumentada para el mantenimiento de equipos industriales [Trabajo de Fin de Grado, Universidad Pontificia Comillas]. https://repositorio.comillas.edu/jspui/handle/11531/41255

Aranda y Gómez (2022)[2], proponen el desarrollo de una aplicación móvil de realidad aumentada para la capacitación del personal de producción de una empresa de fabricación de productos plásticos, donde identifica como problemática en la comprensión, el tempo alto de evaluación y calificación, baja satisfacción del personal capacitado y el número de capacitaciones. Como objetivo propone, el uso de una aplicación móvil de realidad aumentada bajo la metodología Mobile – D para la mejora de las capacitaciones del personal del área. Su solución denominada "C-OPP" utiliza herramientas de software como Unity Hub, Euforia, así como los servicios de Firebase como base de datos, esta aplicación contiene un sistema de navegación basadas en pantallas denominadas: Splash, Login, Selección de funciones y selección de capacitación, en esta última se accede a las capacitaciones que estén activas para poder llevar a cabo el proceso de capacitación. Concluye que logra reducir el tiempo de evaluación de capacitación para los colaboradores en un 47%, mejora la satisfacción del personal, reduce en un 4% los tiempos de calificación de las evaluaciones para finalmente mejorar en un 24% la reducción de capacitaciones necesarias por el personal, con lo que garantiza eficiencia y productividad en el área.

Cruz (2023)[3], implementa un sistema SCADA bajo un contexto de implementación de tecnologías IOT con apoyo de una aplicación de realidad aumentada. Identifica como problemática los inconvenientes que se suelen presentar en el monitoreo de proceso industriales dentro de los cuales resalta la omisión de la identificación de alarmas debido a un exceso de información lo que genera errores en los operarios para intervenir. Por ello su objetivo es implementación de un sistema SCADA utilizando heramientas de Industria 4.0 e internet de las cosas. Dentro de su solución utiliza las plataformas y equipos como:

[2] Aranda, E. y Gómez E. (2022) Aplicación móvil con realidad aumentada para mejorar el proceso de capacitación en el área de producción de la empresa OPPFILM S.A. [Tesis de Titulación, Universidad Autónoma del Perú]. https://repositorio.autonoma.edu.pe/handle/20.500.13067/1954

[3] Cruz, J. (2023) Implementación de un sistema SCADA en el ámbito de la Industria 4.0 e IOT [Trabajo de titulación, Universidad Técnica de Ambato]. https://repositorio.uta.edu.ec/handle/123456789/39216

Unity, Visual Studio, Vuforia, TIA Portal, Meta 2 development kit, Estación de automatización; con estas consigue crear dos aplicaciones, una principal dedicada a las gafas y otra orientada a dispositivos móviles, presentando las aplicaciones una navegación similar. La comunicación entre las aplicaciones y la estación de automatización se basa en una arquitectura de comunicación que utiliza el protocolo MQTT para la comunicación máquina a máquina y protocolos TCP/IP para la comunicación con la estación. De su trabajo concluye el logro de la implementación del sistema SCADA con la utilización de tecnologías asociadas a la Industria 4.0 e internet de las cosas, dentro de su validación de proyecto obtiene un 58% de aceptación por parte de los usuarios confirmando el funcionamiento y cumplimiento de las tareas del monitoreso y control industrial.

Chicaiza y Palacios (2024)[4], desarrollan una aplicación de realidad aumentada para el monitoreo de estados de dispositivos en un módulo de procesado denominado MPS. En donde, primero realizan un monitoreo en tiempo real de estados del equipo mediante una tarjeta ESP32, luego recopilan la información en una base de datos en Fregase, seguido de un tratamiento de la información para finalmente incorporar a la aplicación de realidad aumentada, consiguen comunicar un banco de pruebas del sistema MPS con su aplicación de realidad aumentada y obtienen valores de estados en tiempo real.

1.2. Descripción de la Realidad Problemática

La Institución Tecnológica Superior de estudio ofrece, dentro de su oferta educativa, la formación profesional técnica en la familia ocupacional de Electrotecnia y dentro de ella, la carrera técnica de Mecatrónica Industrial.

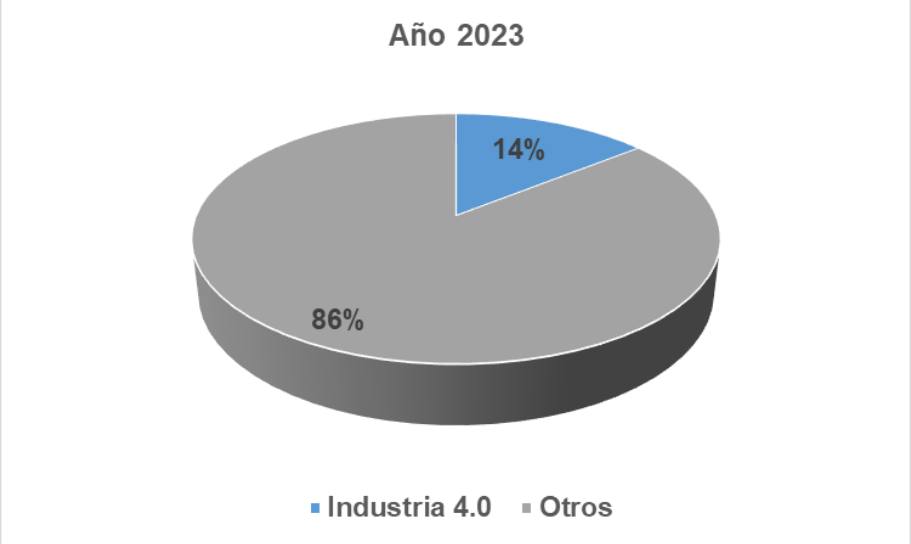
Para cumplir con la formación, cuenta con equipamientos mecatrónicos que involucran a los estudiantes en diferentes conceptos claves que aborda la especialidad.

[4] Chicaiza, A. y Palacios B. (2024) Realidad aumentada para el monitoreo del estado operativo de diferentes dispositivos comúnmente usados en el MPS [Trabajo de titulación, Universidad Politécnica Salesiana]. http://dspace.ups.edu.ec/handle/123456789/27973

Los cuales permiten, desde una experiencia práctica, conocer y dominar temas relacionados a la automatización industrial. Para que luego los estudiantes puedan aplicarlo en las diferentes empresas donde se desempeñan.

Tanto la coyuntura nacional como internacional, acompañados del avance tecnológico global, generan la necesidad de una constante actualización de contenidos curriculares de formación en la especialidad. Por lo que, en el presente año, se introdujeron conceptos de Industria 4.0 dentro de la formación del último semestre de estudios.

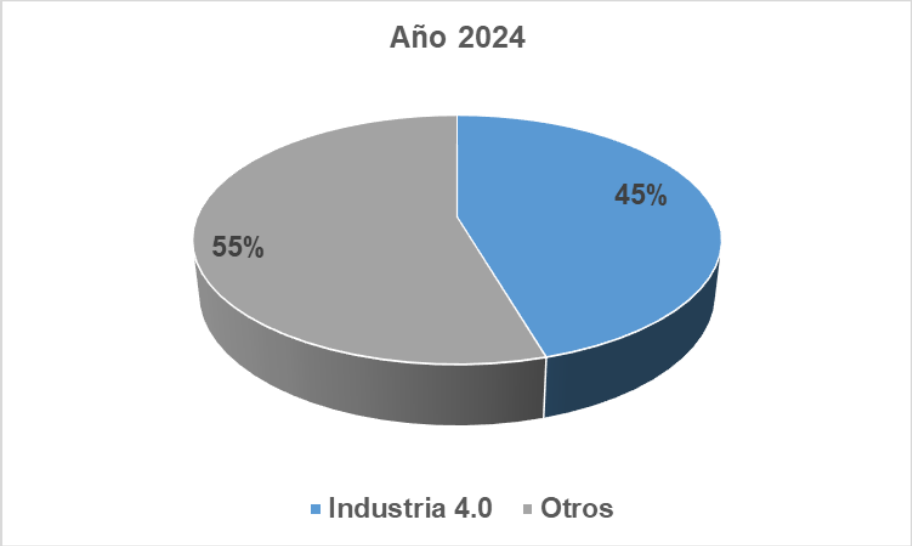
Figura 1: Cantidad de tareas en Industria 4.0 vs otros temas – Año 2023



En la Figura 1, se aprecia la comparación entra la cantidad total de tareas de tópicos de Industria 4.0 que se desarrollaron hasta el año 2023 en la formación de Mecatrónica Industrial, en la cual de un 100% de tareas, sólo el 14% abarcan tareas en Industria 4.0.

En la Figura 2, se aprecia la comparación que se viene aplicando a partir del presente año, donde de un total de 100% de tareas, el 45% tratan en temas de Industria 4.0. Con respecto a la Figura 1, se aprecia el incremento significativo para este año respecto a la aplicación de tareas en Industria 4.0.

Figura 2: Cantidad de tareas en Industria 4.0 vs otros temas - Año 2024



Este incremento de tareas en relación con la Industria 4.0 implica la incorporación de nuevos temas en la formación profesional de la especialidad de Mecatrónica Industrial. En la

Tabla **1**, se puede verificar, lo incorporado en el año 2024 en tareas de Industria 4.0 los diversos tópicos listados, de entre los cuales se puede resaltar la incorporación de temas claves como realidad aumentada (RA) e internet industrial de las cosas (IIOT).

Tabla 1: Temas de Industria 4.0 aplicados desde el año 2024

Tema Industria 4.0	Tarea para desarrollar
RFID	Instalar y comunicar módulo RFID con equipo mecatrónico.
Impresión 3D	Modelar e imprimir objeto 3D.
OPC-UA	Disponer información de equipo mecatrónico mediante OPC-UA
IIOT	Comunicar equipo mecatrónico a través de NODE-RED y MQTT.
Realidad Aumentada	Implementar aplicación de realidad aumentada para equipo mecatrónico.

La tarea final corresponde a una integración del equipo mecatrónico junto a la aplicación de realidad aumentada mediante una estrategia del internet industrial de las cosas (IIOT).

Es por tal, la necesidad de contar con un procedimiento basado en herramientas de software libre que permitan conseguir la experiencia y cumplir con esta tarea del contenido curricular de la formación tecnológica.

1.3. Objetivos

1.3.1. Objetivo General

Proponer una aplicación móvil de realidad aumentada de un equipo mecatrónico para atender el desarrollo formativo sobre Industria 4.0 en un Instituto Tecnológico Superior.

1.3.2. Objetivos Específicos

a. Diseñar la arquitectura de la aplicación de Realidad Aumentada que permita comunicar con el equipo mecatrónico.
b. Publicar datos de equipo mecatrónico a través de protocolo MQTT con plataforma NODE-RED.
c. Elaborar aplicación de Realidad Aumentada con softwares Unity y Vuforia Engine.
d. Comunicar aplicación de Realidad Aumentada con equipo mecatrónico a través del protocolo MQTT.

1.4. Justificación e importancia

Este proyecto pretende desarrollar una aplicación móvil de realidad aumentada que permita conectar con un equipo mecatrónico y adicionar información virtualmente a un entorno real. La razón del desarrollo de la aplicación se debe a la necesidad de afrontar

los cambios en la formación educativa en tópicos de Industria 4.0 a los estudiantes del Instituto Tecnológico Superior en estudio.

El desarrollo de la aplicación importa porque permite a los docentes de la especialidad de Mecatrónica Industrial contar con un procedimiento basado en la aplicación de software libre, con ello se consigue replicar a todos los alumnos del último semestre el desarrollo de la tarea, y así lograr incorporar, de una manera práctica, los conceptos de Industria 4.0 como lo son Realidad Aumentada e Internet industrial de las cosas.

CAPÍTULO II. Marco Teórico y Marco conceptual

2.1. Bases Teóricas

2.1.1. Fundamentos de Industria 4.0

La Industria 4.0 se refiere a la cuarta revolución industrial, caracterizada por la integración de tecnologías avanzadas como el Internet de las Cosas (IoT), la inteligencia artificial (IA), la automatización, el Big Data y la computación en la nube en los procesos productivos. Esta transformación digital permite la creación de fábricas inteligentes, en las que los sistemas ciberfísicos monitorizan los procesos de producción, toman decisiones descentralizadas y optimizan el uso de recursos en tiempo real (Lasi et al., 2014).

Figura 3: Industria 4.0

Fuente: https://www.edsrobotics.com/blog/que-es-la-industria-4-0/

El objetivo de la Industria 4.0 es aumentar la eficiencia, flexibilidad y personalización en la producción, facilitando la interconexión entre máquinas, productos y sistemas. Las empresas que adoptan esta tecnología pueden mejorar la productividad, reducir tiempos de inactividad y crear nuevos modelos de negocio basados en el análisis de datos en tiempo real y la conectividad.

La Industria 4.0 también promueve la convergencia entre los mundos físico y digital, donde las fábricas son capaces de adaptarse de manera dinámica a las necesidades cambiantes del mercado y personalizar productos a gran escala.

Figura 4: Interconexión entre mundo físico y digital

Fuente: https://www.berxa.com.ar/blog/que-es-la-industria-4-0

A continuación, se describen los principales pilares fundamentales que conforman la base de esta revolución industrial:

a. Internet Industrial de las Cosas (IoT)

El Internet Industrial de las Cosas (IIoT) implica la interconexión de dispositivos, máquinas y sistemas a través de redes industriales para compartir datos en tiempo real. Este pilar permite la creación de fábricas inteligentes donde los dispositivos pueden interactuar entre sí y con los sistemas de gestión para optimizar los procesos productivos. Además, el IIoT ofrece visibilidad total de las operaciones, desde la cadena de suministro hasta el control de calidad, lo que permite una gestión más eficiente y una respuesta rápida a los cambios en la demanda del mercado (Boyes et al., 2018).

Figura 5: Internet industrial de las cosas

Fuente: https://engapplic.com/iiot-industrial-internet-of-things-como-me-beneficio-del-iiot-industria-4-0-cloudcomo-implemento-iot/

b. Robots Autónomos

Los robots autónomos juegan un papel central en la automatización avanzada dentro de la Industria 4.0. Estos robots no solo ejecutan tareas repetitivas, sino que también tienen la capacidad de colaborar con humanos y adaptarse dinámicamente a entornos cambiantes. Gracias a la inteligencia artificial, los robots autónomos pueden tomar decisiones en tiempo real, lo que aumenta la flexibilidad y reduce los tiempos de inactividad en las líneas de producción. Además, los robots colaborativos, o cobots, permiten a los humanos y robots trabajar en conjunto, mejorando la seguridad y la eficiencia en las fábricas (Chui et al., 2016).

Figura 6: Robots autónomos

Fuente: https://www.freepik.es/imagen-ia-premium/robots-autonomos-revolucionan-instalaciones-industria-maquinaria-futurista_187295076.htm

c. Computación en la Nube

La computación en la nube proporciona una infraestructura escalable y flexible para el almacenamiento y procesamiento de datos, lo que es crucial en la Industria 4.0. Las empresas pueden acceder a grandes capacidades de procesamiento de forma remota, permitiendo que los sistemas ciberfísicos y las fábricas inteligentes operen de manera eficiente y en tiempo real. Esta tecnología permite a las organizaciones gestionar sus operaciones de manera global, integrar aplicaciones de análisis avanzado y colaborar entre diferentes ubicaciones sin necesidad de infraestructura física adicional (Xu et al., 2018).

Figura 7: Computación en la nube

Fuente: https://whitestack.com/es/blog/cloud-computing/

d. Realidad Aumentada

La realidad aumentada (RA) ofrece nuevas formas de interactuar con el entorno físico al superponer información digital sobre el mundo real. En el contexto de la Industria 4.0, la RA se utiliza para mejorar la capacitación de los trabajadores, guiar a los técnicos durante el mantenimiento de maquinaria y visualizar datos de producción en tiempo real. Esto permite que los operarios tengan acceso inmediato a información relevante, lo que mejora la eficiencia, la precisión y la seguridad en los procesos industriales (Wang et al., 2016).

Figura 8: Aplicaciones de realidad aumentada

Fuente: https://www.edsrobotics.com/blog/realidad-aumentada-que-es/

e. Ciberseguridad

A medida que las fábricas se vuelven más interconectadas, la ciberseguridad se convierte en un aspecto crucial de la Industria 4.0. La protección de los sistemas industriales frente a ataques cibernéticos y la preservación de la integridad de los datos es esencial para garantizar la continuidad operativa. Los sistemas avanzados de ciberseguridad integran soluciones para la protección de datos y la autenticación de dispositivos, lo que minimiza el riesgo de intrusiones y asegura la privacidad y la seguridad de las operaciones en las fábricas inteligentes (Boyes et al., 2018).

Figura 9: Ciberseguridad

Fuente: https://academia-ciberseguridad.com/

f. Sistemas de Integración Horizontal y Vertical

La integración horizontal y vertical es clave para lograr la plena conectividad en la Industria 4.0. La integración horizontal conecta diferentes sistemas y procesos a lo largo de la cadena de valor, como proveedores, distribuidores y socios logísticos. Por otro lado, la integración vertical conecta todos los niveles dentro de una organización, desde la planta de producción hasta los sistemas de gestión empresarial. Esta interconexión permite a las empresas optimizar su capacidad de respuesta, mejorar la eficiencia y coordinar operaciones de manera integral (Kagermann et al., 2013).

g. Fabricación Aditiva

La fabricación aditiva, comúnmente conocida como impresión 3D, es un pilar que transforma la producción industrial al permitir la creación de productos personalizados de

forma rápida y eficiente. Esta tecnología reduce los costos de fabricación al minimizar el desperdicio de materiales y permitir la producción de piezas complejas directamente a partir de modelos digitales. Además, la fabricación aditiva facilita la innovación en el diseño de productos, ya que permite realizar prototipos y producir lotes pequeños de manera más económica (Berman, 2012).

Figura 10: Fabricación aditiva

Fuente: https://academia-ciberseguridad.com/

h. Simulación

La simulación permite modelar y analizar los procesos productivos en tiempo real utilizando gemelos digitales. Los gemelos digitales son representaciones virtuales de sistemas físicos que permiten realizar pruebas y optimizaciones antes de implementar cambios en la producción. Esto reduce el tiempo de inactividad, mejora la eficiencia y disminuye los costos al predecir cómo funcionarán los sistemas antes de que se ejecuten en la realidad (Tao et al., 2019).

i. Big Data y Análisis

El análisis de Big Data es un componente esencial de la Industria 4.0, ya que permite procesar grandes volúmenes de datos generados por sensores y dispositivos en tiempo real. El análisis avanzado de estos datos proporciona información clave para la toma de decisiones estratégicas, la optimización de procesos y la predicción de fallos en equipos. Esto ayuda a mejorar la eficiencia operativa, reducir tiempos de inactividad y aumentar la calidad de los productos (Lee et al., 2014). Los sistemas basados en Big Data

también facilitan la implementación de mantenimiento predictivo, lo que minimiza las interrupciones y prolonga la vida útil de las máquinas.

Figura 11: Big Data

Fuente: https://iarchiva.com/que-es-big-data-y-como-transforma-el-analisis-de-documentos/

2.1.2. Fundamentos de realidad aumentada

El avance tecnológico permite involucrar tecnologías a entornos que promuevan la inserción de la realidad aumentada en diversos sectores económicos. Muñoz Saavedra et al. (2020) señalan que la realidad aumentada y el avance tecnológico han permitido que estas tecnologías se utilicen en una amplia gama de campos, como la educación y la industria, entre otros. En la educación y la industria, estas tecnologías se destacan como herramientas poderosas para el aprendizaje y la formación, logrando una enseñanza más eficiente, interactiva y participativa. Además, las mejoras tecnológicas en dispositivos y hardware de procesamiento han permitido que la realidad aumentada y virtual se utilicen en dispositivos más accesibles, como los smartphones de gama media (Kimura et al. 2019). En efecto, incorporar las tecnologías dentro de un entorno educativo para la industria se ve potenciada para la enseñanza con aplicaciones de realidad aumentada dado que se cuentan con dispositivos mas accesibles para su implementación.

Para Berryman (2012), la realidad aumentada es una tecnología que incorpora contenido digital sobre objetos o lugares del entorno real para elevar la experiencia del usuario. Por ello, para un entorno educativo para la industria, conseguir añadir información sobre capturas o imágenes de un equipamiento real, viene a ser un desarrollo de realidad aumentada.

Según Azuma (1997), la realidad aumentada debe cumplir tres criterios fundamentales: combinar elementos reales y virtuales, ser interactiva en tiempo real, estar registrada espacialmente en tres dimensiones. Es decir, en el contexto de la Industria 4.0, los criterios indicados ayudan a mejorar operaciones y acciones de soporte para el mantenimiento de equipos de las fábricas, de donde nace el término Realidad aumentada industrial.

Realidad aumentada industrial es la aplicación de la tecnología de realidad aumentada (RA) en entornos industriales con el objetivo de mejorar y optimizar procesos de producción, mantenimiento, capacitación y gestión operativa. Esta tecnología permite superponer información digital, como instrucciones de ensamblaje, datos de sensores, modelos 3D y diagramas técnicos, directamente sobre el entorno físico de trabajo. De este modo, los operarios pueden interactuar de manera más eficiente con sus tareas, reduciendo errores, aumentando la productividad y mejorando la seguridad en el lugar de trabajo.

Por ejemplo, en el ámbito del mantenimiento, la RA industrial puede proporcionar a los técnicos guías visuales paso a paso para la reparación de maquinaria compleja, lo que disminuye el tiempo de inactividad y asegura una mayor precisión en las intervenciones. Además, en la capacitación de empleados, la realidad aumentada facilita la creación de simulaciones interactivas que aceleran el aprendizaje y mejoran la retención de conocimientos prácticos.

Figura 12: Realidad aumentada aplicada en la industria

Fuente: https://innoarea.com/noticias/industria-4-0-a-traves-de-realidad-virtual-y-realidad-aumentada/

La implementación de la RA industrial también contribuye a la toma de decisiones basada en datos en tiempo real, ya que integra información procedente de diversos sensores y sistemas de monitoreo, permitiendo una respuesta rápida ante cualquier eventualidad en el proceso productivo (Schmalstieg & Höllerer, 2016).

Vuforia Engine es una plataforma de desarrollo de realidad aumentada que permite a los creadores integrar contenido digital tridimensional en entornos físicos utilizando dispositivos móviles y gafas inteligentes. Vuforia utiliza la visión por computadora para detectar y rastrear objetos, imágenes y superficies del mundo real, sobre los cuales se superponen elementos digitales interactivos.

Figura 13: Vuforia Engine – uso de image target

Fuente: https://developer.vuforia.com/library/objects/image-targets

Una de las principales características de Vuforia es su capacidad para trabajar en una amplia gama de dispositivos y plataformas, permitiendo a los desarrolladores crear experiencias de RA multiplataforma. La plataforma también ofrece soporte para el reconocimiento de objetos 3D, planos y de imágenes, y se ha utilizado ampliamente en áreas como la educación, el entretenimiento y la publicidad (Pereira et al., 2020).

Unity es una plataforma de desarrollo de software que permite la creación de aplicaciones interactivas en 2D y 3D, siendo ampliamente utilizada en la creación de experiencias de realidad aumentada (RA) y realidad virtual (RV). Unity proporciona un entorno de desarrollo integrado (IDE) que facilita la creación, prueba e implementación de proyectos de RA a través de su motor gráfico y herramientas especializadas, como el AR Foundation, que permite a los desarrolladores integrar funcionalidades de RA en diferentes dispositivos.

Unity es compatible con plataformas de RA como Vuforia y ARCore, permitiendo la creación de aplicaciones multiplataforma con capacidades avanzadas de seguimiento, reconocimiento de objetos y superposición de contenido digital en el mundo real. Su flexibilidad y robustez lo han convertido en una herramienta esencial en el desarrollo de RA en áreas como los videojuegos, la educación y las aplicaciones industriales (Pallavicini et al., 2019).

2.1.3. Fundamentos de internet industrial de las cosas

El Internet de las Cosas (IoT) se refiere a la interconexión de objetos físicos cotidianos a través de Internet, lo que les permite enviar, recibir y procesar datos. Estos dispositivos, equipados con sensores, software y otras tecnologías, recopilan información del entorno y se comunican entre sí o con otros sistemas a través de redes, con el fin de automatizar procesos, mejorar la toma de decisiones y optimizar el uso de recursos. El IoT tiene aplicaciones en diversos campos, como la domótica, la salud, la industria y las ciudades inteligentes, donde se utiliza para mejorar la eficiencia y la calidad de vida (Atzori et al., 2010).

Una característica clave del IoT es su capacidad para proporcionar un entorno interconectado en el que los dispositivos funcionan de manera autónoma, adaptándose a las necesidades de los usuarios y a las condiciones cambiantes del entorno. Esto ha generado grandes expectativas sobre su impacto en áreas como la automatización industrial, la gestión de energía y la atención médica.

El Internet Industrial de las Cosas (IIoT) es una subcategoría del Internet de las Cosas (IoT) que se enfoca en la interconexión de dispositivos y sistemas en entornos industriales, como fábricas, plantas de energía y cadenas de suministro. El IIoT implica el uso de sensores, software y otras tecnologías para recopilar, procesar y analizar grandes cantidades de datos en tiempo real, con el objetivo de mejorar la eficiencia operativa, reducir costos y optimizar los procesos industriales.

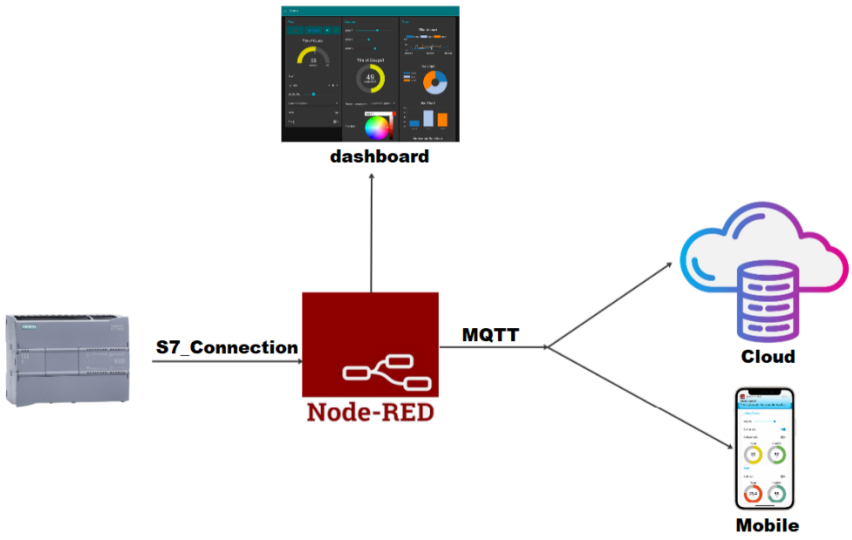
A través del IIoT, las empresas pueden monitorizar equipos, predecir fallos antes de que ocurran, automatizar procesos y mejorar la toma de decisiones mediante el análisis de datos. Además, su capacidad para interconectar dispositivos y sistemas permite una integración más eficiente entre diversas etapas de la cadena de valor industrial, facilitando la colaboración entre máquinas y humanos (Boyes et al., 2018).

El IIoT es clave en la transformación hacia la Industria 4.0, donde la digitalización de las operaciones industriales es fundamental para lograr una producción más inteligente y conectada.

Node-RED es una herramienta de programación visual de código abierto desarrollada por IBM, que facilita la creación de flujos de trabajo para la integración de dispositivos y sistemas, especialmente en el contexto del Internet Industrial de las Cosas (IIoT). Node-RED utiliza un enfoque basado en nodos para conectar dispositivos, bases de datos y servicios en la nube, lo que simplifica la gestión de datos en tiempo real y la automatización de procesos en entornos industriales. Esta plataforma es especialmente útil para IIoT debido a su flexibilidad y capacidad para integrar diferentes protocolos industriales como MQTT, OPC-UA y Modbus, que son esenciales en los sistemas ciberfísicos (Mineraud et al., 2016).

Además, Node-RED permite a los usuarios no necesariamente expertos en programación desarrollar aplicaciones IIoT mediante una interfaz intuitiva, lo que acelera la creación de soluciones de monitoreo, control y análisis en tiempo real dentro de plantas industriales. Su enfoque modular y extensible también facilita la personalización y escalabilidad en proyectos de IIoT, adaptándose a las necesidades de distintas industrias.

Figura 14: NODE-RED para Internet industrial de las cosas



Fuente: https://www.solisplc.com/tutorials/iiot-software

2.1.4. Fundamentos de comunicación de equipos mecatrónicos

OPC-UA (Open Platform Communications - Unified Architecture) es un estándar de comunicación para la interoperabilidad de dispositivos y sistemas industriales en la automatización, que permite el intercambio de datos de manera segura y confiable entre dispositivos, aplicaciones y sistemas de control. A diferencia de sus predecesores, OPC-UA no solo se enfoca en la comunicación a nivel de campo o dispositivo, sino que también abarca el intercambio de información a nivel empresarial, lo que lo convierte en una herramienta clave en la implementación de la Industria 4.0 y el Internet Industrial de las Cosas (IIoT).

OPC-UA es independiente de la plataforma, lo que significa que puede funcionar en diferentes sistemas operativos y arquitecturas. Además, es compatible con una amplia

gama de tecnologías de transporte y ofrece robustas características de seguridad, como la encriptación y la autenticación, para garantizar que los datos intercambiados entre dispositivos y sistemas estén protegidos. Este protocolo es ampliamente utilizado en entornos industriales por su capacidad para integrar máquinas y sistemas heterogéneos (Mahmoud et al., 2020).

MQTT (Message Queuing Telemetry Transport) es un protocolo ligero de mensajería diseñado para la transmisión eficiente de datos en redes con limitaciones de ancho de banda o alta latencia, características comunes en aplicaciones del Internet de las Cosas (IoT) y el Internet Industrial de las Cosas (IIoT). MQTT utiliza un modelo de comunicación publicador-suscriptor, en el que los dispositivos (publicadores) envían mensajes a un "broker" central, que luego distribuye esos mensajes a los dispositivos suscritos. Este enfoque reduce el consumo de recursos, haciendo que MQTT sea ideal para dispositivos con capacidades de procesamiento y energía limitadas, como sensores y actuadores en entornos industriales (Hunkeler et al., 2008).

MQTT también ofrece características importantes como la calidad de servicio (QoS), lo que permite a los desarrolladores controlar el nivel de fiabilidad de la entrega de mensajes, garantizando la transmisión de datos críticos en entornos industriales. Además, su simplicidad y flexibilidad lo han convertido en uno de los protocolos más utilizados en aplicaciones IoT, donde la escalabilidad y el bajo uso de recursos son fundamentales.

Figura 15: OPC UA y MQTT como protocolos de comunicación

Fuente: https://hilscher.com/technology/industrial-iot/iiot-protocols-opc-ua-and-mqtt

2.2. Marco Conceptual

Aquí van los conceptos claves del trabajo

a. **Desarrollo Formativo:** Término que hace referencia al proceso de aprendizaje de un alumno dentro de la Institución Tecnológica. Para el presente trabajo, corresponde a la secuencia de temas y tareas dentro de las especialidades que se brindan.

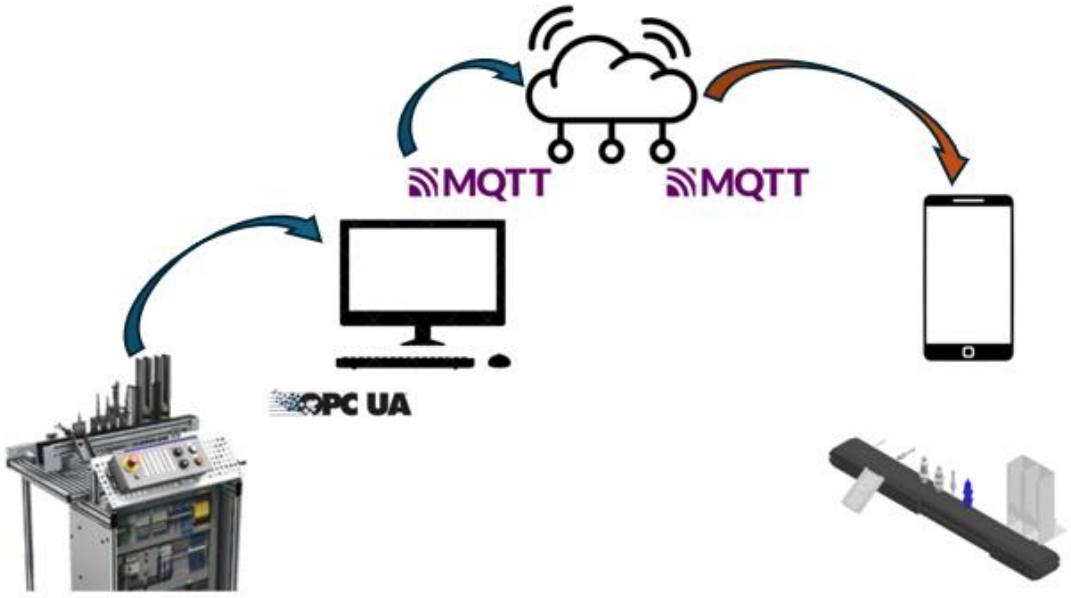
b. **Módulo Mecatrónico:** Máquina automática para la formación en Mecatrónica Industrial, contiene como componentes Controladores Lógicos Programables (PLC), paneles terminal operador (HMI), conectividad a una red local (LAN-Ethernet), y diversos dispositivos de campo (sensores, electroválvulas, cilindros neumáticos, cintas transportadoras y elementos mecánicos).

CAPÍTULO III. Desarrollo del Trabajo

3.1. Arquitectura del diseño del proyecto

Se detalla cómo se elabora el diseño del proyecto para que los estudiantes puedan desarrollar el paso a paso con los diversos softwares.

Figura 16: Esquema de la arquitectura del diseño del proyecto



De la Figura 16 se verifica que el equipo mecatrónico físico se conecta mediante protocolo OPC-UA con un servidor de datos representado por una computadora, luego la computadora conecta a la nube mediante el protocolo MQTT con un bróker, por otro lado, se tiene la aplicación móvil de realidad aumentada que conecta con el bróker MQTT para poder mostrar los datos publicados por el equipo mecatrónico, completando el flujo de acceso a los datos.

3.2. Comunicación OPC-UA de equipo mecatrónico

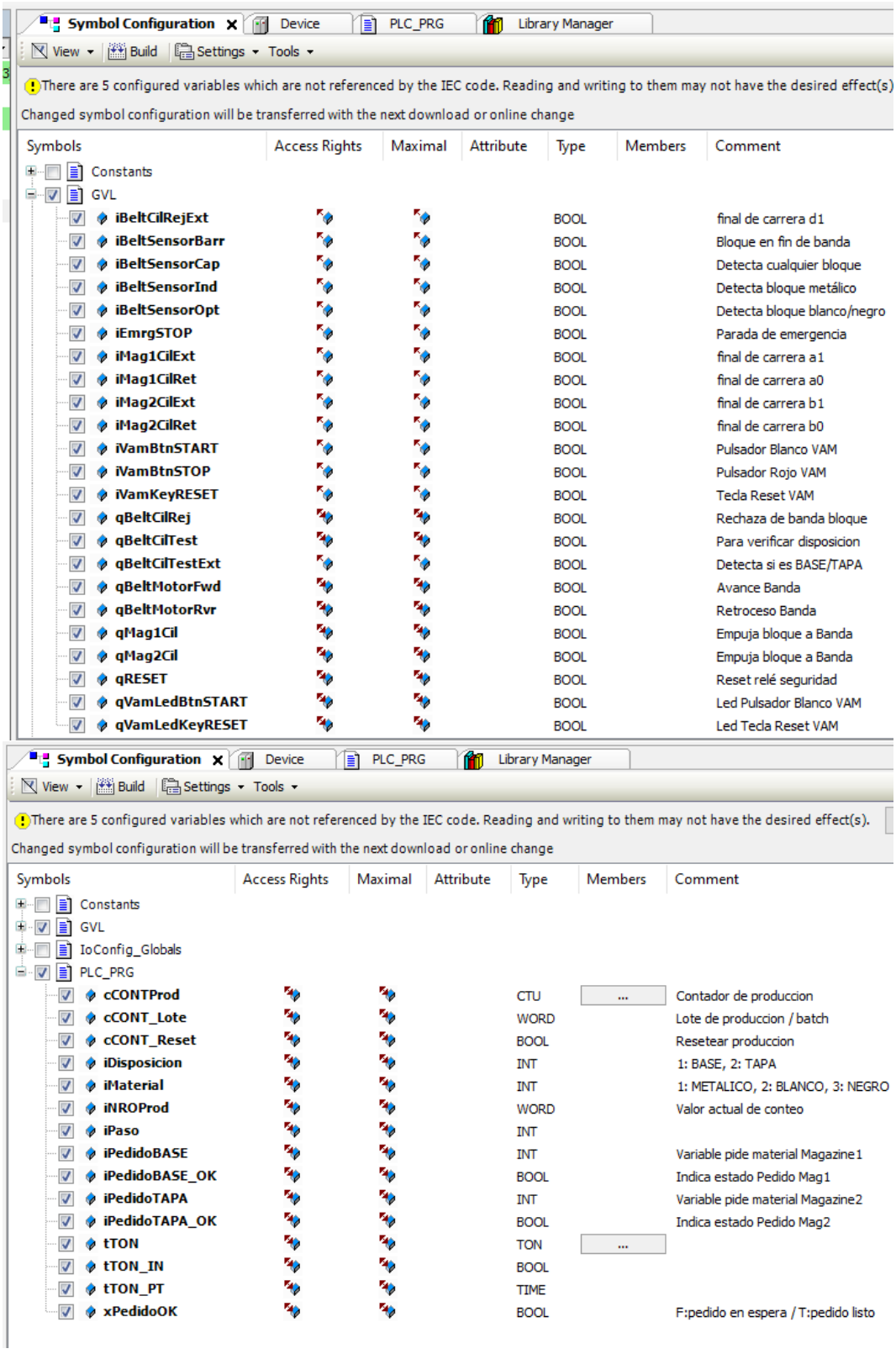
Para garantizar que los datos del equipo mecatrónico sean accesibles desde la aplicación de realidad aumentada, se requiere en primera instancia compartir datos de este a través del protocolo OPC-UA. En esta ocasión el equipo mecatrónico cuenta con un PLC compatible con este protocolo de comunicación, por lo cual el acceso a sus datos es directo. Para realizar esta acción se siguen los siguientes pasos:

a. ***Compartir variables del programa a través de símbolos de sistema hacia un PC-servidor***: el PLC trabaja bajo una estructura CODESYS, en la Figura 17, se verifica las variables compartidas como las indicadas en la Tabla 2.

Tabla 2: Lista de variables compartidas por OPC-UA del equipo mecatrónico

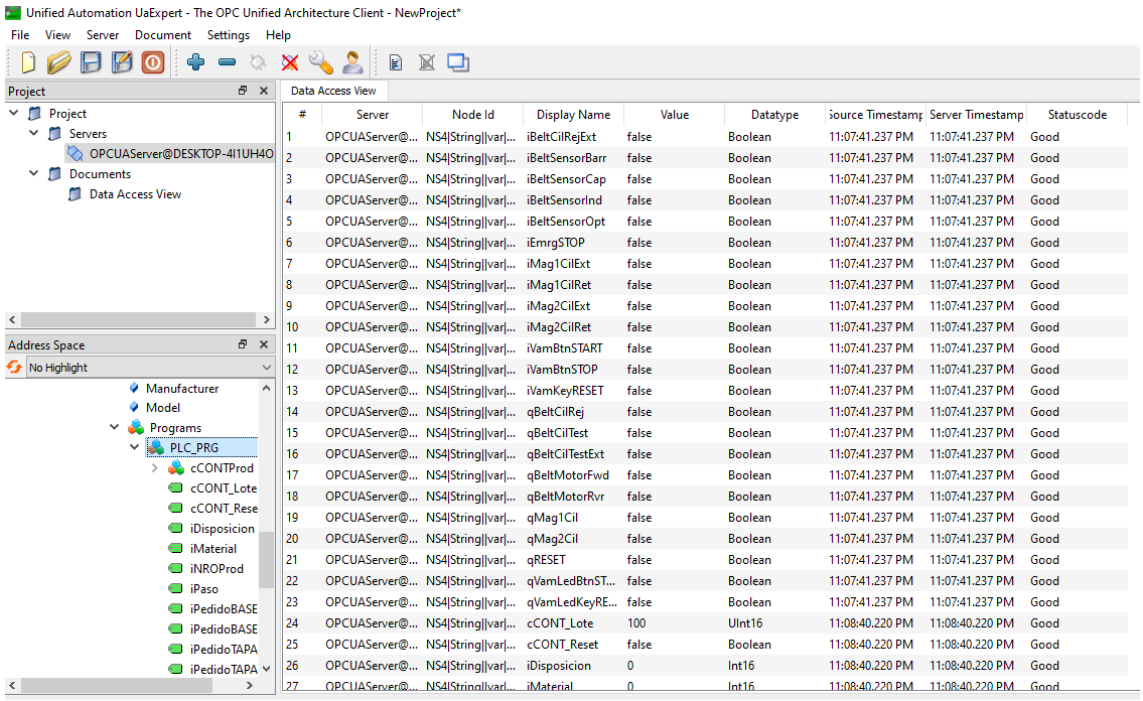
Ítem	Descripción	Nombre Variables
1	Sensor de descarte material	GVL.iBeltCilRejExt
2	Sensor de barrera de fin de banda	GVL.iBeltSensorBarr
3	Sensor capacitivo	GVL.iBeltSensorCap
4	Sensor inductivo	GVL.iBeltSensorInd
5	Sensor óptico	GVL.iBeltSensorOpt
6	Sensor testeo	GVL.qBeltCilTestExt
7	Sensor despacho magazine 1	GVL.iMag1CilExt
8	Sensor despacho magazine 2	GVL.iMag2CilExt
9	Parada de emergencia	GVL.iEmrgSTOP
10	Lote de producción	PLC_PRG.cCONT_Lote
11	Producción actual	PLC_PRG.iNROProd
12	Disposición de producto	PLC_PRG.iDisposicion
13	Material de producto	PLC_PRG.iMaterial
14	Confirmación de base solicitada	PLC_PRG.iPedidoBASE_OK
15	Confirmación de tapa solicitada	PLC_PRG.iPedidoTAPA_OK

Figura 17: Configuración de símbolos de sistema para comunicación OPC-UA del equipo mecatrónico – parte 1



b. ***Identificar los "node-Id" de las variables compartidas mediante comunicación OPC-UA***: se utiliza la aplicación UA expert para verificar que se encuentra disponible los datos del equipo mecatrónico, en la Figura 18 se puede visualizar la verificación de las señales obtenidas mediante Opc-Ua del equipo mecatrónico.

Figura 18: Configuración de símbolos de sistema para comunicación OPC-UA del equipo mecatrónico – parte 2



Ahora que se verificó que se tiene acceso a las variables del equipo mecatrónico a través de OPC-UA, se debe listar los identificadores en la comunicación para cada una de las variables, por ello se identifica en cada variable a necesitar en el panel derecho de la aplicación UA expert el "Node-Id" respectivo (ver Figura 19). En la Tabla 3 se listan las variables del equipo mecatrónico seguido de su Node-Id respectiva.

Figura 19: Ventana para identificar valor del NodeId de la variable respectiva

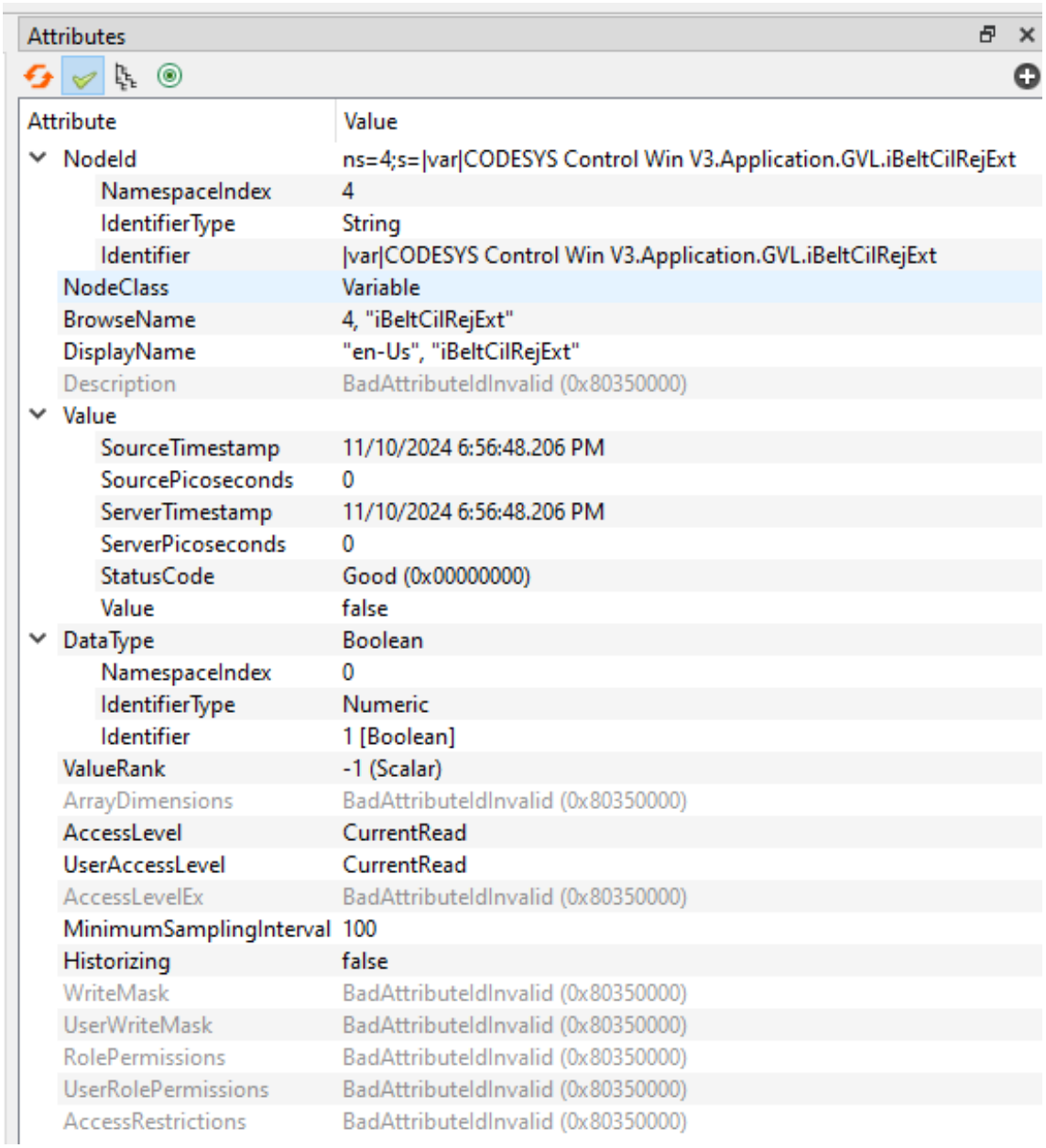
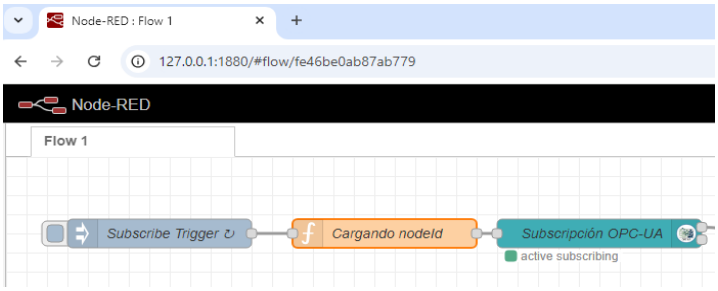


Tabla 3: Lista de Node-Id correspondientes a las variables compartidas por OPC-UA del equipo mecatrónico

Ítem	Descripción	Node-Id
1	Sensor de descarte material	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iBeltCilRejExt
2	Sensor de barrera de fin de banda	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iBeltSensorBarr
3	Sensor capacitivo	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iBeltSensorCap
4	Sensor inductivo	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iBeltSensorInd
5	Sensor óptico	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iBeltSensorOpt
6	Sensor testeo	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.qBeltCilTestExt
7	Sensor despacho magazine 1	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iMag1CilExt
8	Sensor despacho magazine 2	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iMag2CilExt
9	Parada de emergencia	ns=4;s=|var|CODESYS Control Win V3.Application.GVL.iEmrgSTOP
10	Lote de producción	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.cCONT_Lote
11	Producción actual	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.iNROProd
12	Disposición de producto	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.iDisposicion
13	Material de producto	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.iMaterial
14	Confirmación de base solicitada	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.iPedidoBASE_OK
15	Confirmación de tapa solicitada	ns=4;s=|var|CODESYS Control Win V3.Application.PLC_PRG.iPedidoTAPA_OK

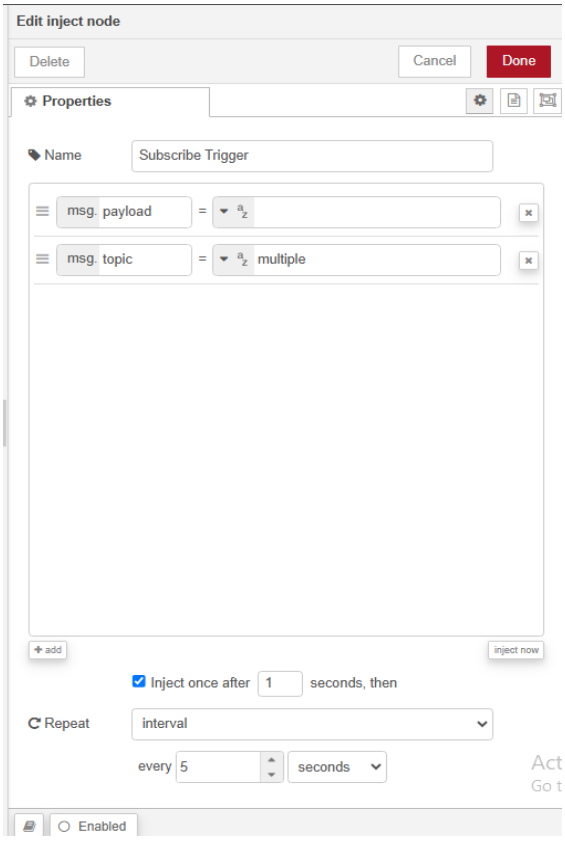
c. ***Obtener los datos del equipo mecatrónico a través de la plataforma NODE-RED***: Se utilizan los nodos OPC-UA client a través del método subscripción, se accede a los datos. Como se verifica en la Figura 20, se utilizan 3 nodos: Inject (Subscribe Trigger), function (Cargando nodeId), OPC-UA client (Subscripción OPC-UA).

Figura 20: Flujo en NODE-RED para acceso a datos de equipo mecatrónico mediante OPC-UA



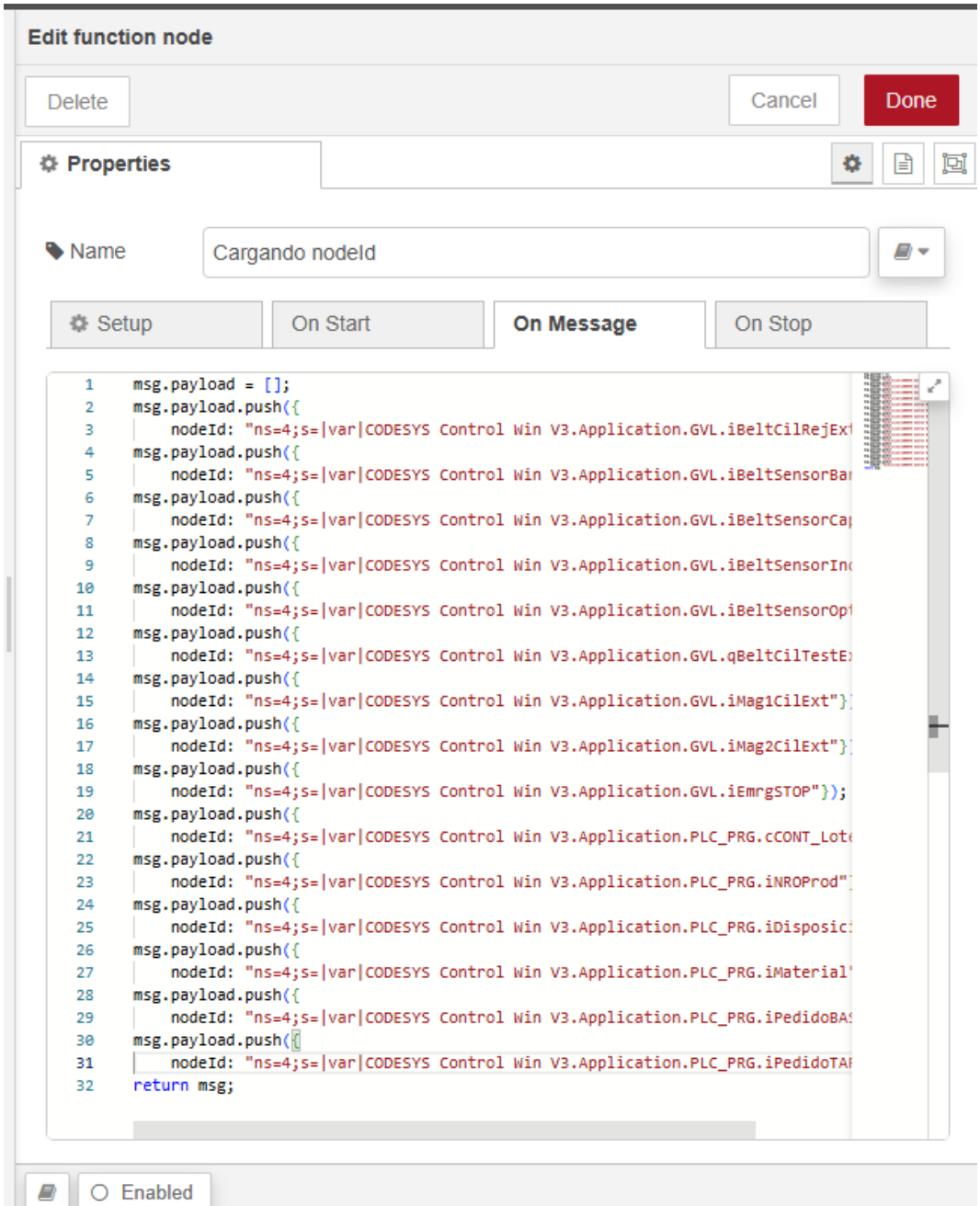
Se verifica en la Figura 21, la configuración del nodo Inject (Subscribe Trigger) que permite mandar una señal de pulso cada 5 segundos (Repeat: Interval, every 5 seconds) y comienza luego de 1 segundo de iniciada la aplicación (inject one after 1 seconds).

Figura 21: Configuración nodos Inject, Function y OpcUa-Client para el acceso a los datos OPC-UA del equipo mecatrónico



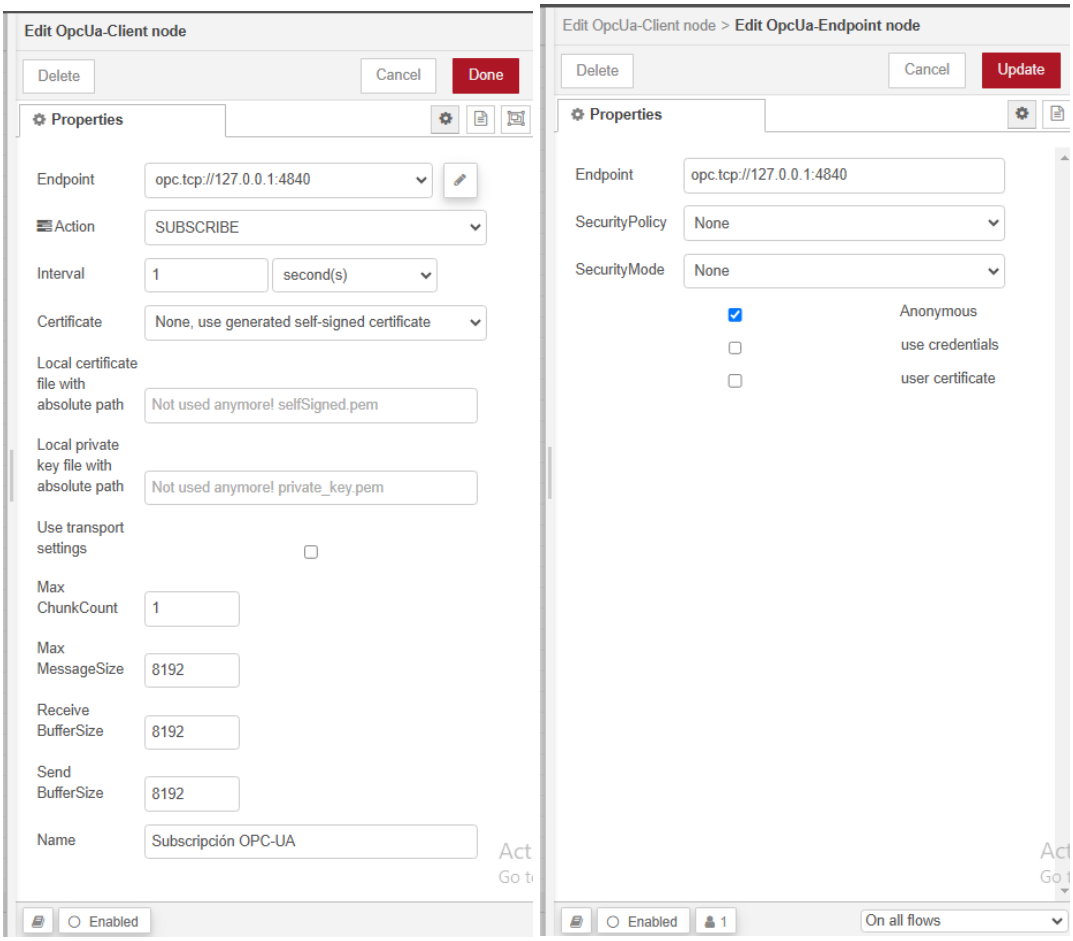
Se verifica en la Figura 22, la configuración del nodo Function (Cargando nodeId) el cual contiene la secuencia de agregar (método ".push") los nodos identificados en la Tabla 3, de acuerdo al nodeId respectivo, el mensaje retornado para el siguiente nodo contiene ahora la lista de nodeId's que se requiere para consultar por sus valores en el nodo siguiente.

Figura 22: Configuración nodo Function para el acceso a los datos OPC-UA del equipo mecatrónico



Se verifica en la Figura 23, la configuración del nodo OpcUa-Client (Subscripción OPC-UA) en donde se configura el acceso al servidor OPCUa (para el ejemplo se utiliza el mismo dispositivo 127.0.0.1:4840) y se selecciona la acción “SUBSCRIBE” con la cual se solicita la lista de node-Id’s indicados en la Figura 22, con ello se obtendrá de respuesta un listado de mensajes (uno por cada node-Id) donde el “msg.payload” corresponde al valor respectivo de la variable relacionada según la Tabla 3.

Figura 23: Configuración nodo OpcUa-Client para el acceso a los datos OPC-UA del equipo mecatrónico



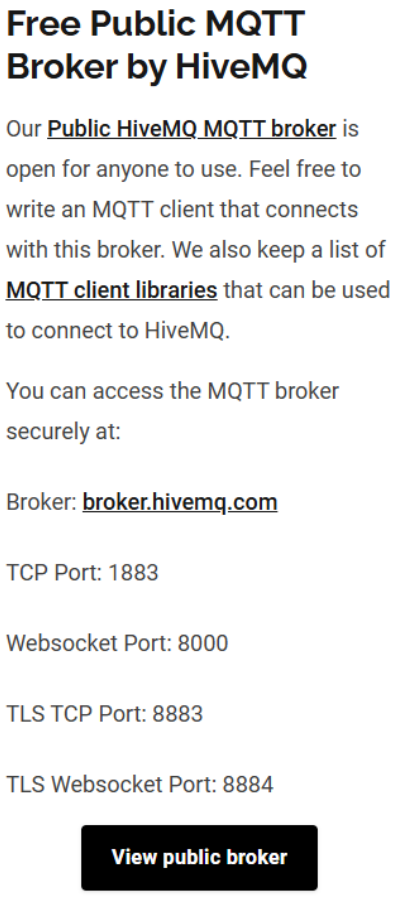
Con estos pasos se consigue comunicar el equipo mecatrónico al PC-servidor a través de comunicación OPC-UA.

3.3. Comunicación del equipo industrial con Internet a través de MQTT y NODE-RED

Hasta el momento, se tiene acceso a los valores de las variables correspondientes del equipo mecatrónico en un PC-servidor, ahora lo que se debe realizar es comunicar este con internet para poder acceder a los datos desde la aplicación de realidad aumentada, a continuación, se listan los pasos para garantizar que la información se esté publicando a internet a través del protocolo MQTT.

a. ***Definir Bróker MQTT***: para este paso, se definió como Bróker MQTT al Bróker HiveMQ, el cual cuenta con un servidor de prueba libre que permitirá que cada usuario pueda realizar la práctica sin contratiempo mayor, las configuraciones necesarias para acceder al servidor libre del Bróker HiveMQ se pueden verificar en la Figura 24. Para la conexión con el mismo se utilizará el puerto TCP (1883).

Figura 24: Bróker de acceso público por HiveMQ



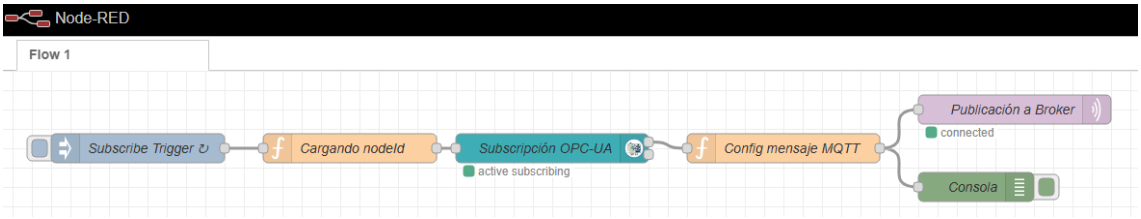
b. ***Establecer tópicos en el Bróker***: Ahora se debe definir el nombre del tópico donde se almacenará los valores de las variables que llegarán al Bróker desde el PC-servidor, para ello se designa el siguiente tópico contenedor “Modulo4.0” seguido de un tópico propio de cada variable según las variables de la Tabla 2, que se muestran ahora en la Tabla 4, cabe indicar que si se envía alguna variable no listada en la tabla se utilizará un tópico genérico definido como “Hola”.

Tabla 4: Lista de tópicos asociados a las variables del equipo mecatrónico – Parte 1

Ítem	Variable	Tópico asociado
1	Sensor de descarte material	Modulo4.0/iBeltCilRejExt
2	Sensor de barrera de fin de banda	Modulo4.0/iBeltSensorBarr
3	Sensor capacitivo	Modulo4.0/iBeltSensorCap
4	Sensor inductivo	Modulo4.0/iBeltSensorInd
5	Sensor óptico	Modulo4.0/iBeltSensorOpt
6	Sensor testeo	Modulo4.0/qBeltCilTestExt
7	Sensor despacho magazine 1	Modulo4.0/iMag1CilExt
8	Sensor despacho magazine 2	Modulo4.0/iMag2CilExt
9	Parada de emergencia	Modulo4.0/iEmrgSTOP
10	Lote de producción	Modulo4.0/cCONT_Lote
11	Producción actual	Modulo4.0/iNROProd
12	Disposición de producto	Modulo4.0/iDisposicion
13	Material de producto	Modulo4.0/iMaterial
14	Confirmación de base solicitada	Modulo4.0/iPedidoBASE_OK
15	Confirmación de tapa solicitada	Modulo4.0/iPedidoTAPA_OK
16	Otro caso	Modulo4.0/Hola

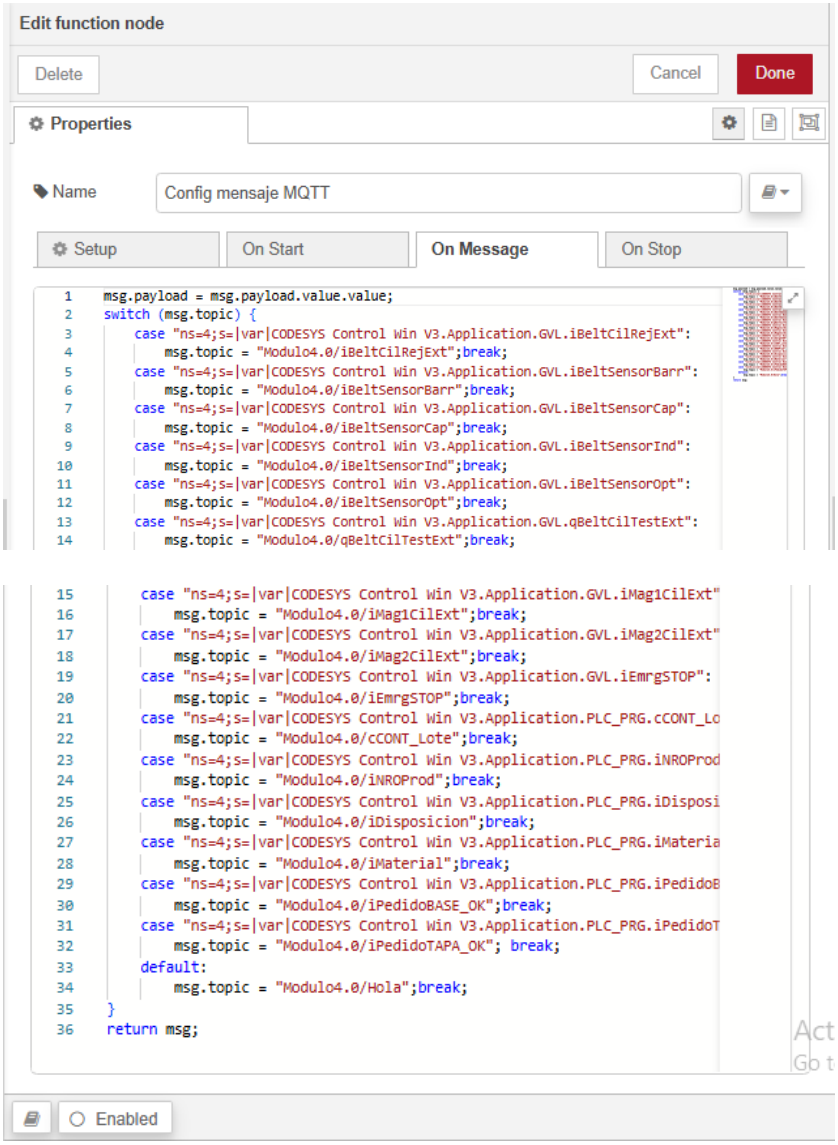
c. ***Incorporar nodos de publicación MQTT en la plataforma NODE-RED***: Para lograr publicar los datos obtenidos por OpcUa del equipo mecatrónico hacia el Bróker MQTT, se incorpora los nodos mostrados en la Figura 25 que son: Function (Config mensaje MQTT), Mqtt Out (Publicación a Bróker), Debug (Consola) Figura 24, con ellos se consigue tener los estados de las variables respectivas en el Bróker MQTT para ser accesibles posteriormente desde la aplicación de realidad aumentada.

Figura 25: Flujo en NODE-RED incorporados para publicación al Bróker MQTT



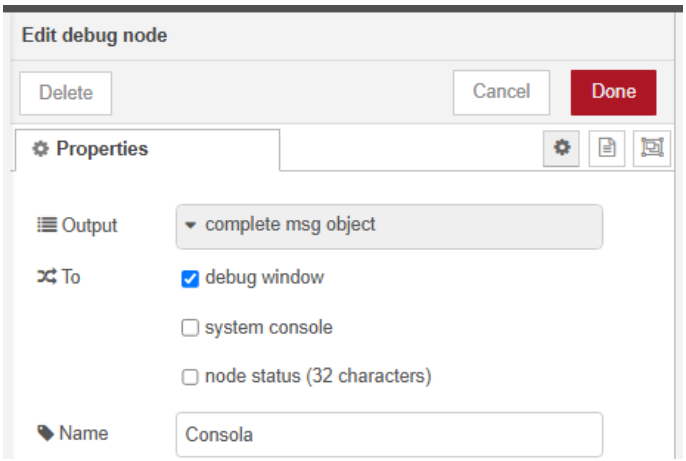
Se verifica en la Figura 26, la configuración del nodo Function (Config mensaje MQTT), en el cual, apoyándose de la instrucción ramificada “switch” en función al Node-Id correspondiente a cada variable obtenida desde la comunicación OPC-UA, se asigna el tópico respectivo según lo indicado en la Tabla 4.

Figura 26: Nodo Function para la configuración del tópico respectivo en los mensajes al bróker MQTT



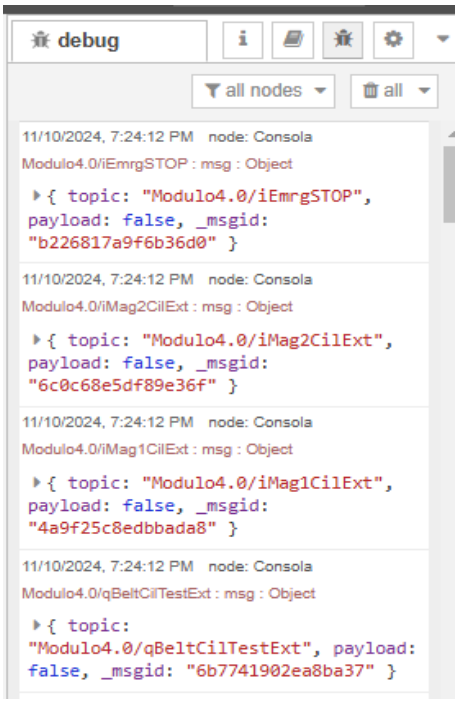
Se verifica en la Figura 27, la configuración del nodo Debug (Consola) que nos permite mostrar los resultados publicados al Bróker MQTT y verificar que salga el mensaje correcto.

Figura 27: Configuración de nodo Debug para visualizar resultado del mensaje MQTT hacia el Bróker



En la Figura 28, se puede verificar un ejemplo de mensajes que se muestra en la consola, donde indican valores como el tópico de las variables, así como el estado respectivo de la misma.

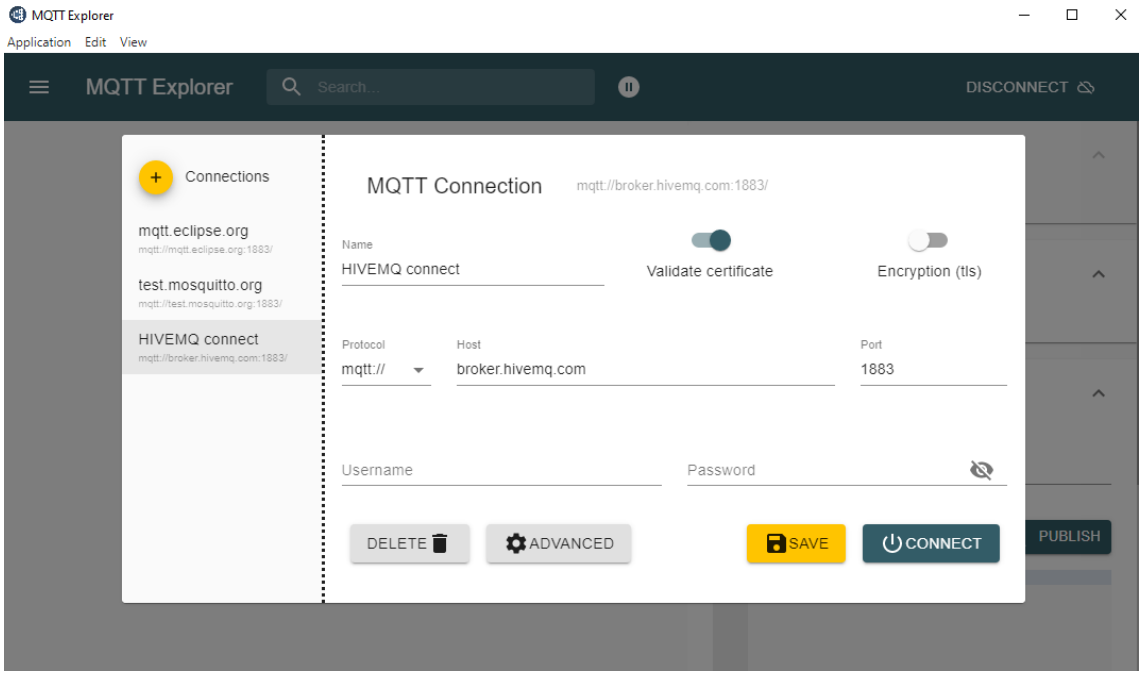
Figura 28: Ejemplo de resultado en consola del mensaje al Bróker MQTT



d. ***Verificar llegada de mensajes hacia el Bróker MQTT***: Para la verificación de que los mensajes están llegando correctamente al Bróker MQTT, se utiliza la aplicación MQTT Explorer en la cual se realiza la subscripción al tópico contenedor “Modulo4.0” y verificamos los mensajes recibidos desde el PC-servidor a través de la plataforma NODE-RED.

En la Figura 29, se indica la configuración en la aplicación MQTT Explorer al Broker, en la misma se identifica la dirección del Host del bróker “bróker.hivemq.com” así como el puerto “1883” identificados en la Figura 24.

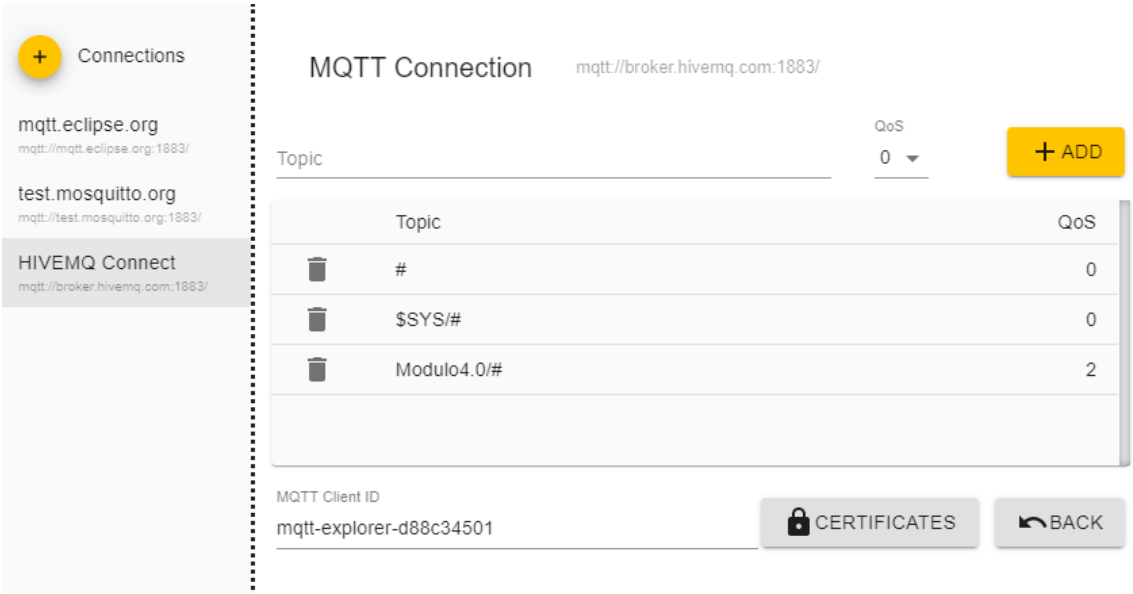
Figura 29: Configuración de aplicación MQTT Explorer para conexión con Bróker HiveMQ



Para subscribirse al tópico contenedor, se ingresa a la opción “ADVANCED”, donde se incorpora el tópico “Modulo4.0/#” y se asigna la calidad “QoS 2” tal como se muestra en la Figura 30.

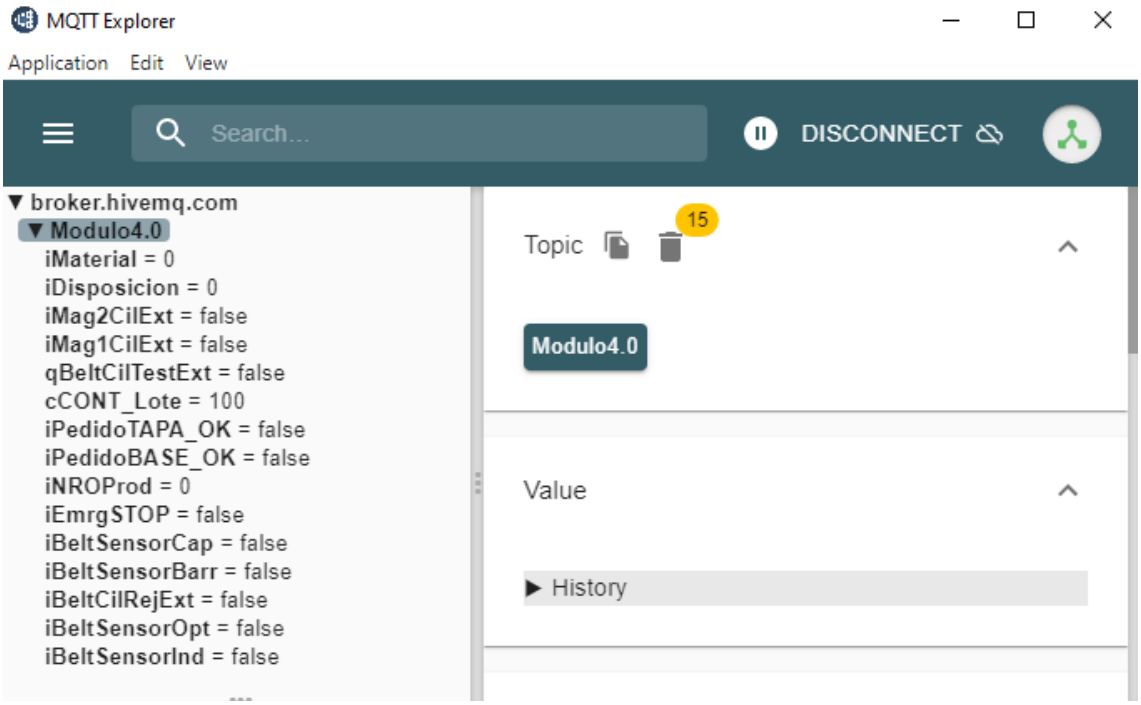
.

Figura 30: Configuración para la subscripción a tópico contenedor en el Bróker MQTT



Una vez conectado se pude verificar (Figura 31) que los mensajes están siendo recibidos por el Bróker y cualquier dispositivo que se subscriba al Bróker con el tópico indicado visualizará los estados de las variables del equipo mecatrónico.

Figura 31: Resultado de conexión con el Bróker MQTT y valores resultantes de la subscripción

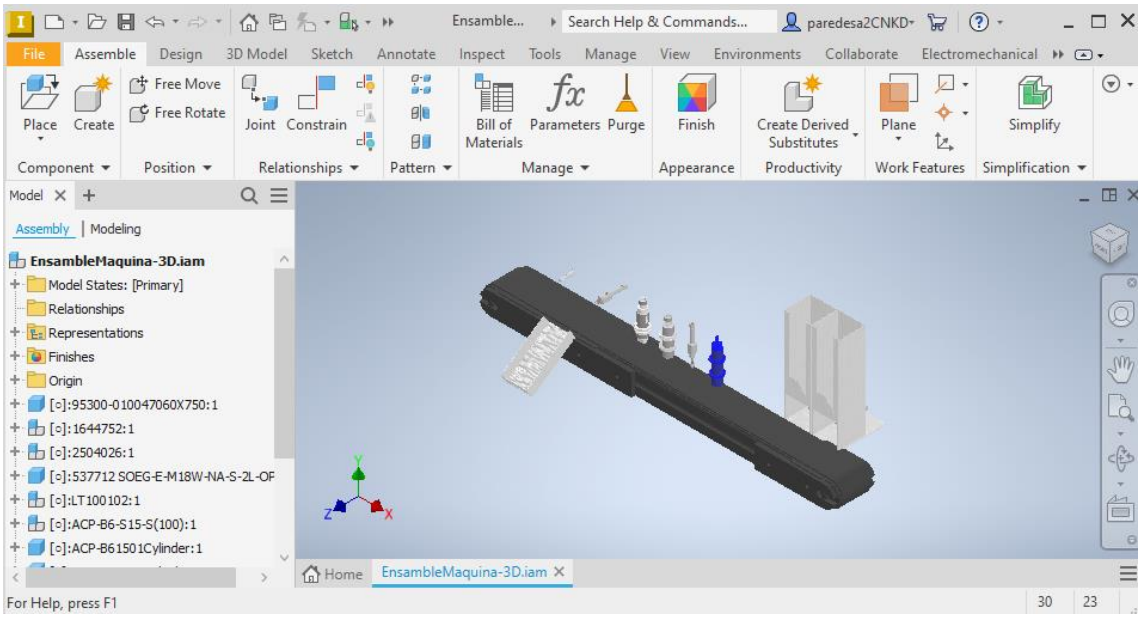


Hasta este punto se consiguió publicar a un Bróker MQTT los estados de variables importantes definidas en la Tabla 2 para que pueda ser accedido desde cualquier dispositivo, lo que culmina con la primera parte del desarrollo según lo planteado en la Figura 16.

3.4. Desarrollo de Objeto 3D

En esta sección se detalla el modelo en 3D elaborado para utilizarse como objeto en el desarrollo de la aplicación de realidad aumentada, en la Figura 32 se muestra el resultado del ensamble del modelo 3D del mismo elaborado en el software Autodesk Inventor.

Figura 32: Ensamble de objeto 3D elaborado en software Autodesk Inventor



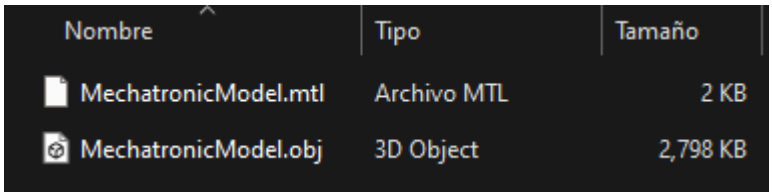
Cabe indicar que los dispositivos comerciales como: sensores, cilindros neumáticos y banda transportadora, fueron obtenidos desde la aplicación web “3D find it”, el detalle de los dispositivos utilizados de la web está listados en la Tabla 5, de los cuales la bandeja de productos descartados y los contenedores de productos para despacho fueron elaboración propia.

Tabla 5: Lista de utilizados para objeto 3D con códigos respectivos

Ítem	Variable	Código en plataforma 3D find it
1	Banda transportadora	95300-010047060X750
2	Sensor de barrera de fin de banda	LT100102
3	Sensor capacitivo	2504026
4	Sensor inductivo	1644752
5	Sensor óptico	537712 SOEG-E-M18W-NA-S-2L
6	Cilindro de testeo	ACP-B6-S15-S(100)
7	Cilindro descarte material	ACP-B6-S15-S(100)
8	Cilindro despacho 1	ACP-B6-S15-S(100)
9	Cilindro despacho 2	ACP-B6-S15-S(100)
10	Bandeja de productos descartados	Elaboración propia
11	Contenedor para despacho 1	Elaboración propia
12	Contenedor para despacho 2	Elaboración propia

Para obtener el objeto 3D listo para ser utilizado en el desarrollo de la aplicación de realidad aumentada, se requiere exportar el modelo en formato “.OBJ” el cual como resultado brinda dos archivos con extensiones “.obj” y “.mtl” según se verifica en la Figura 33.

Figura 33: Archivos del objeto 3D necesarios a ser utilizados en el desarrollo de la aplicación de realidad aumentada



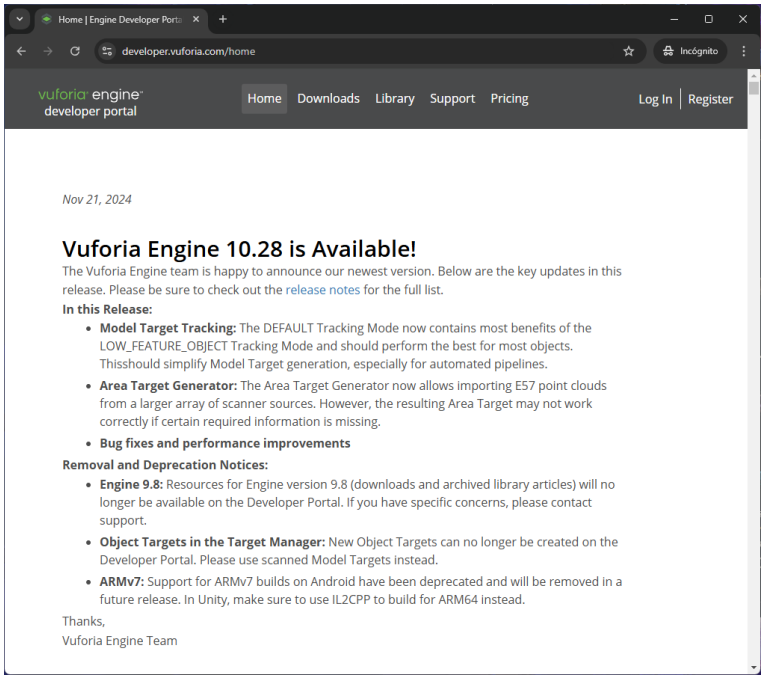
Con esto se tiene listo el modelo en 3D que permitirá simular el gemelo digital del equipo mecatrónico en la aplicación de realidad aumentada.

3.5. Generación de licencia y base de datos en el servicio Vuforia-Engine

En esta sección se detalla el procedimiento desarrollado para elaborar la aplicación de realidad aumentada, para ello se debe tener en cuenta que se requiere crear la licencia libre en el servicio Vuforia Engine para que en función a una imagen objetivo “Image Target” podamos identificar el punto de relación entre el entorno físico real y el entorno virtual (representado por el modelo 3D).

a. ***Obtención de licencia en el servicio de Vuforia Engine***: Se ingresa a la web oficial de Vuforia Engine (Figura 34), se crea una cuenta, inicia sesión para ingresar al panel de control.

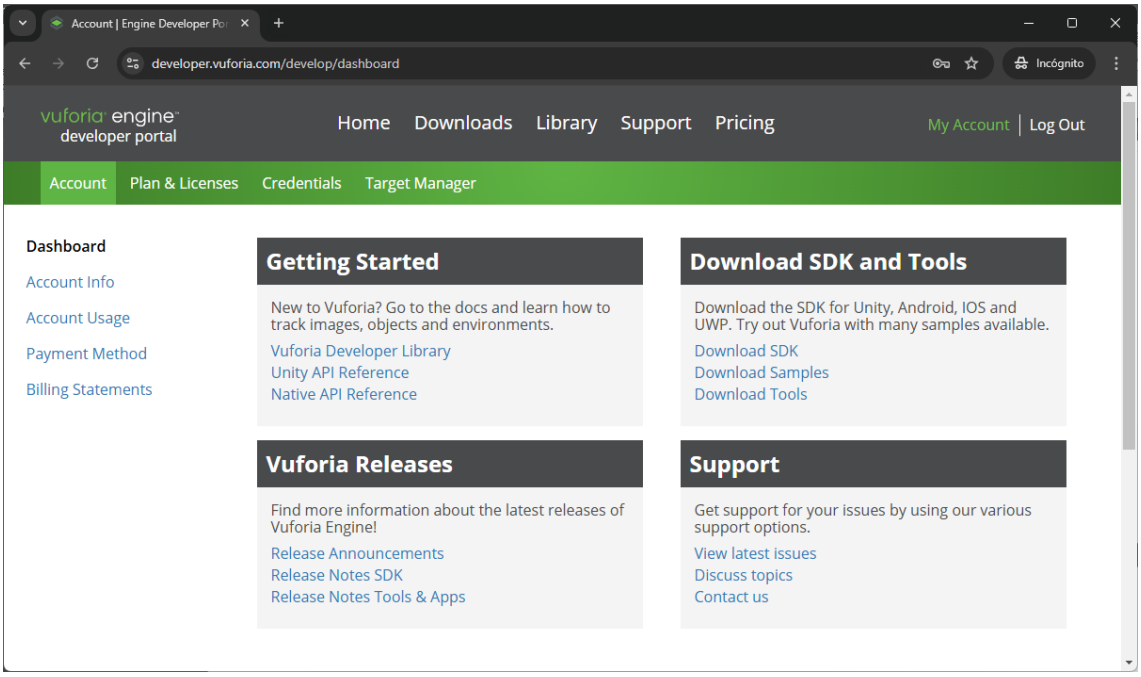
Figura 34: Ingreso a Servicio de Vuforia para desarrolladores



Fuente: https://developer.vuforia.com/home

En el panel de control, ingresar a la sección “Plan & Licenses” para gestionar una primera licencia que permita su uso (Figura 35).

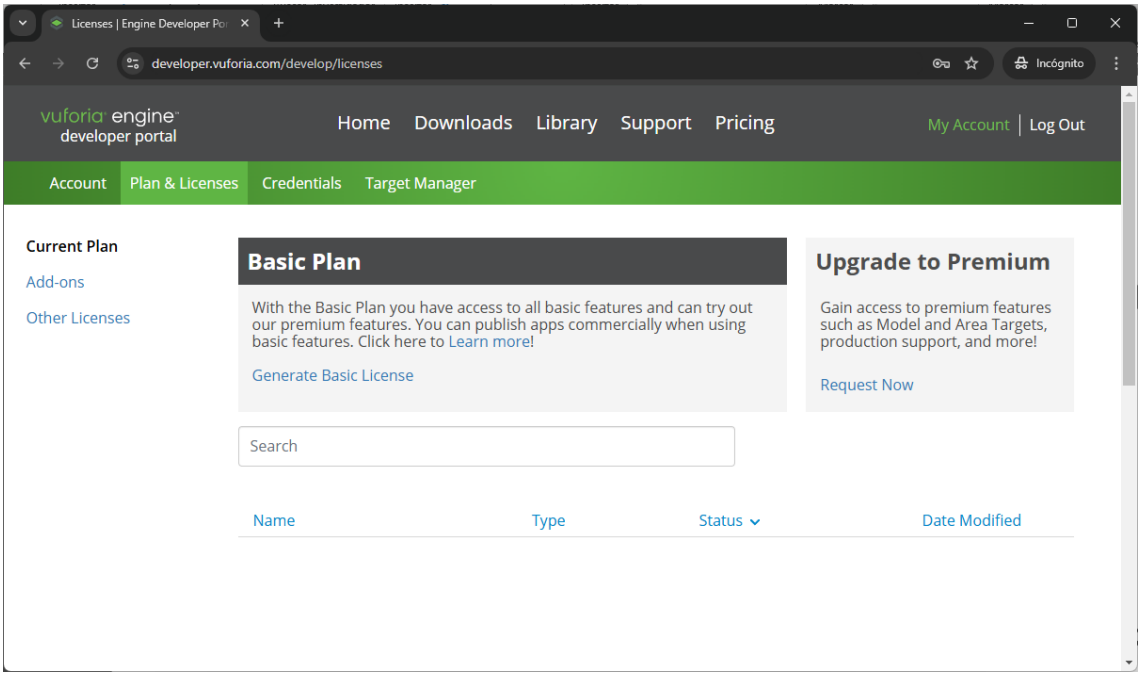
Figura 35: Dashboard principal de Vuforia Engine para desarrolladores



Fuente: https://developer.vuforia.com/develop/dashboard

Para crear una licencia dirigirse a la sección “Generate Basic License” (Figura 36).

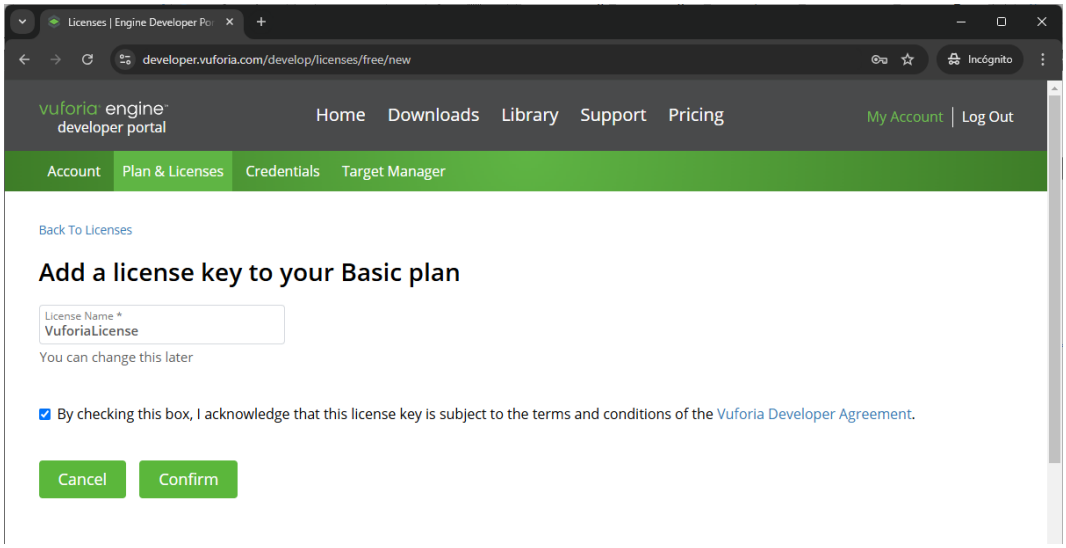
Figura 36: Sección de licencias del servicio Vuforia Engine



Fuente: https://developer.vuforia.com/develop/licenses

Introducir un nombre para la licencia y confirmar el check de las condiciones de uso del “license key” (Figura 37).

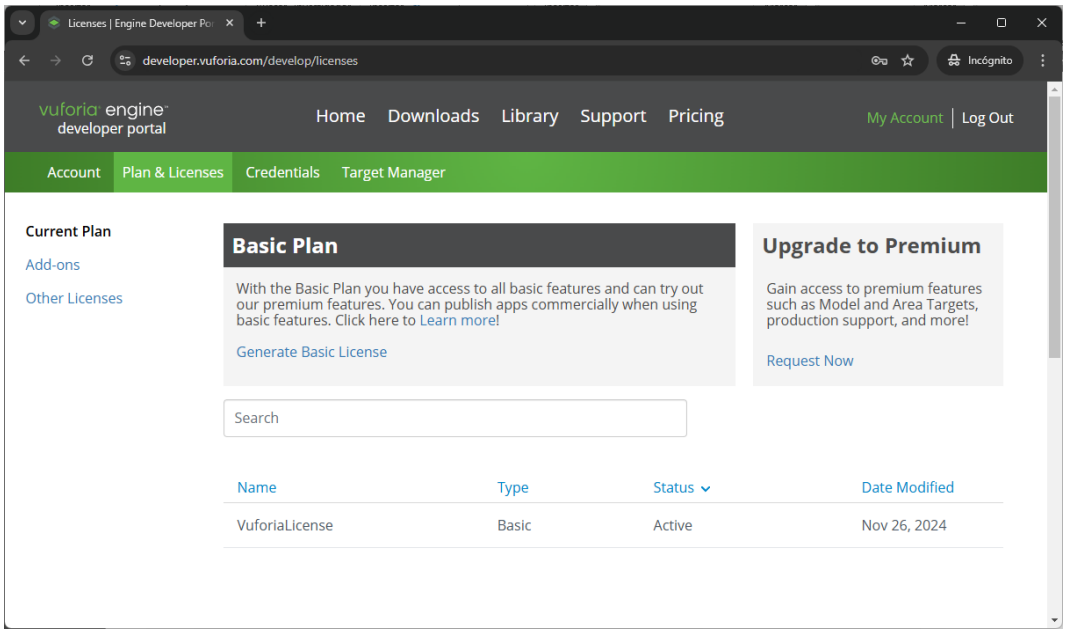
Figura 37: Creación de licencia básica



Fuente: https://developer.vuforia.com/develop/licenses/free/new

Luego, ya se cuenta con la licencia básica creada como en la Figura 38 donde se verifica en la lista de licencias la creada recientemente. Ingresar a la licencia para verificar las claves de acceso.

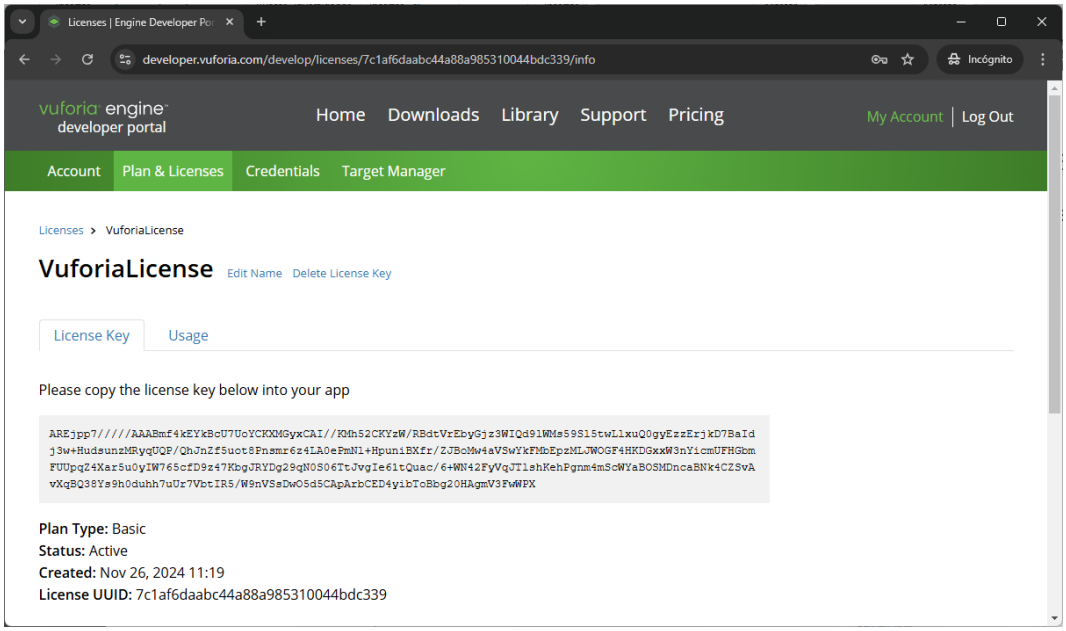
Figura 38: Licencia creada



Fuente: https://developer.vuforia.com/develop/licenses

Dentro de la información de la licencia creada, copiar el “License Key” (Figura 39) la cual permitirá conectar la aplicación de realidad aumentada con el servicio de Vuforia Engine.

Figura 39: Identificación del License Key



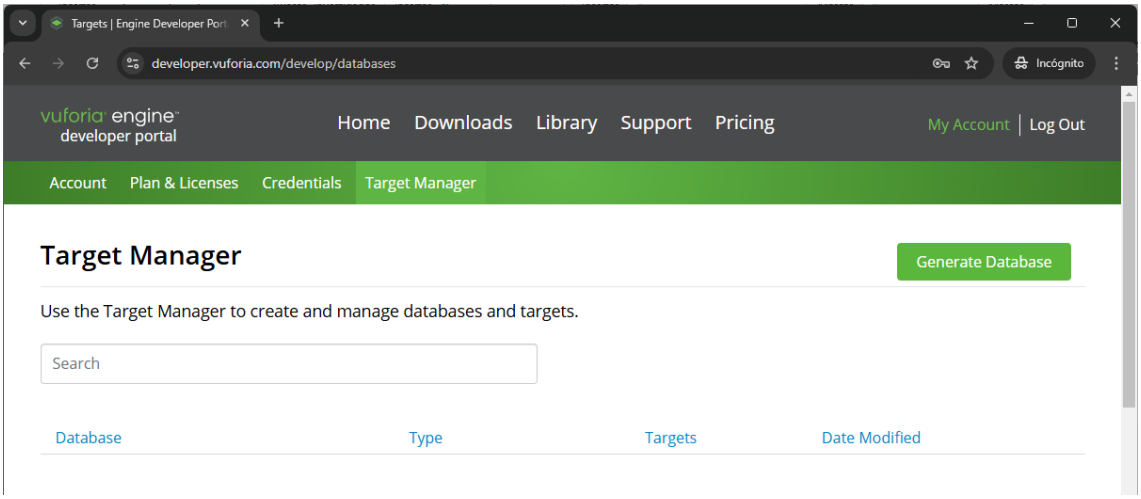
Fuente: https://developer.vuforia.com/develop/licenses

b. ***Creación de Image Target y descarga de package***: Para la creación de una Image Target, se requiere contar con una imagen de alta calidad, como por ejemplo la Figura 40.

Figura 40: Imagen para utilizar como Image Target

Contando con la imagen, se dirige a la sección “Target Manager” de la web de Vuforia Engine para proceder a cargarla (Figura 41), dirigirse al botón “Generate Database”.

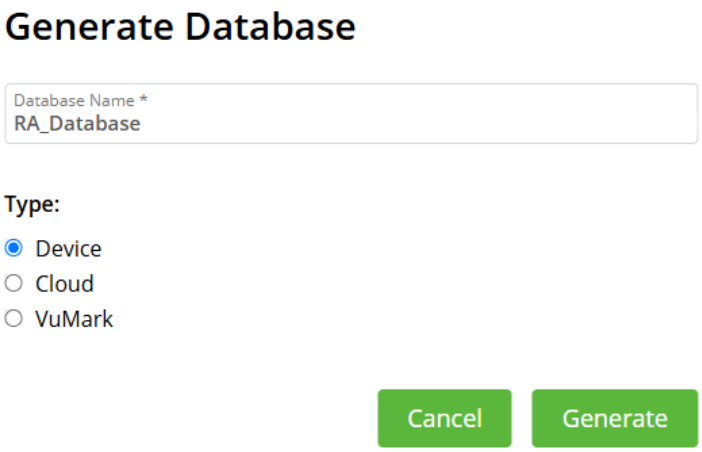
Figura 41: Sección Target Manager del servicio Vuforia Engine



Fuente: https://developer.vuforia.com/develop/databases

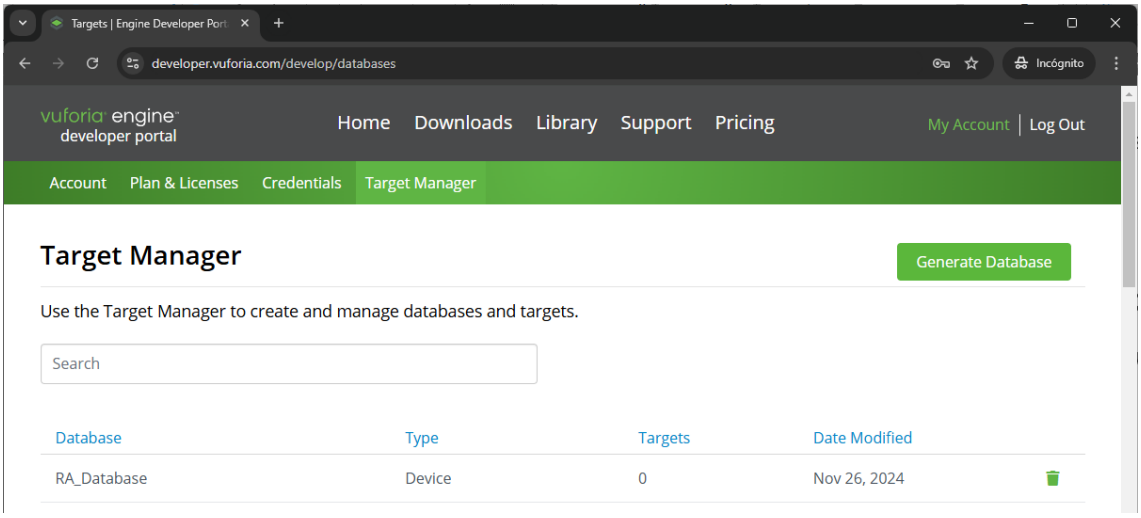
Introducir el nombre a asignar al Database y seleccionar el tipo como Device (Figura 42), luego darle al botón Generate.

Figura 42: Creación del Database



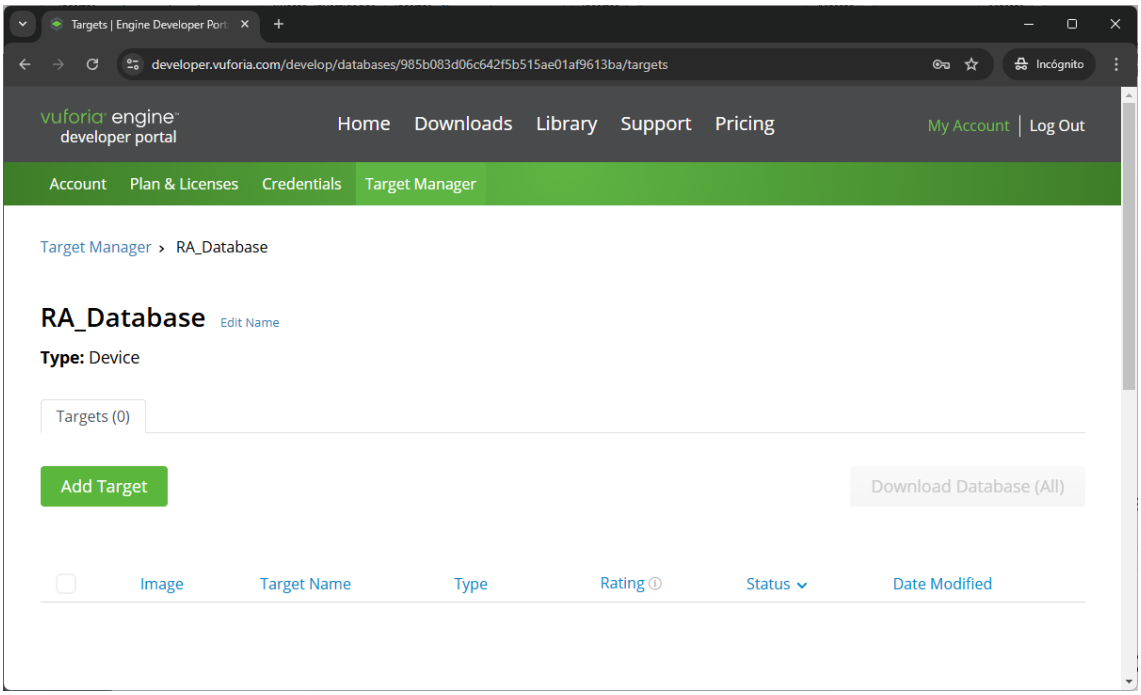
En la sección de “Target Manager” aparece el Database creado, ingresar al mismo para poder realizar la creación del “Target Manager” (Figura 43).

Figura 43: Database creado - 1



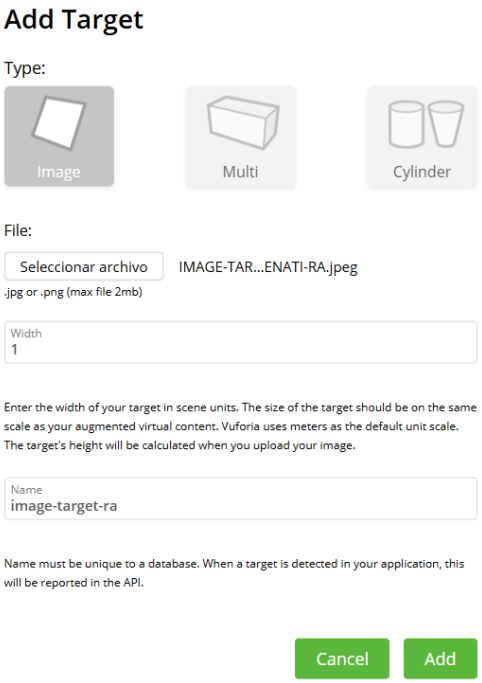
Ingresado al Database creado, dirigirse al botón “Add Target” para cargar la imagen preparada (Figura 44).

Figura 44: Database creado - 2



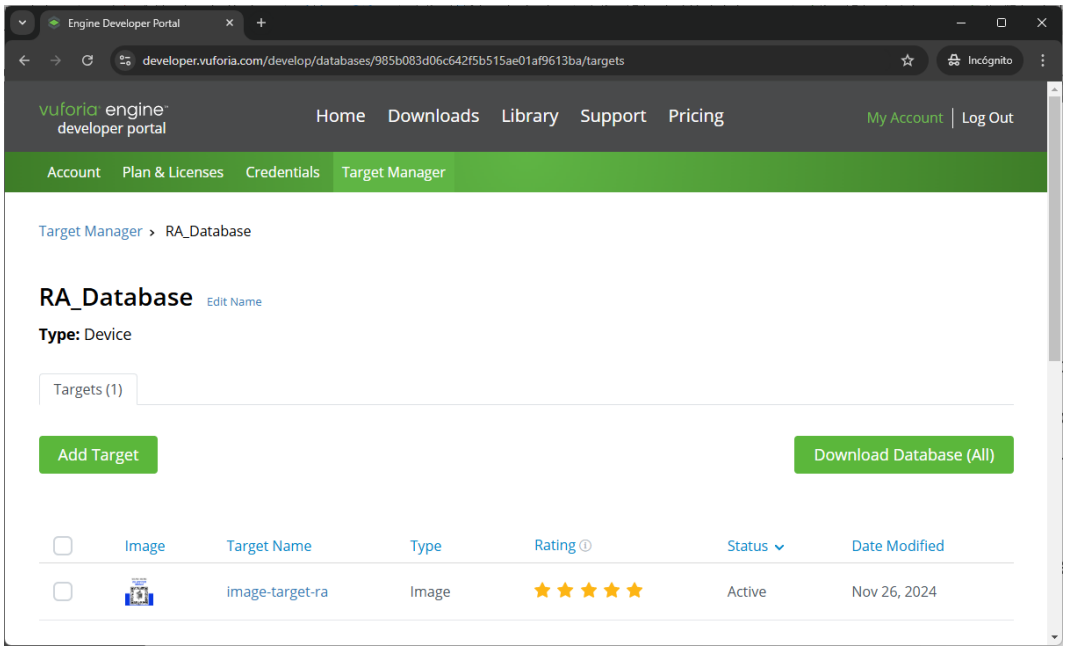
Seleccionar el tipo de Target Image, cargar el archivo preparado (de preferencia en formato .jpeg), asignar un ancho de 1 unidad e indicar el nombre que tendrá el Image Target dentro del Database creado (Figura 45), terminar dando al botón “Add”.

Figura 45: Creación de Image Target



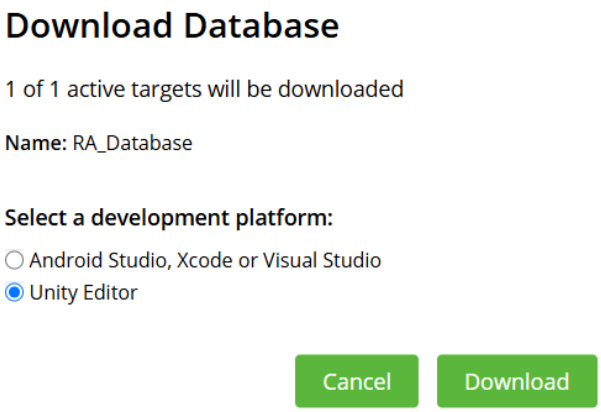
Ahora se verifica el Image Target agregado al Database, verificar actualizando la página que el "Rating" asignado por el servicio a la imagen llegue a las 5 estrellas, con ello garantiza un eficiente reconocimiento para el enlace de la aplicación de realidad aumentada (Figura 46), ahora dar en el botón "Download Database (All).

Figura 46: Image Target creada en Database



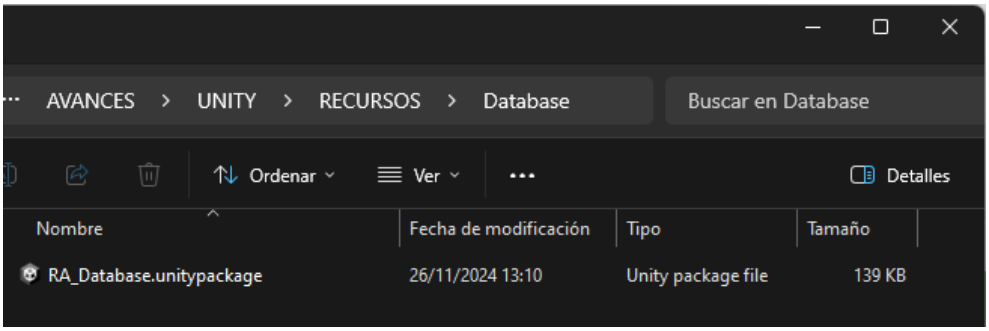
Seleccionar como plataforma de desarrollo “Unity Editor” y continuar con el botón “Download” (Figura 47).

Figura 47: Descarga de Database



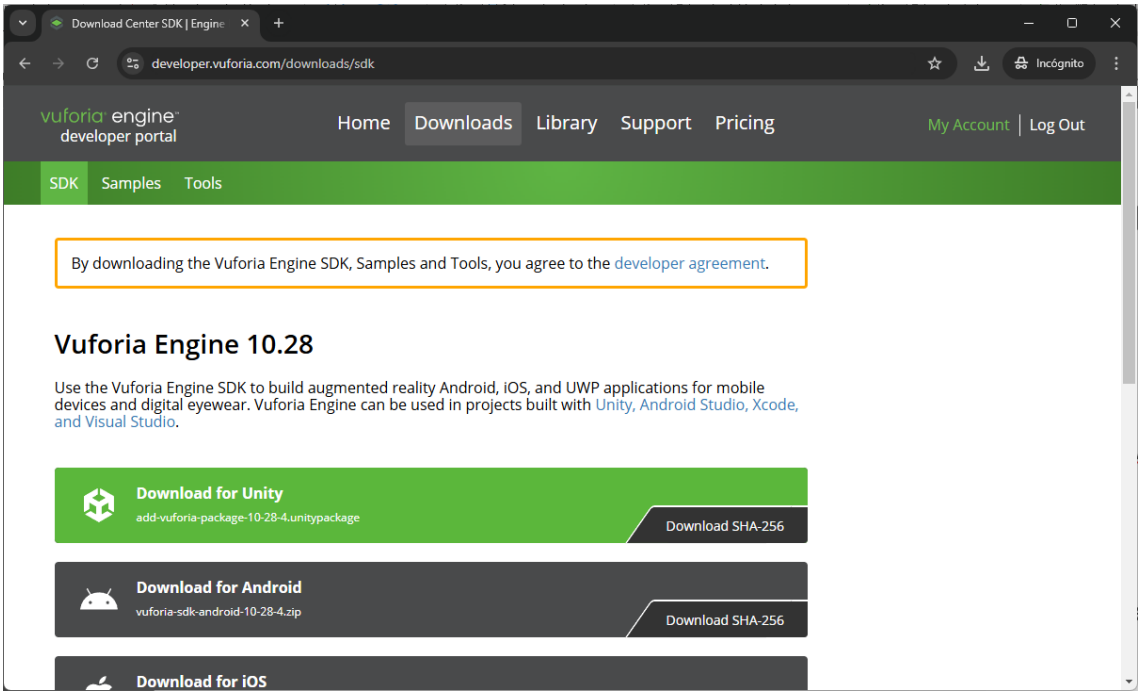
Guardar el archivo “.unitypackage” en una carpeta que luego será utilizada en la plataforma de la aplicación de realidad aumentada.

Figura 48: Guardar Database en carpeta de recursos



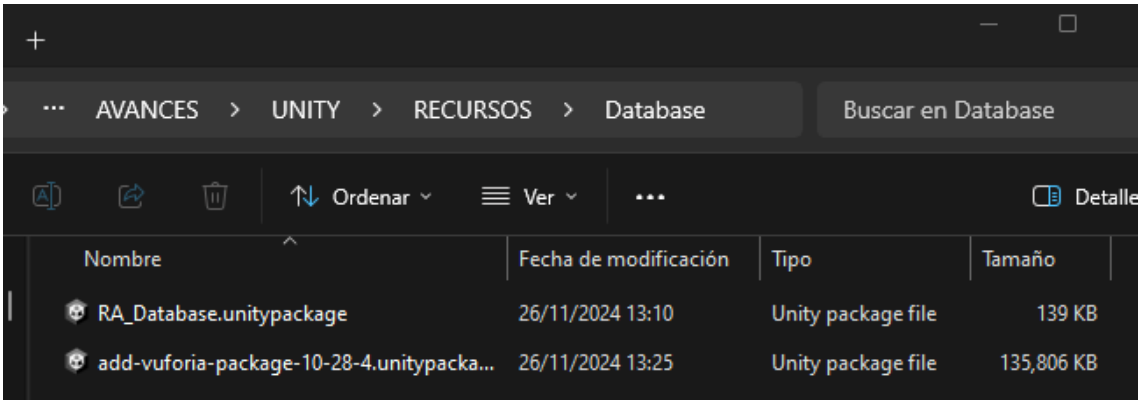
Finalmente toca descargar el paquete que relaciona el servicio de Vuforia Engine con la plataforma Unity, para ello dirigirse a la sección “Downloads” de la página principal de Vuforia (Figura 49), se descarga el archivo Vuforia Engine SDK que permitirá la creación de aplicaciones de realidad aumentada para Android, darle a la sección “Download for Unity”.

Figura 49: Descarga de SDK para Unity



Guardar el archivo descargado en la carpeta junto al Database (Figura 50). Con ello se podrá realizar la creación de la aplicación de realidad aumentada en la siguiente sección.

Figura 50: Package Vuforia en carpeta de recursos – parte 1

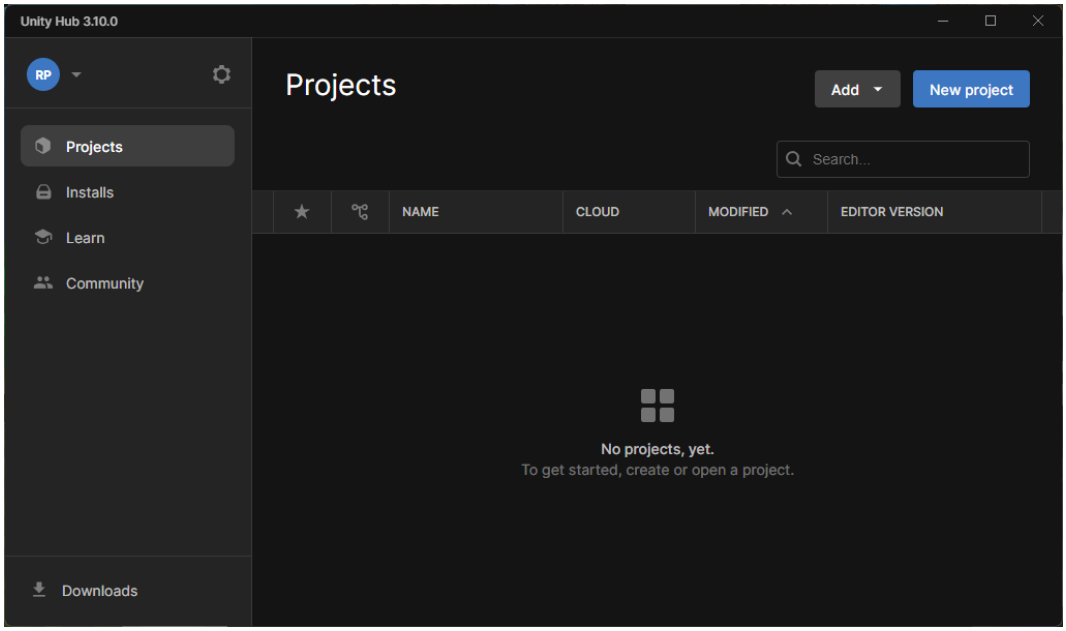


3.6. Creación de aplicación de realidad aumentada con Unity

Se detalla el procedimiento para la creación de la aplicación móvil de realidad aumentada con la plataforma Unity. Se comienza con la preparación de plataforma, inserción de objetos a escena y la creación de menú y botones de navegación.

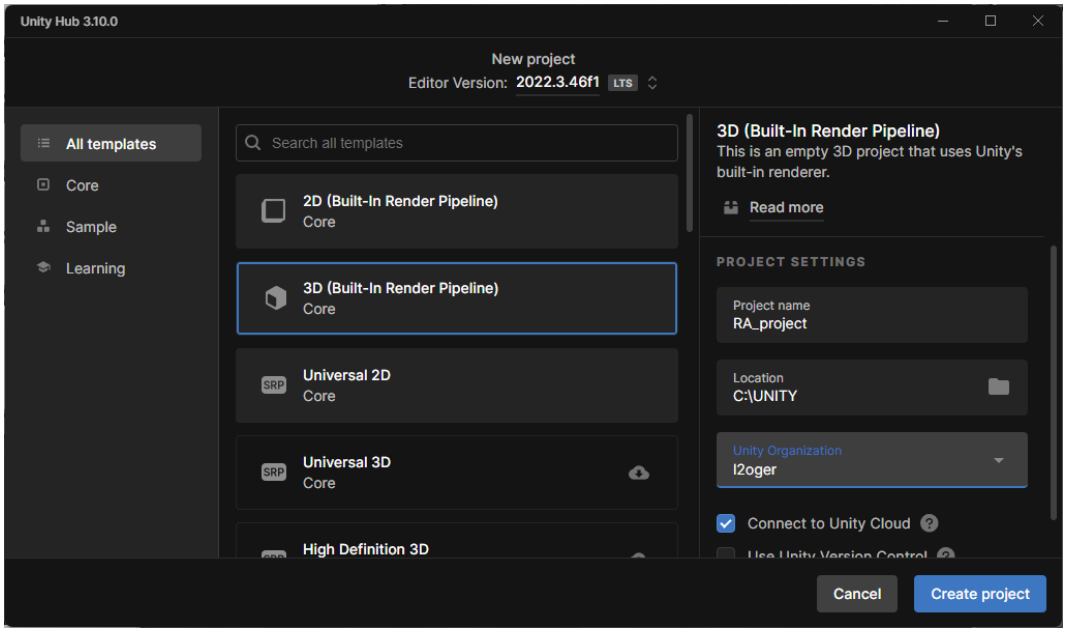
a. ***Creación de proyecto y configuración de plataforma***: Una vez abierta la plataforma Unity Hub, crear un nuevo proyecto en el botón “New project” (Figura 51).

Figura 51: Package Vuforia en carpeta de recursos - parte 2



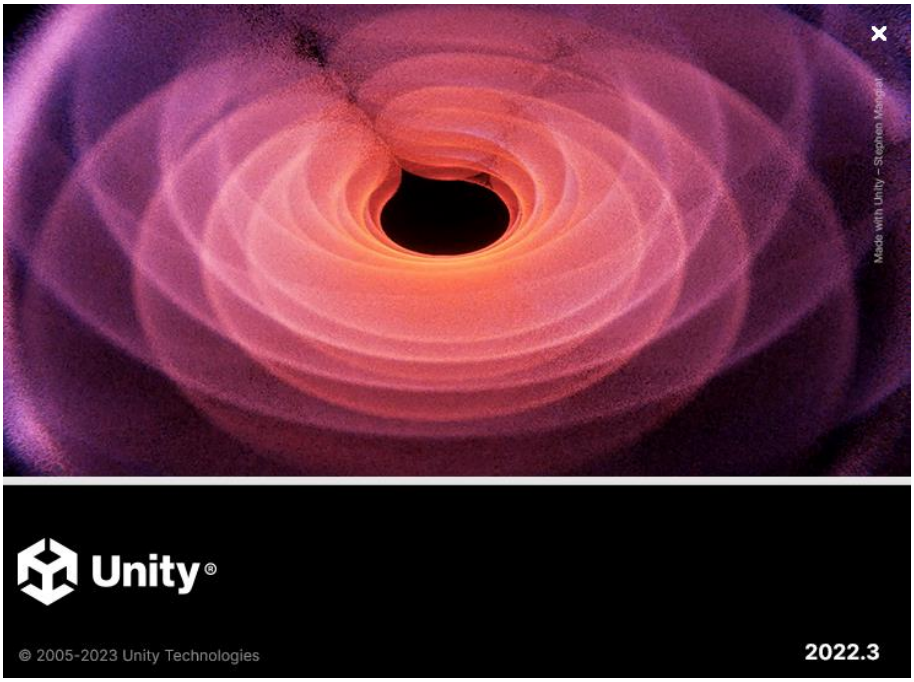
Verificar la versión de Unity en la parte superior, seleccionar la plantilla “3D (Built-in Render Pipeline) Core”, asignar un nombre al proyecto, la carpeta donde se ubica el proyecto y dar en botón “Create Project” (Figura 52).

Figura 52: Package Vuforia en carpeta de recursos - parte 3



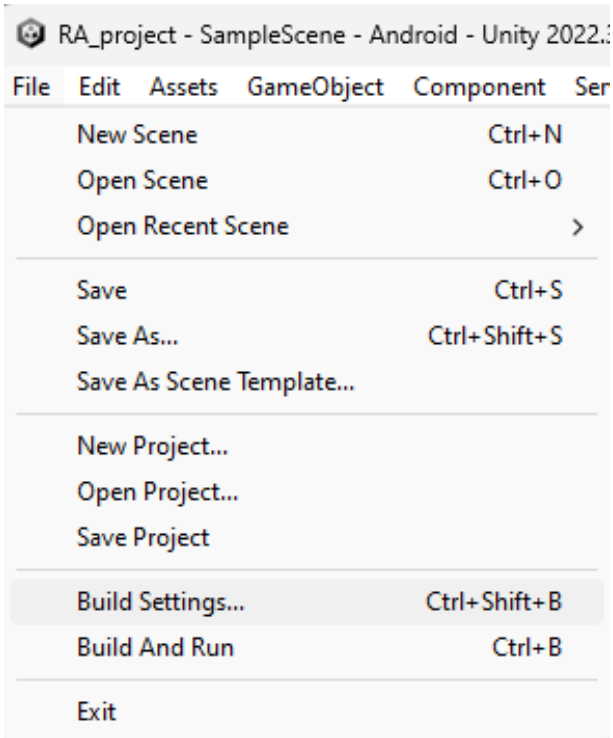
Se abre la ventana de carga del Unity, esta carga demora unos minutos (Figura 53).

Figura 53: Package Vuforia en carpeta de recursos - parte 4



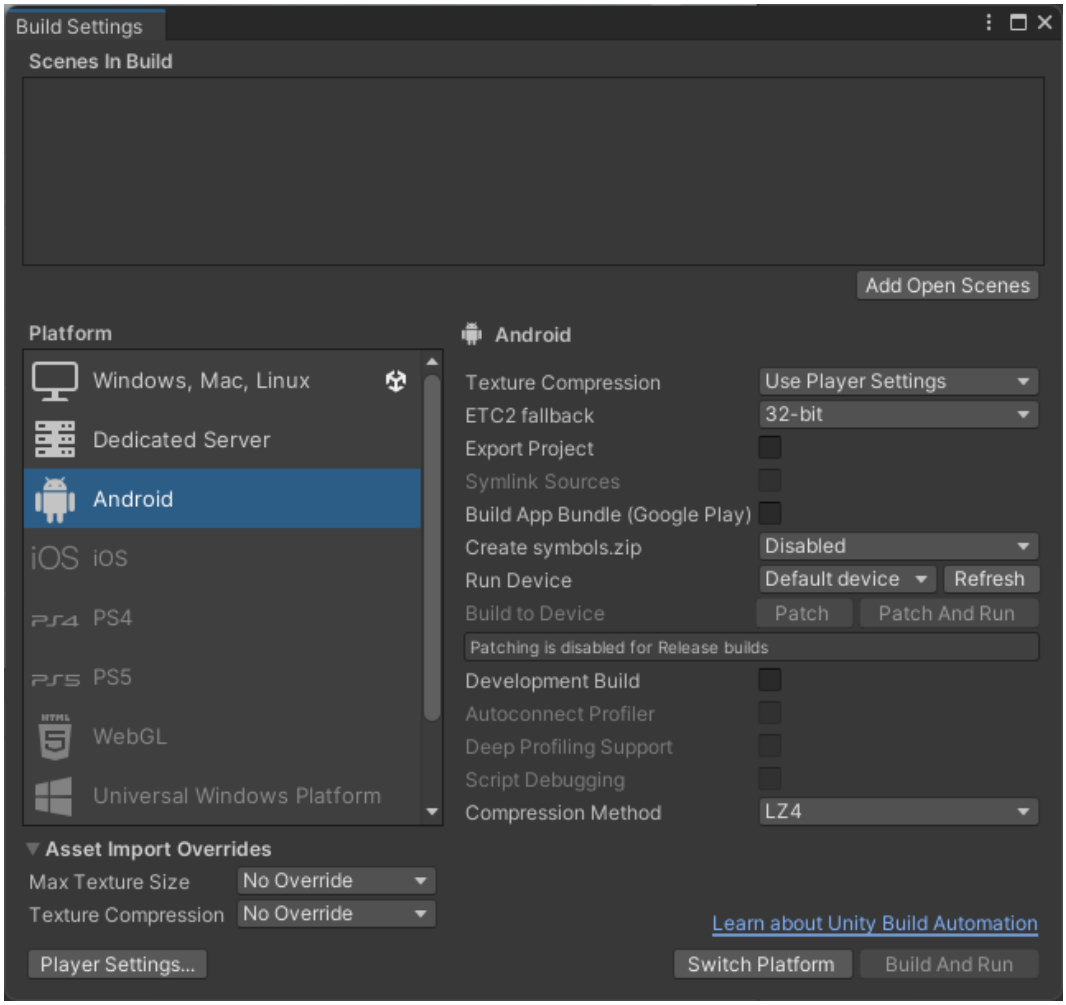
Se inicia la plataforma Unity dirigirse a la sección File -> Build Settings…, para configurar la aplicación (Figura 54).

Figura 54: Package Vuforia en carpeta de recursos - parte 5



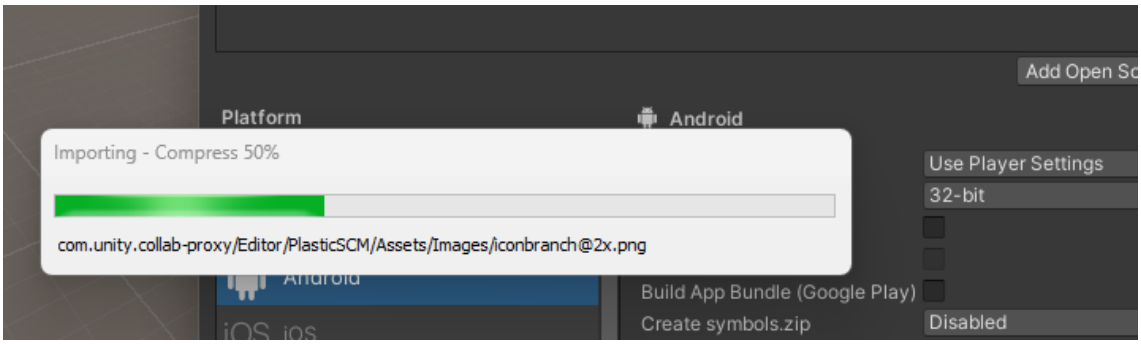
Seleccionar la plataforma Android y dar al botón “Switch Platform” (Figura 55).

Figura 55: Package Vuforia en carpeta de recursos - parte 6



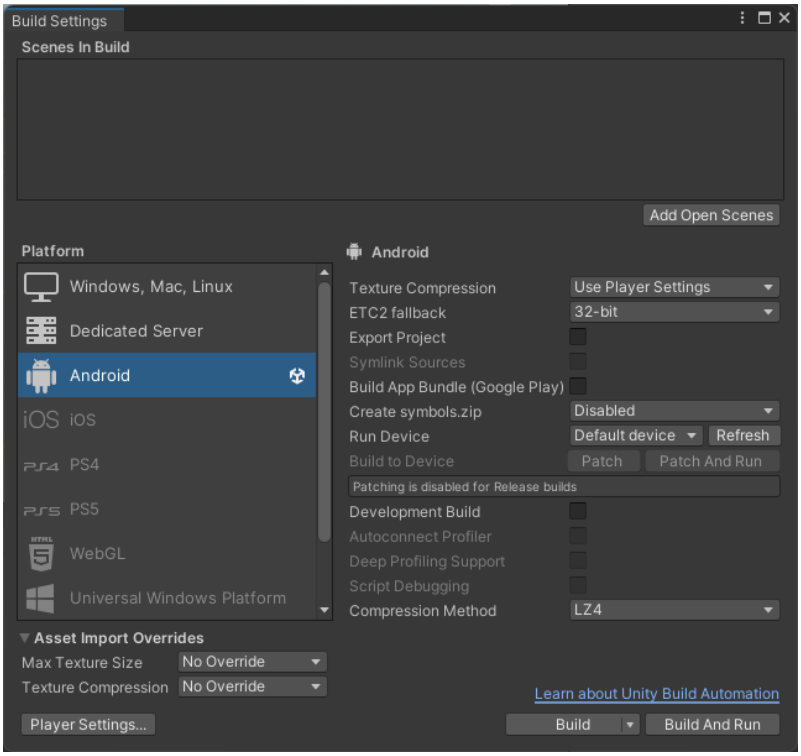
Se realiza la importación de las librerías para trabajar con aplicaciones para Android (Figura 56).

Figura 56: Package Vuforia en carpeta de recursos - parte 7



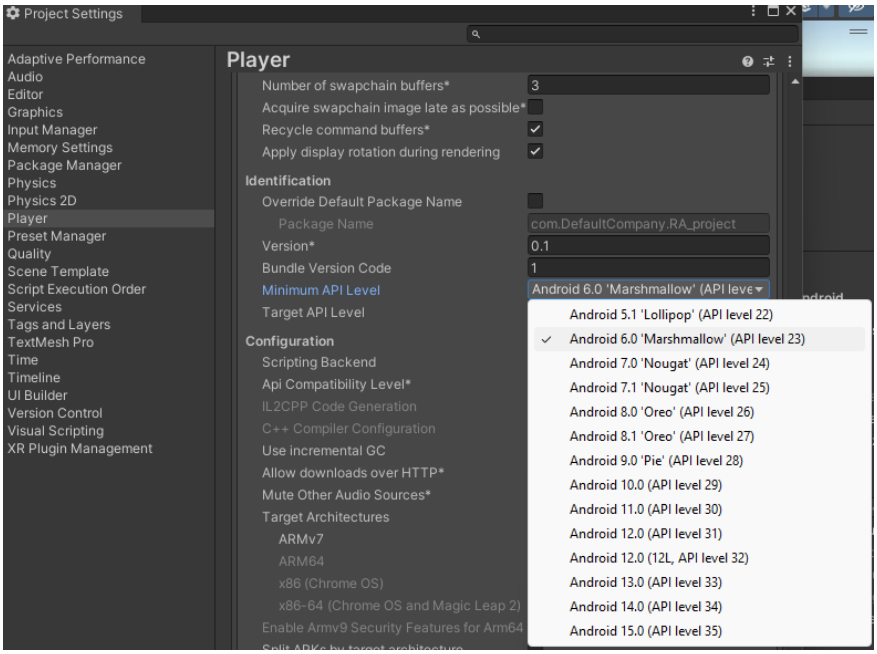
Una vez cargada la nueva plataforma, dar al botón “Player Settings…” (Figura 57).

Figura 57: Package Vuforia en carpeta de recursos - parte 8



Ubicar la sección de “Identification” y en el campo “Minimum API Level” seleccionar la versión “Android 6.0 Mashmallow” (Figura 58), tener en cuenta que esto corresponde a la menor versión de Android con la cual será compatible la aplicación. Proceder a cerrar la ventana.

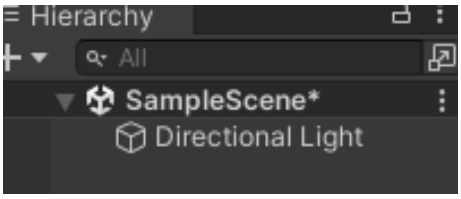
Figura 58: Package Vuforia en carpeta de recursos - parte 9



b. ***Desarrollo de conexión para aplicación de realidad aumentada con objeto 3D***: Para desarrollar la aplicación, en esta etapa corresponde a enlazar el mundo físico con el mundo virtual, para ello se agregará la cámara de realidad aumentada de Vuforia y también el objeto 3D.

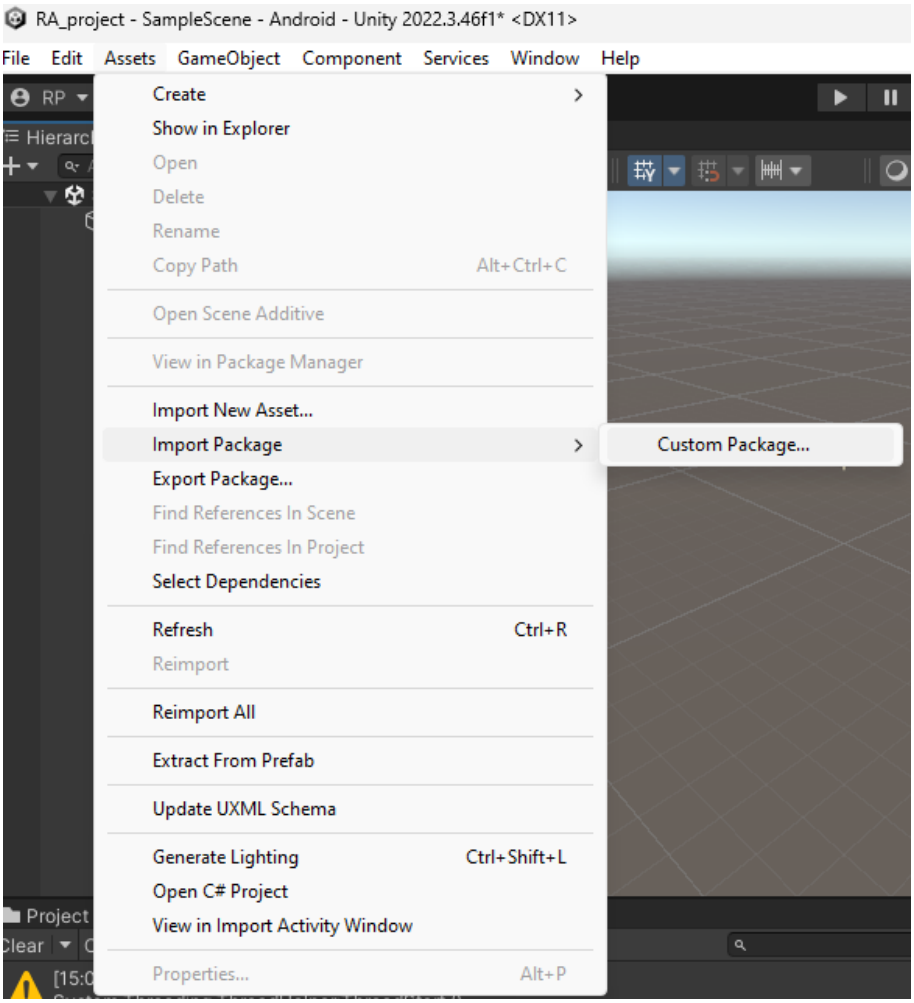
Eliminar el objeto “Camera” que aparece por defecto (Figura 59).

Figura 59: Escena con Camera eliminada



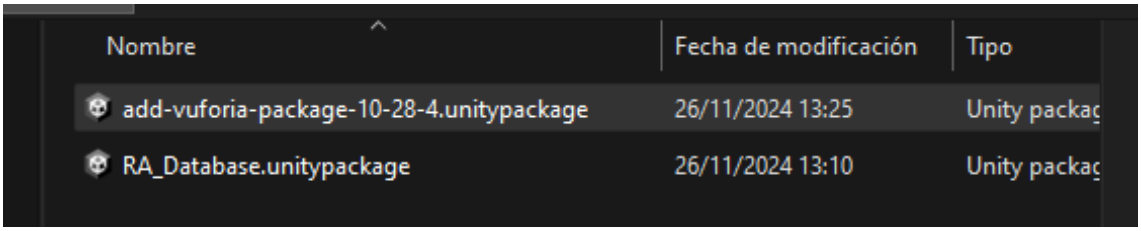
Dirigirse a la sección Assets -> Import Package -> Custom Package, para agregar el SDK de Vuforia descargado previamente (Figura 60).

Figura 60: Importación de Paquete SDK para Vuforia



Verificar que corresponda al SDK descargado, mostrado en la Figura 50, ver Figura 61.

Figura 61: Package Vuforia en carpeta de recursos - parte 10



Confirmar en la ventana y dar en el botón “Import” (Figura 62).

Figura 62: Package Vuforia en carpeta de recursos - parte 11

Comienza la compilación (Figura 63Figura 62) y solicita confirmación de actualización de Package.

Insertar a la Escena el elemento ubicado en Vuforia Engine -> AR Camera (Figura 64), confirmar la licencia de Vuforia, se inserta el AR Camera a la escena.

Figura 63: Compilación y confirmación de Update de Vuforia Engine Package

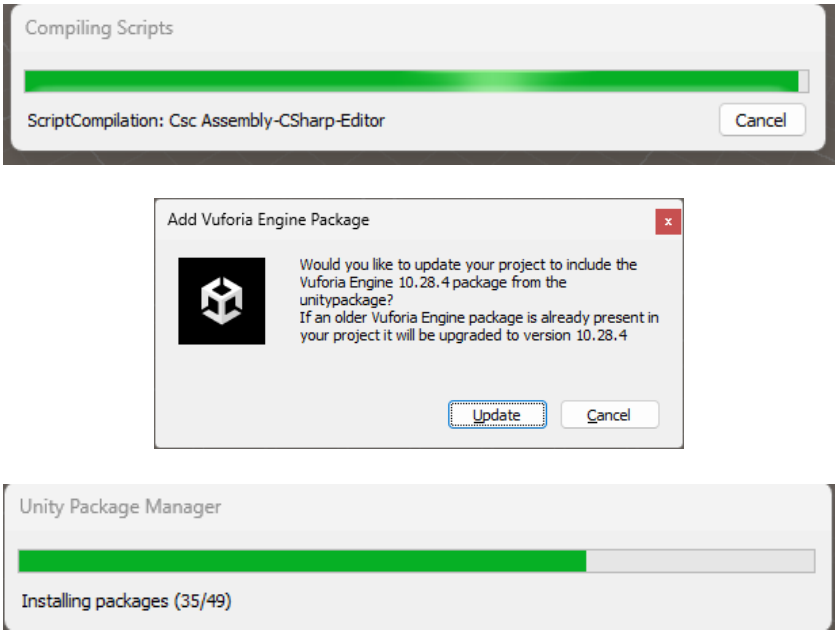
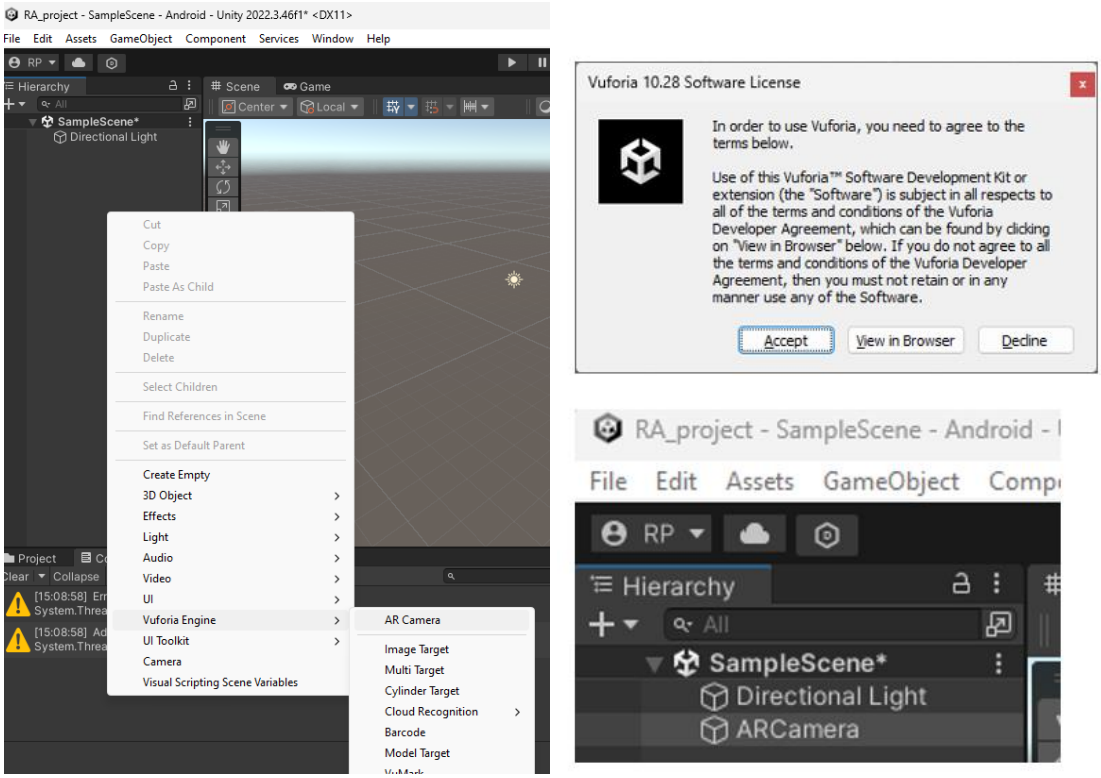
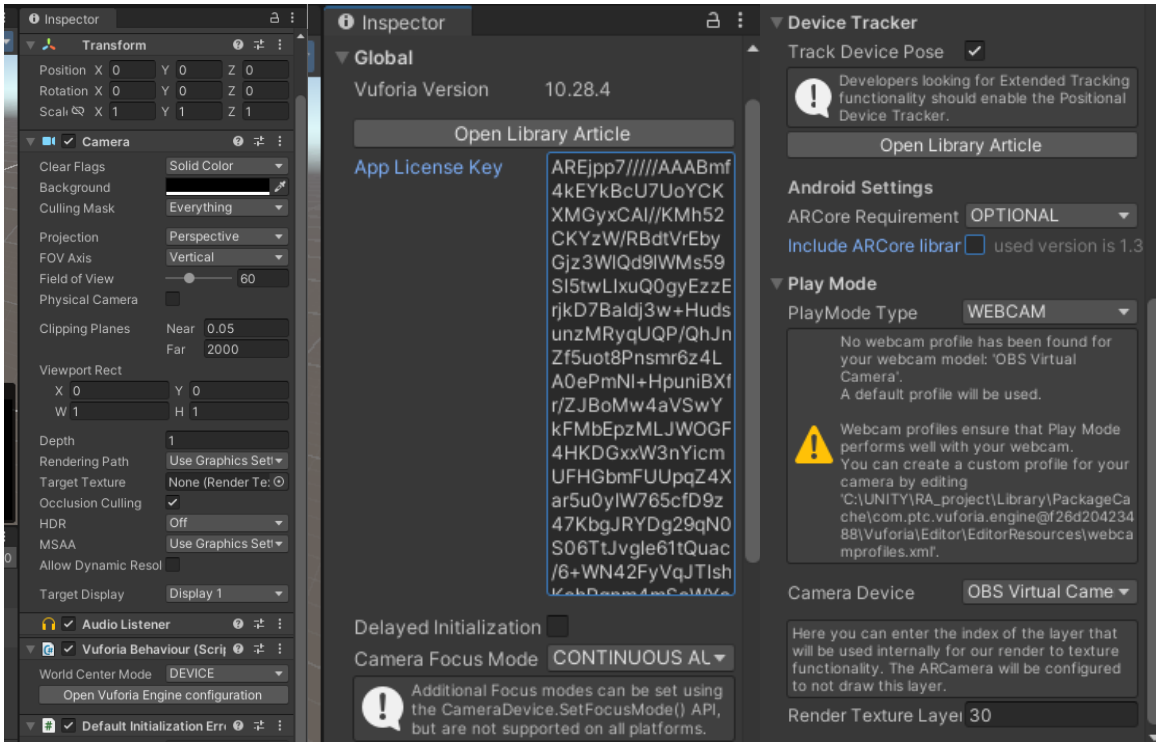


Figura 64: Inserción de AR Camera en Escena



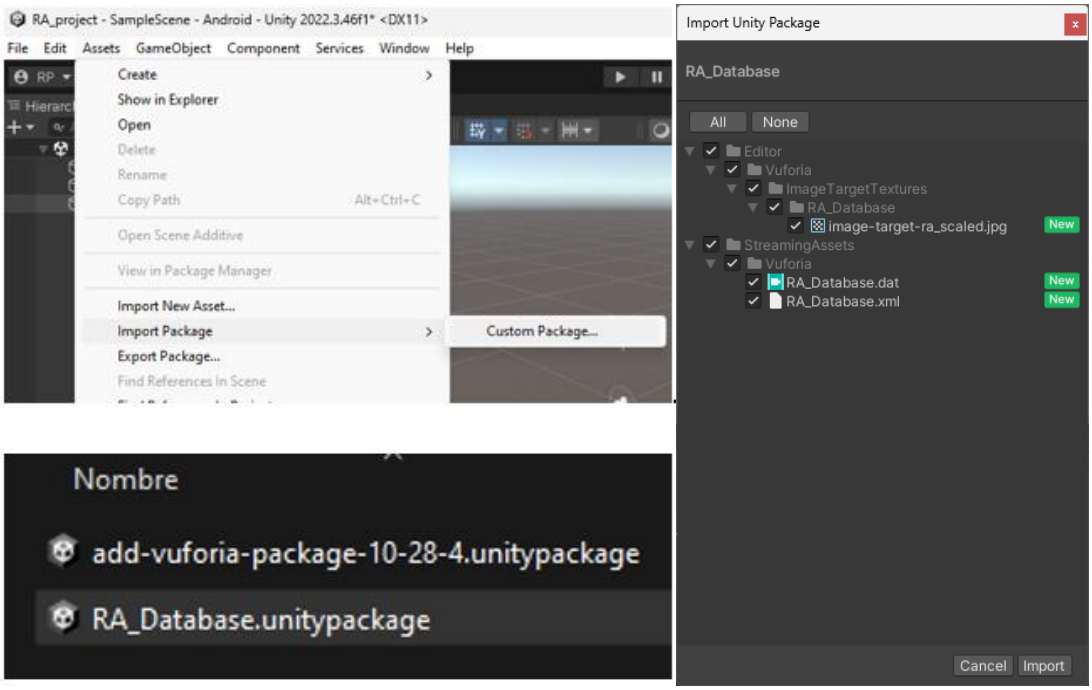
En la sección Inspector del AR Camera, darle al botón “Open Vuforia Engine Configuration”, pegar la licencia de Vuforia en el cuadro “App License Key” y desactivar el recuadro “Include AR Core Library” (Figura 65).

Figura 65: Configuración de AR Camera - parte 1



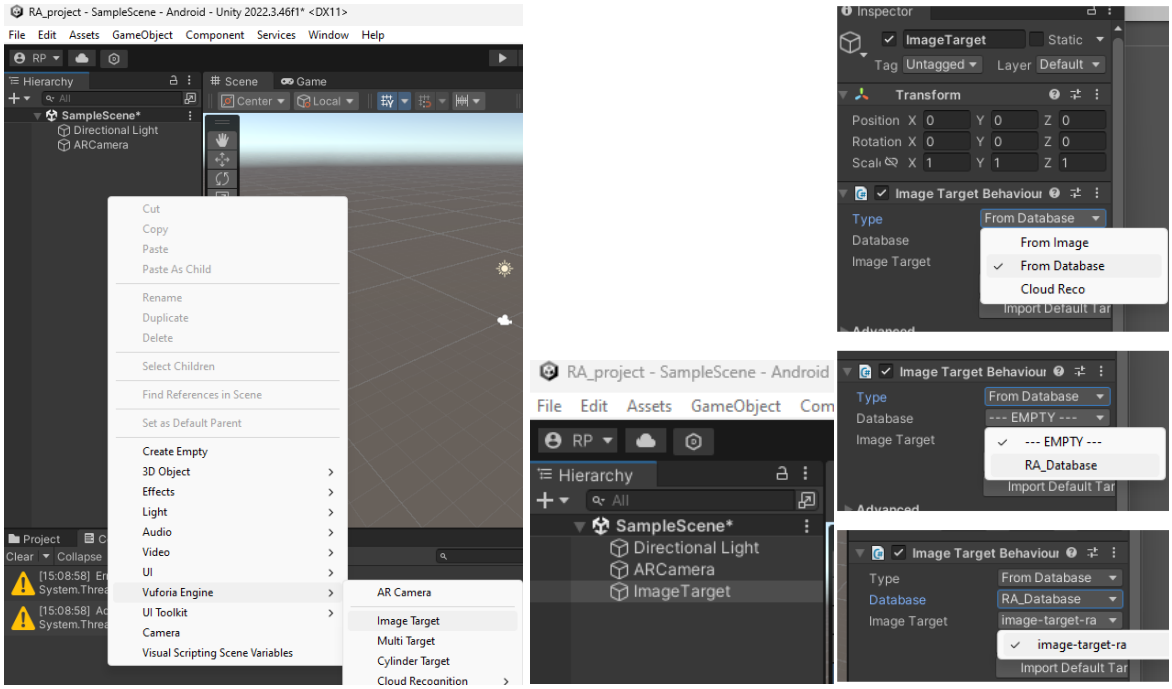
Ahora se importará el Database generado anteriormente, dirigirse a la sección Assets -> Import Package -> Custom Package, seleccionar el archivo correspondiente al Database previamente guardado, confirmar la importación (Figura 66).

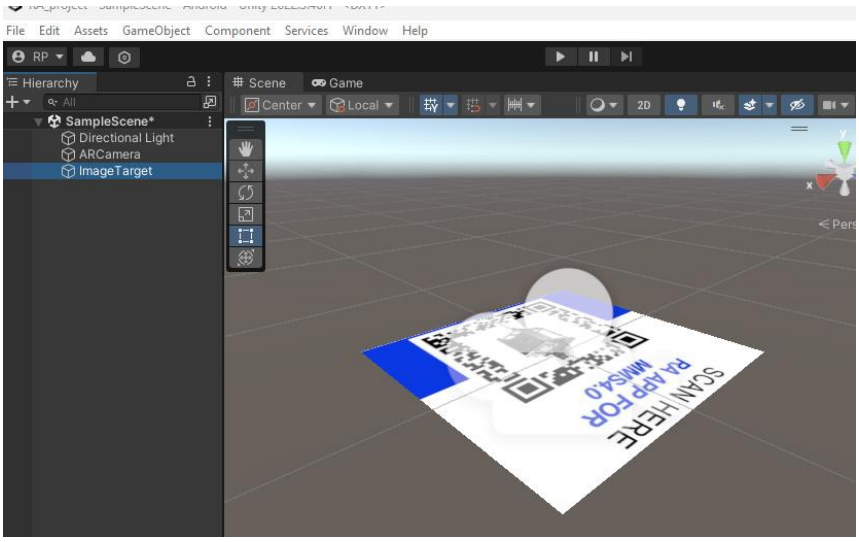
Figura 66: Configuración de AR Camera - parte 2



Insertar a la escena el Image Target ubicado en la sección Vuforia Engine -> Image Target (Figura 67), En su sección de Inspector, seleccionar el Database y el Image Target previamente descargado desde Vuforia Engine, la imagen aparecerá en la Escena.

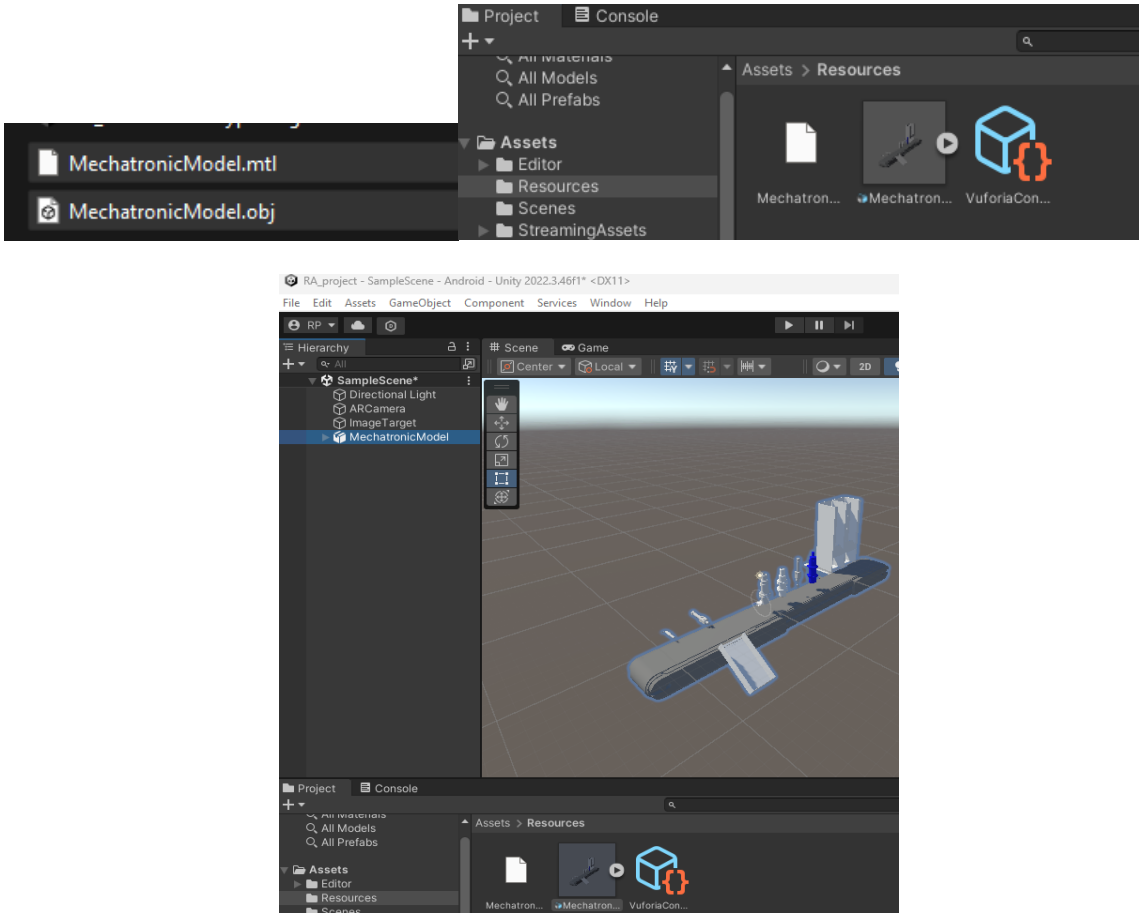
Figura 67: Inserción de Image Target





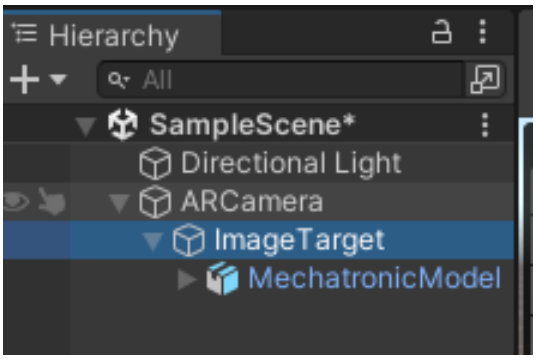
Ahora se insertará el Objeto 3D guardado en la Figura 33, arrastrar los archivos a la carpeta del Proyecto ubicado en Project -> Assets -> Resources, luego incorporarlo en la escena, y aparece en la misma (Figura 68).

Figura 68: Inserción de objeto 3D a escena



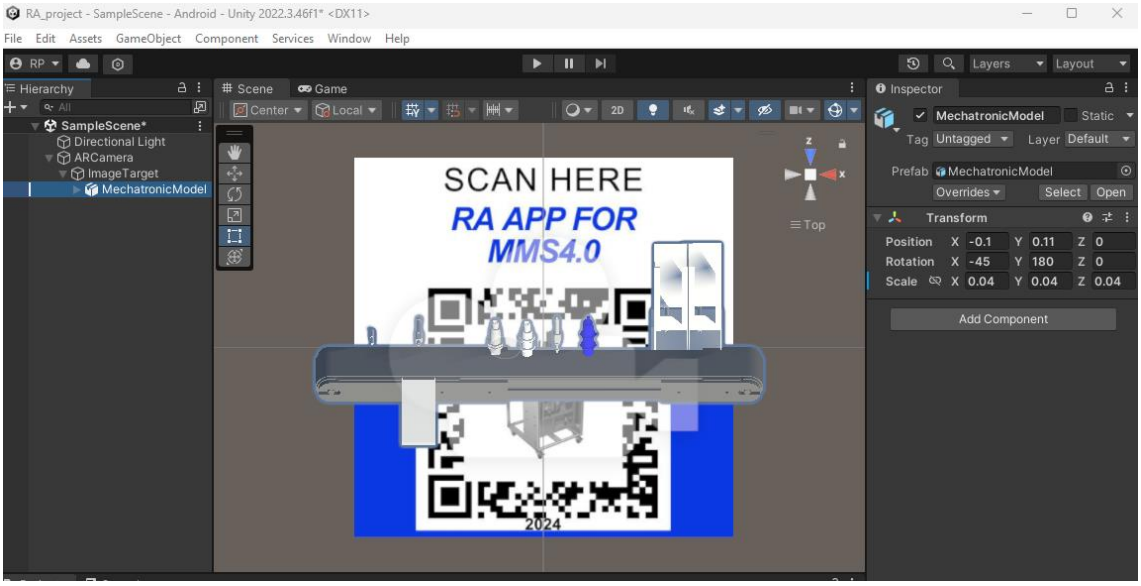
Establecer una jerarquía entre los objetos de la escena tal como se muestra en la Figura 69, basta con seleccionar y arrastrar dentro de cada objeto respectivo.

Figura 69: Establecimiento de jerarquías en los objetos



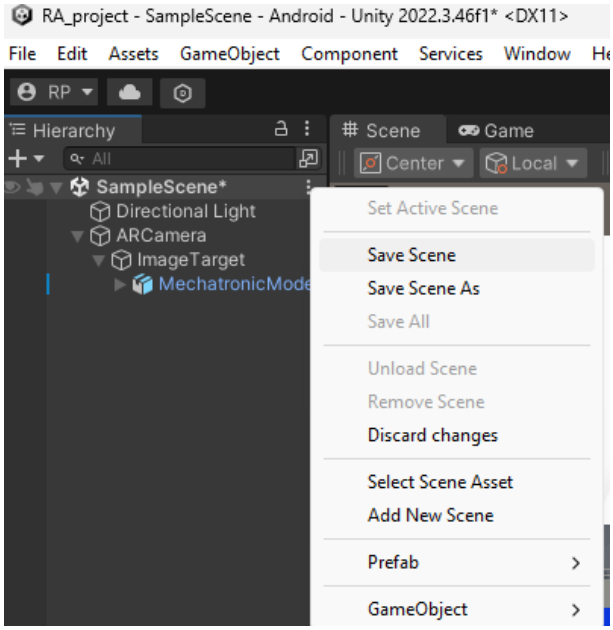
Ajustar la escala y posición del Objeto 3D, apoyarse de la sección Inspector, en la Figura 70se muestra un ejemplo de cómo puede verse con fondo el Image Target.

Figura 70: Escalamiento de Objeto 3D respecto al Image Target



Guardar los cambios realizados en la escena dando en la sección Sample Scene -> Save Scene (Figura 71).

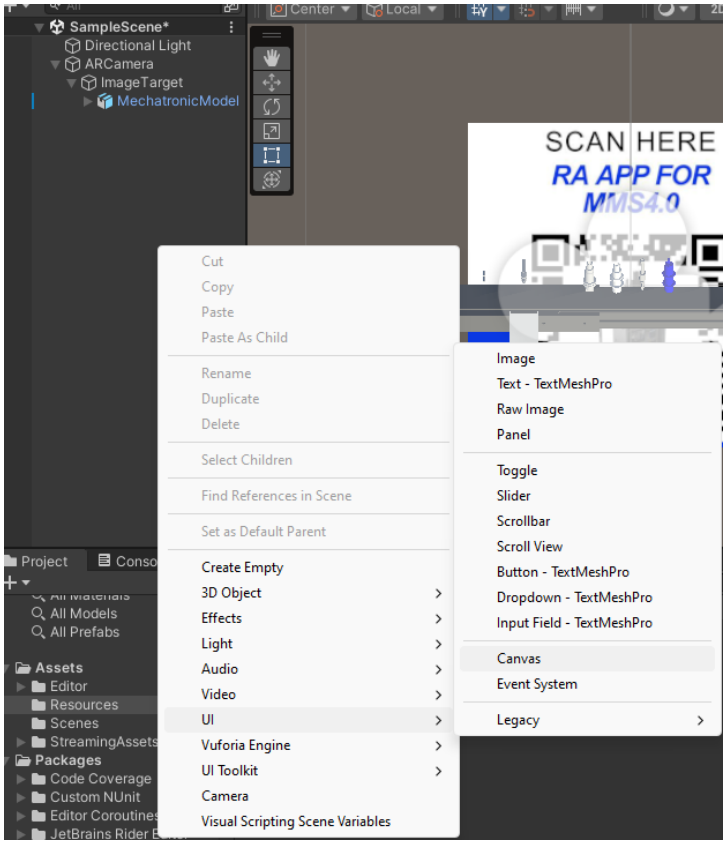
Figura 71: Guardar Escena



c. ***Creación del menú y ventanas de navegación para la aplicación***: Ahora se realizará las ventanas de navegación a través de un menú, esto permite poder mostrar diferente contenido como datos, el objeto 3D.

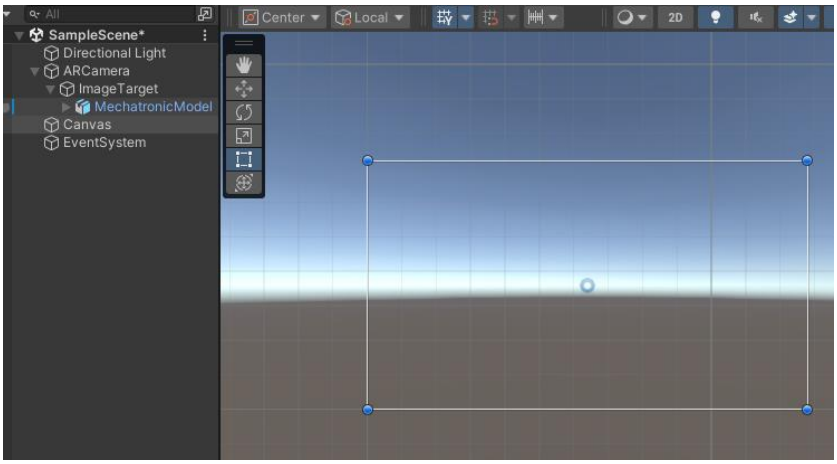
Insertar un Canvas en la escena UI -> Canvas (Figura 72), el cual contendrá toda la información que requerimos mostrar dentro de la aplicación.

Figura 72: Insertar Canvas a escena



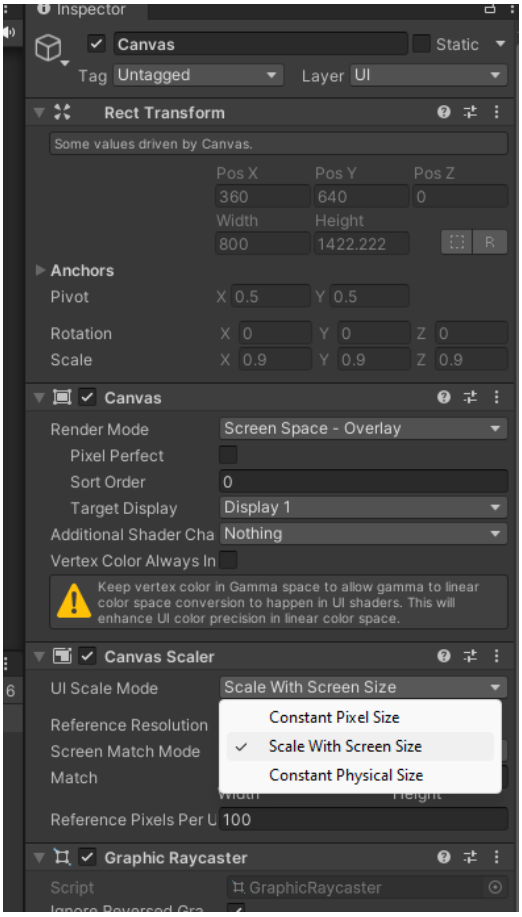
Configurar la orientación del Canvas para que pueda mostrar los elementos dispuestos en formato de pantalla horizontal (1280x720 Landscape) como se muestra en la Figura 73.

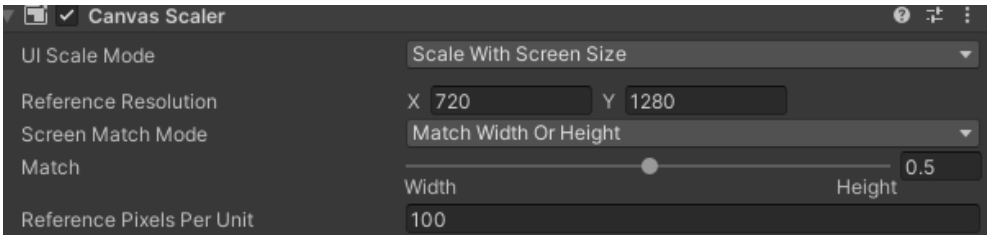
Figura 73: Orientación de Canvas



Configurar el inspector del Canvas Canvas Scaler -> Scale With Screen Size, para que el contenido se ajuste al tamaño de la ventana y asignar una resolución de 1280x720 y configurar el parámetro Match en 0.5 (Figura 74).

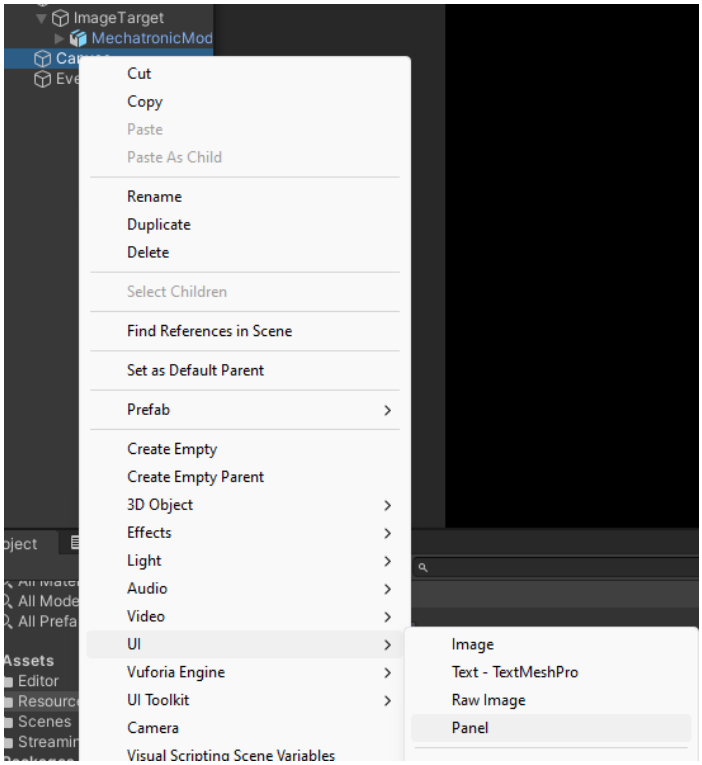
Figura 74: Configuración de Canvas





Ahora corresponde insertar un objeto UI tipo Panel, que permitirá colocar los elementos visibles de la interfaz, en la escena ubicar UI -> Panel (Figura 75).

Figura 75: Insertar un UI Panel



En la sección Inspector del Panel, asignar un color de fondo (Color) y cambiar la sección “Source Image” como “None (Sprite)” (Figura 74).

Ahora se cargará la imagen de fondo de inicio de la aplicación en la carpeta de recursos del proyecto, ubicar la imagen seleccionada como fondo principal, luego se debe asignar en las propiedades de Inspector: “Texture Type” como “Sprite (2D and UI)” para que sea factible insertarlo en el Panel posteriormente (Figura 77). La imagen debe contener un ícono de reproducción.

Figura 76: Configurar UI Panel

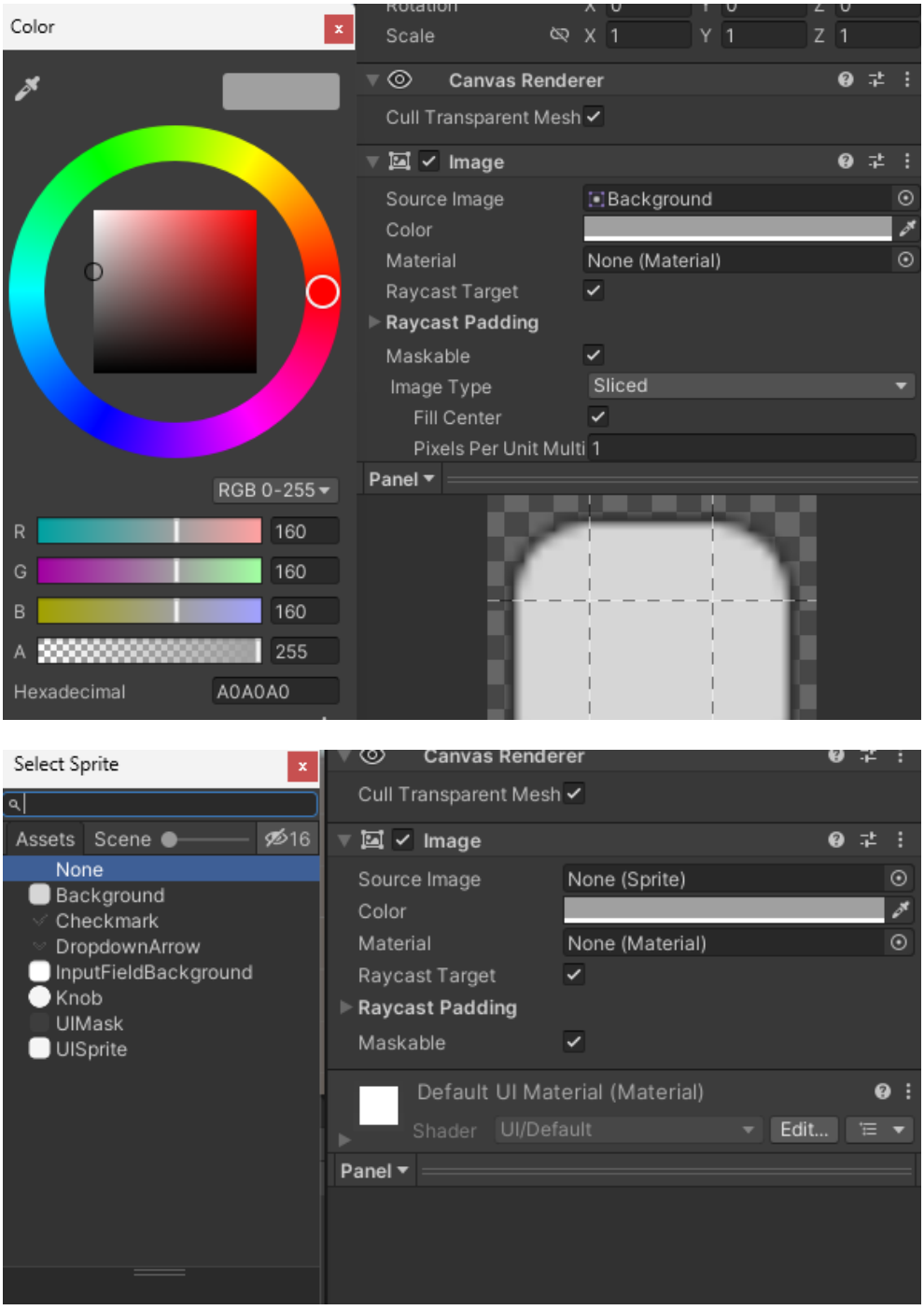
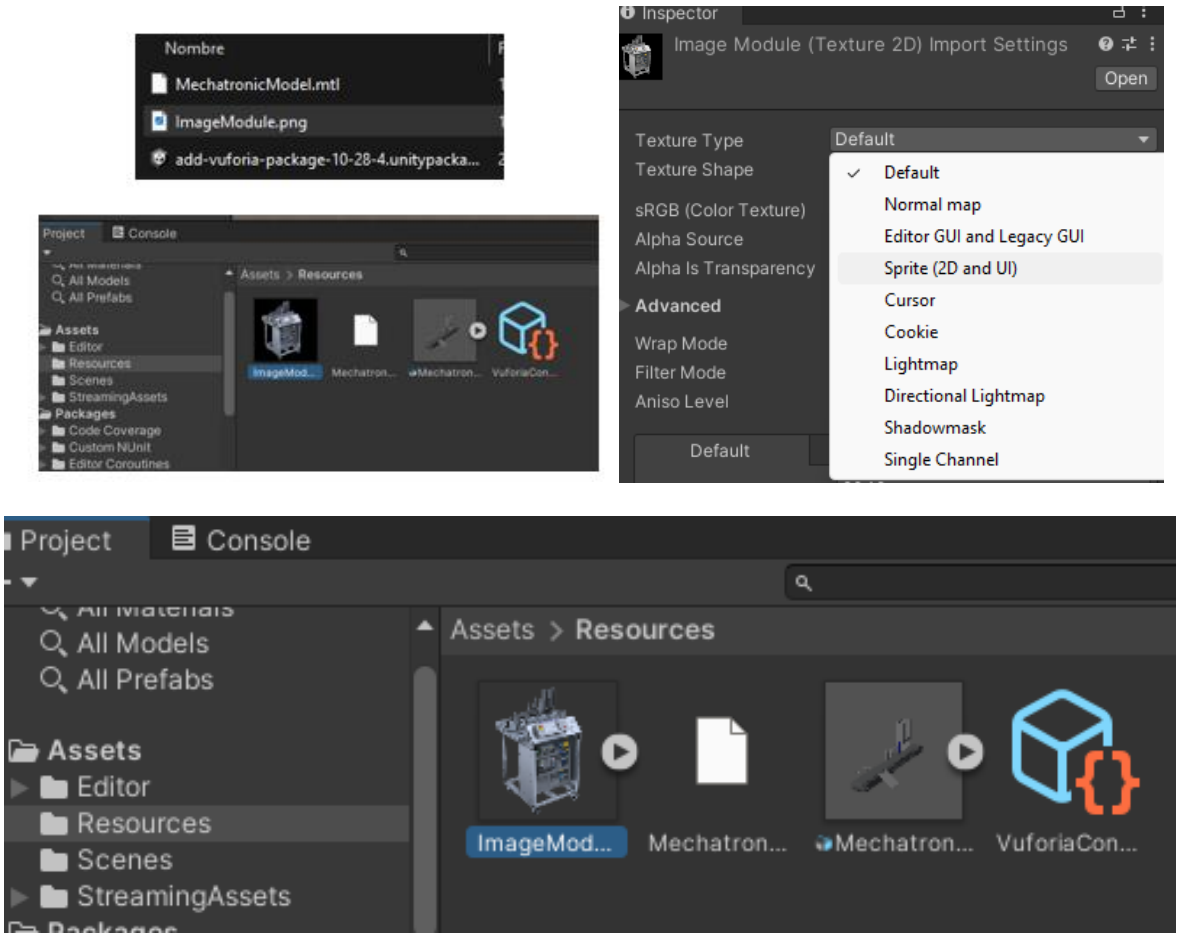


Figura 77: Cargar imagen de fondo en sección Recursos



Insertar a la escena un objeto UI -> Image para que contenga la imagen preparada en el paso anterior. (Figura 78). En la sección Inspector, asignar en el campo “Source Image” la imagen de la carpeta Recursos, ésta aparece ahora en la escena.

Insertar a la escena un objeto UI -> Text - TextMeshPro para colocar el texto que representa el título de la aplicación. (Figura 79). En la sección Inspector, asignar en el campo “Text Input” el título que se requiera colocar, ésta aparece ahora en la escena.

Figura 78: Insertar objeto UI Imagen

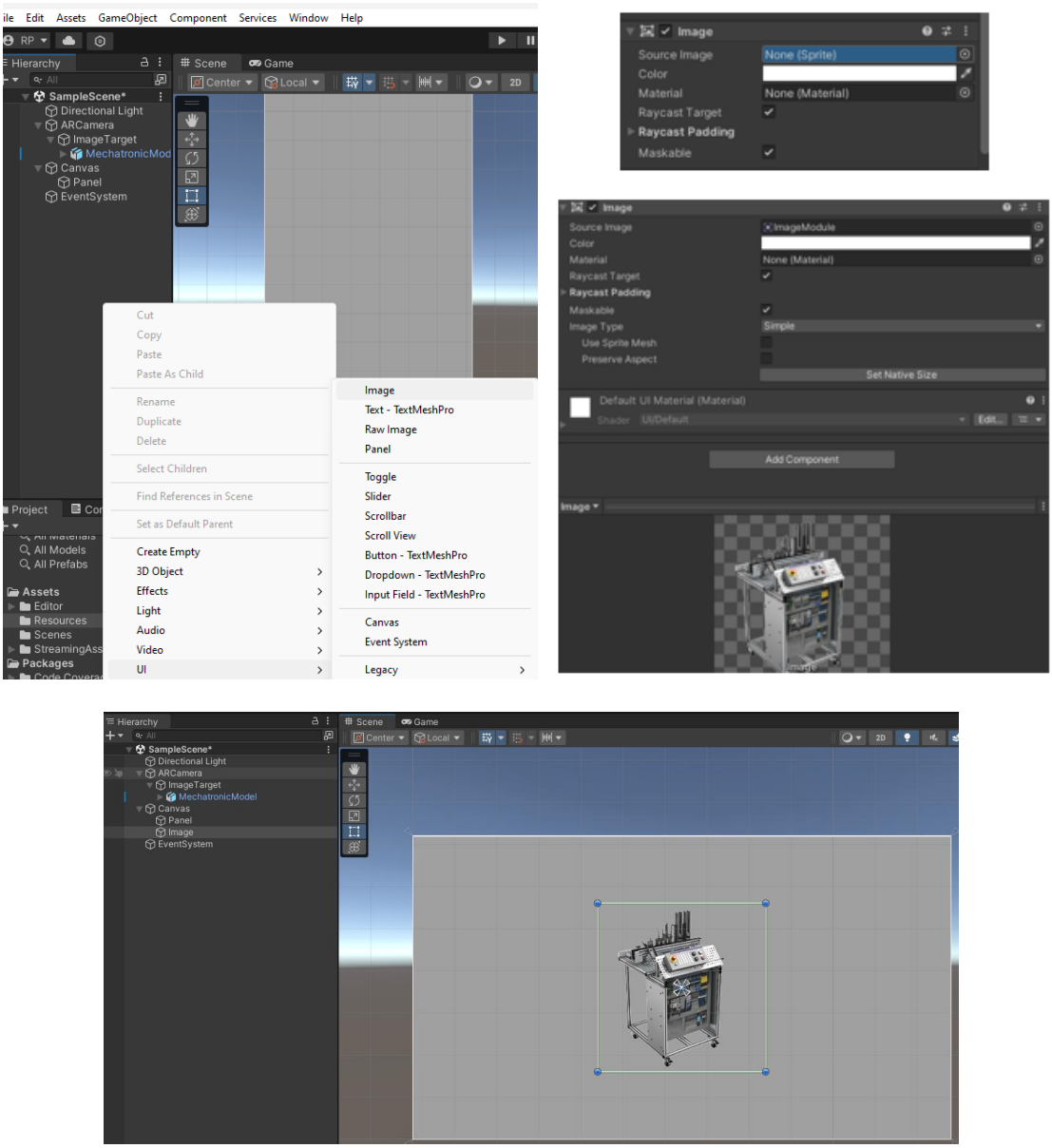
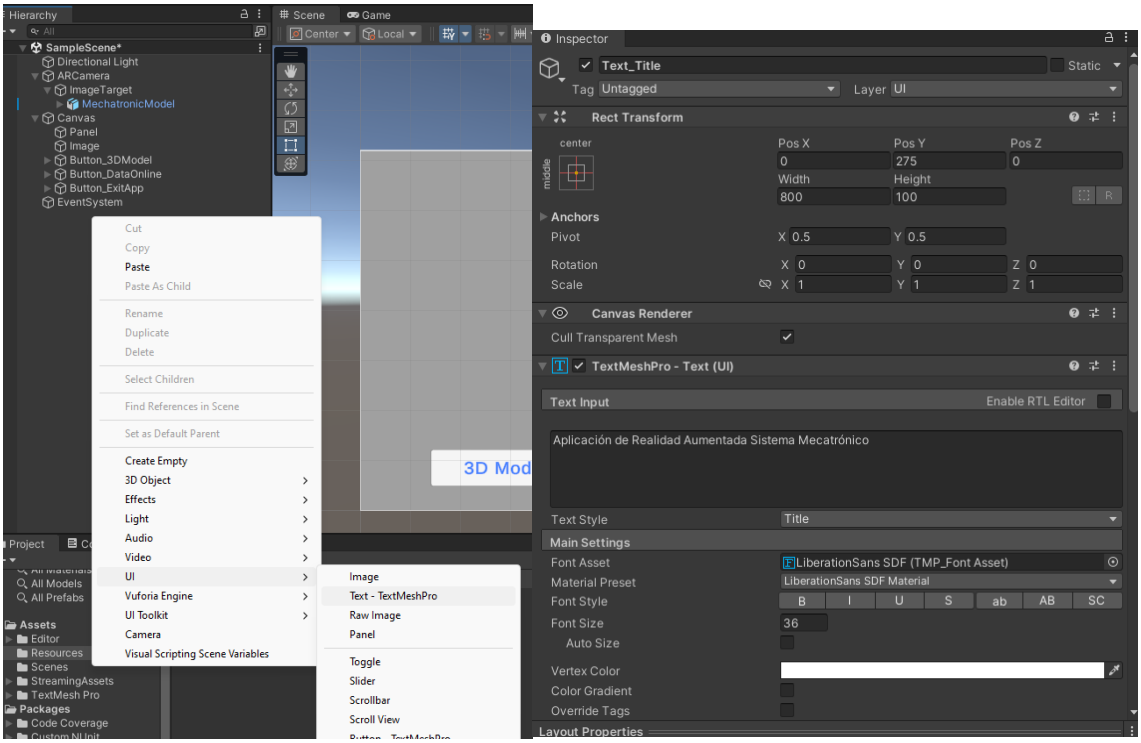


Figura 79: Insertar Texto UI TextMeshPro



Insertar un objeto vacío que contendrá el conjunto de textos que mostrará la data del equipo proveniente de la comunicación en el Canvas -> Create Empty (Figura 80). Asignar el nombre “Data Panel” en la sección Inspector del objeto, dejar el objeto desactivado por defecto.

Insertar más objetos UI -> Text - TextMeshPro para colocar los textos que representan la información proveniente de la comunicación con el equipo. (Figura 81). Estos textos deben quedar bajo la jerarquía del “Data_Panel” creado en el paso anterior. Los textos con caracteres “…” serán elementos que entrarán para la comunicación de la aplicación con el Bróker MQTT.

Figura 80: Insertar Empty Object – Data Panel

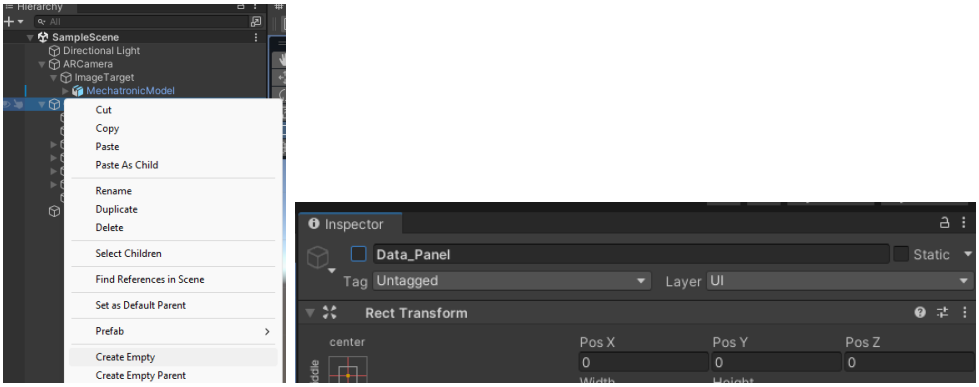
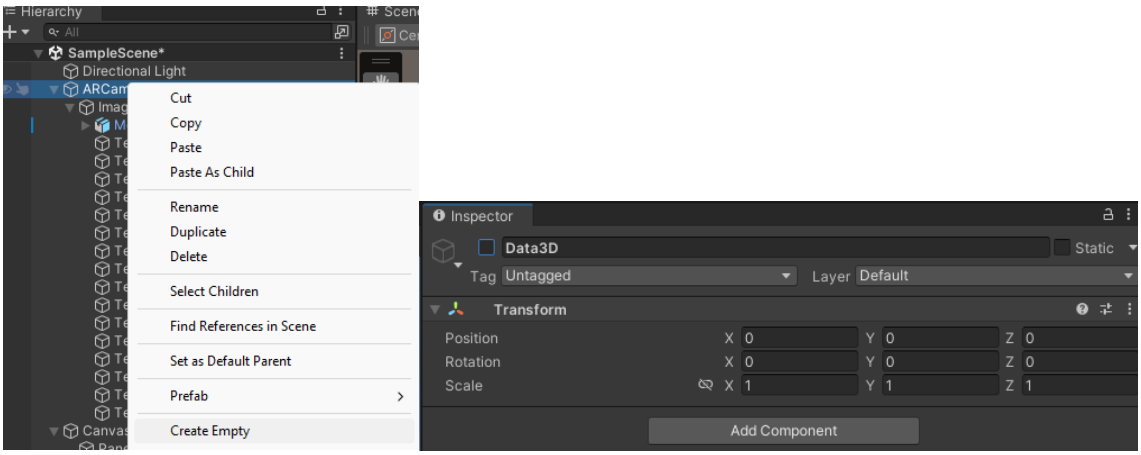


Figura 81: Insertar Texto UI TextMeshPro

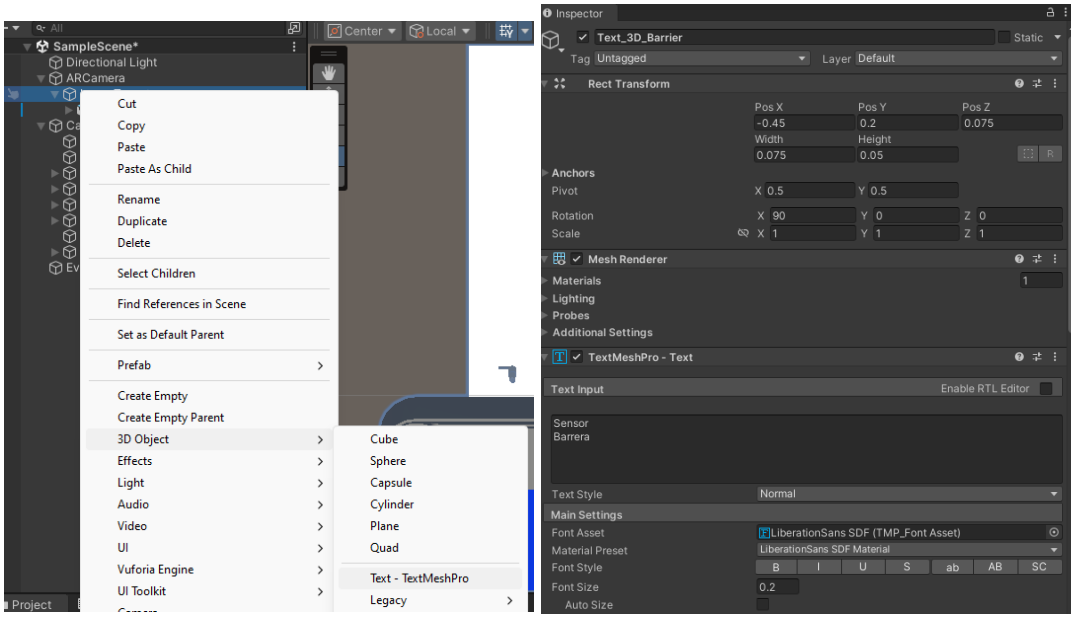
Insertar un objeto vacío que contendrá el conjunto de textos 3D que mostrará la data del equipo proveniente de la comunicación, insertar en la escena Create Empty (Figura 82). Asignar el nombre “Data 3D” en la sección Inspector del objeto, dejar el objeto desactivado por defecto.

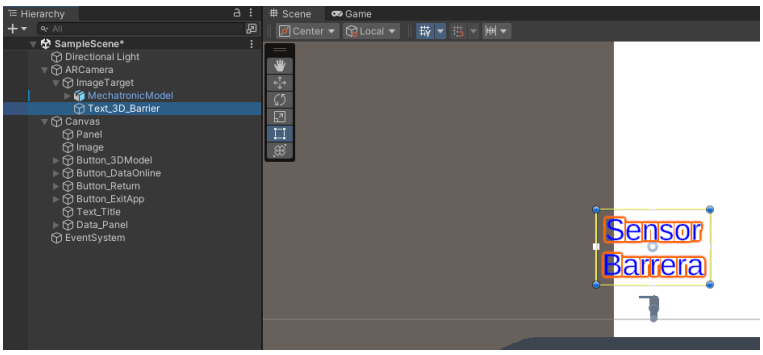
Figura 82: Insertar Empty Object – Data 3D - parte 1



Insertar un objeto 3D object -> Text - TextMeshPro para colocar los textos que representan la información proveniente de la comunicación con el equipo. (Figura 83).

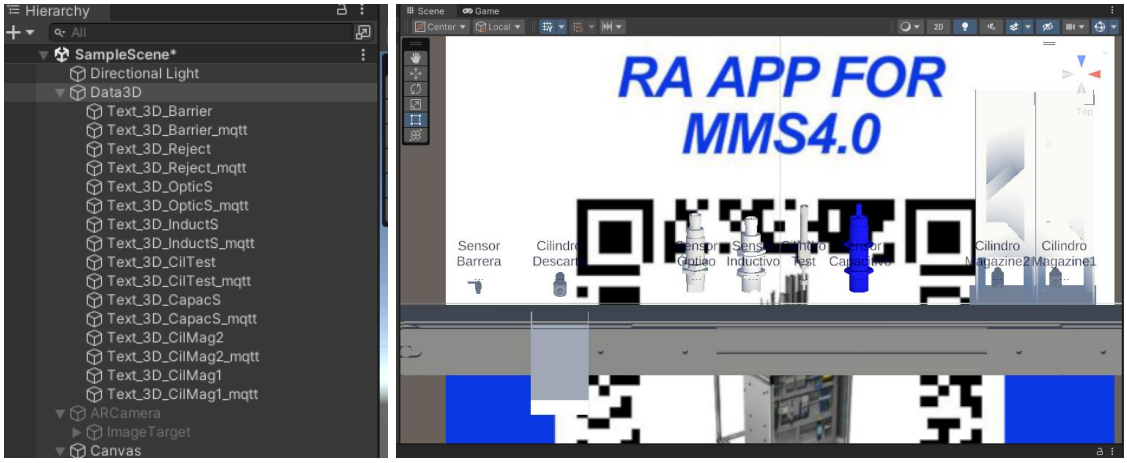
Figura 83: Insertar Object 3D TextMeshPro





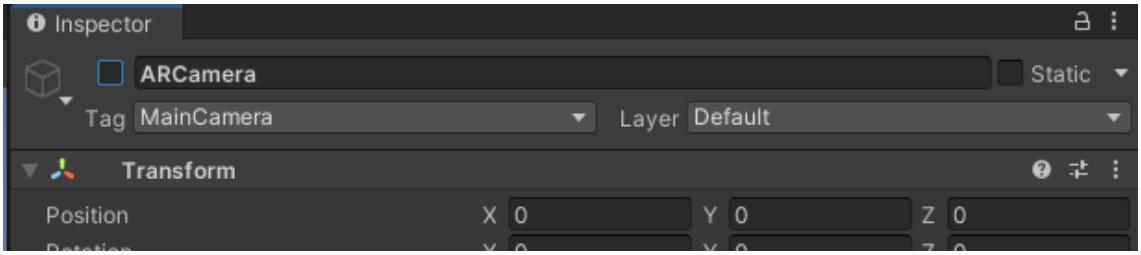
Terminar de insertar demás objetos 3D TextMeshPro (Figura 84). Los textos con caracteres "…" serán elementos que entrarán para la comunicación de la aplicación con el Bróker MQTT.

Figura 84: Insertar Empty Object – Data 3D - parte 2



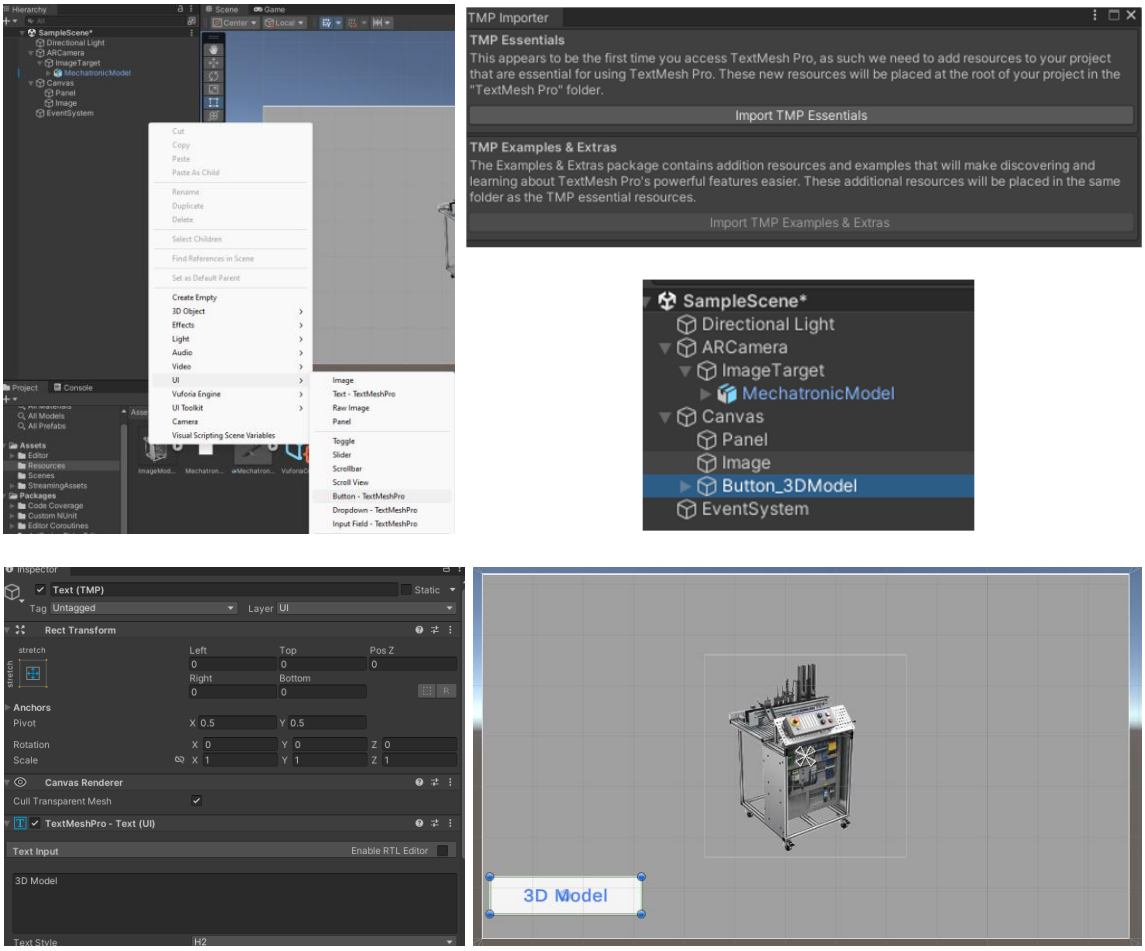
Seleccionar el objeto ARCamera (Figura 85), desactivar en el inspector la visibilidad por defecto.

Figura 85: Insertar Empty Object – Data 3D - parte 3



Ahora se insertarán los botones que permitirán navegar con la información que se preparó, en la escena insertar UI -> Button - TextMeshPro. Confirmar la importación de TMP Essentials (Figura 86). Asignarle el nombre al objeto y el texto que representa el botón

Figura 86: Insertar Button TextMeshPro



Insertar 3 botones más (Data Online, Exit App, Return) que permitirán navegar dentro de la aplicación, la ubicación de los botones Return y Exit App deben coincidir, además el botón Return debe estar desactivada la visibilidad (Figura 87).

En la ventana inspector de cada botón (Figura 88) ubicar la sección “On Click()” agregar 10 elementos a la lista, los cuales cada uno debe corresponder a lo mostrado en la Tabla 6.

Figura 87: Insertar botones para completar el menú de navegación - parte 1

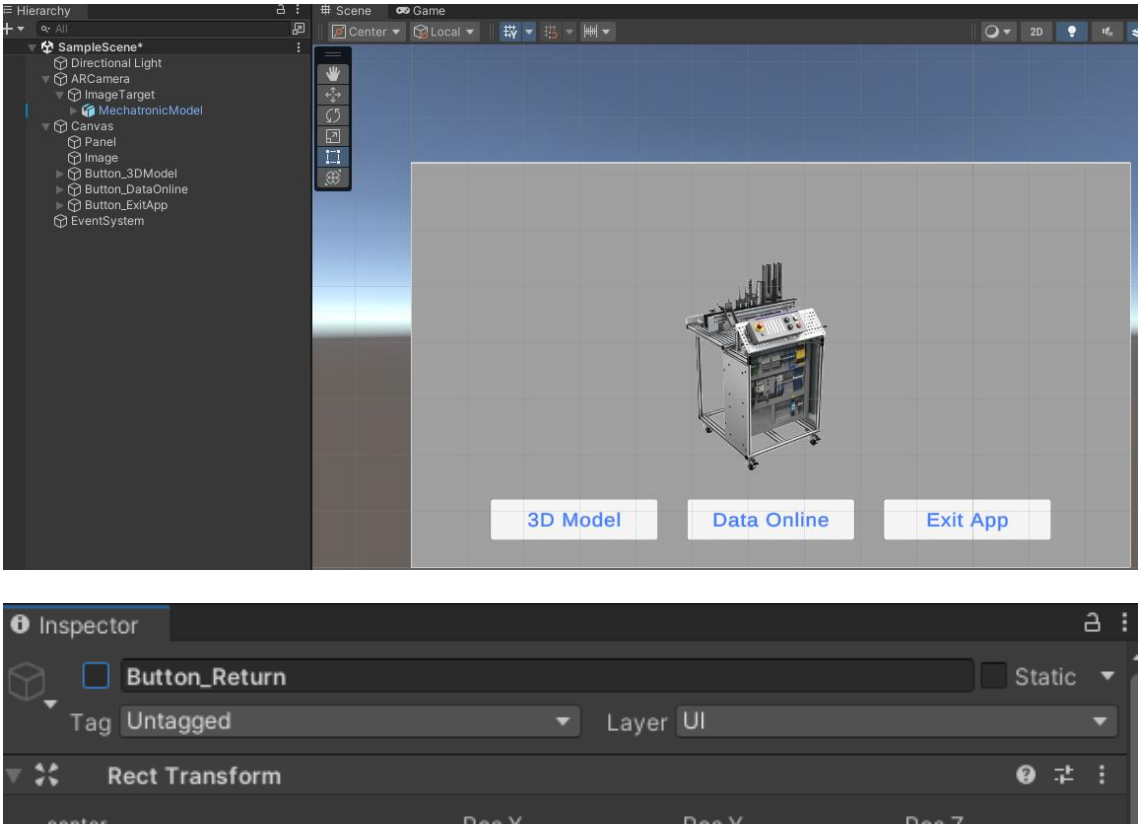
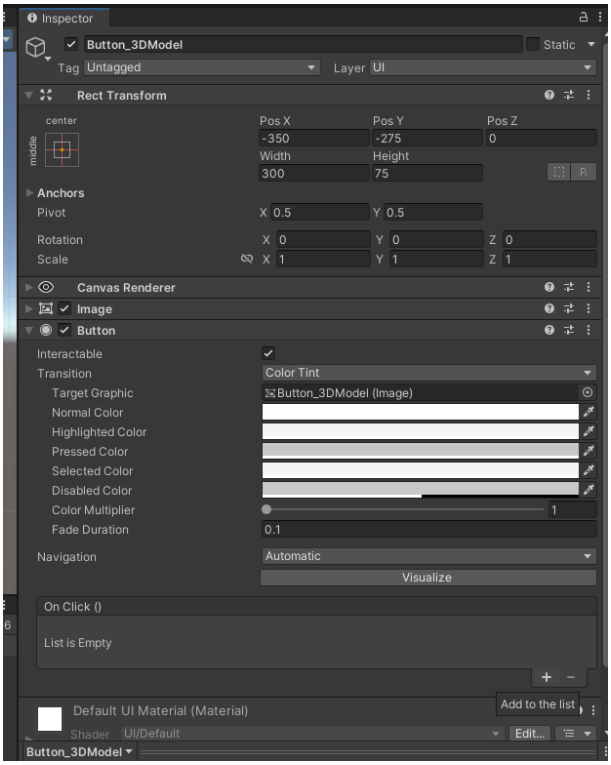
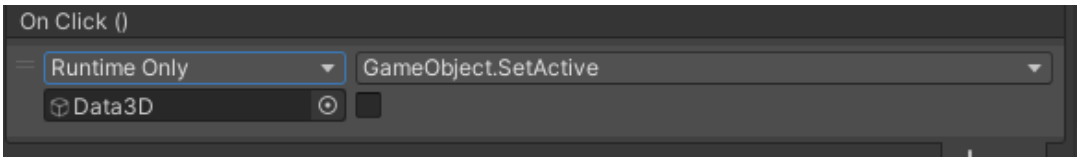


Tabla 6: Lista de tópicos asociados a las variables del equipo mecatrónico – Parte 2

Ítem	Object
1	Data 3D
2	AR Camera
3	Panel
4	Image
5	Button 3D Model
6	Button Data Online
7	Button Return
8	Button Exit App
9	Text Title
10	Data Panel

Figura 88: Insertar botones para completar el menú de navegación - parte 2





Como acción cada elemento debe considerarse “GameObject.SetActive” y la visibilidad estará en función a lo mostrado para cada caso (Figura 89, Figura 90, Figura 91).

Figura 89: Configuración de visibilidad de objetos para Botón 3D Model

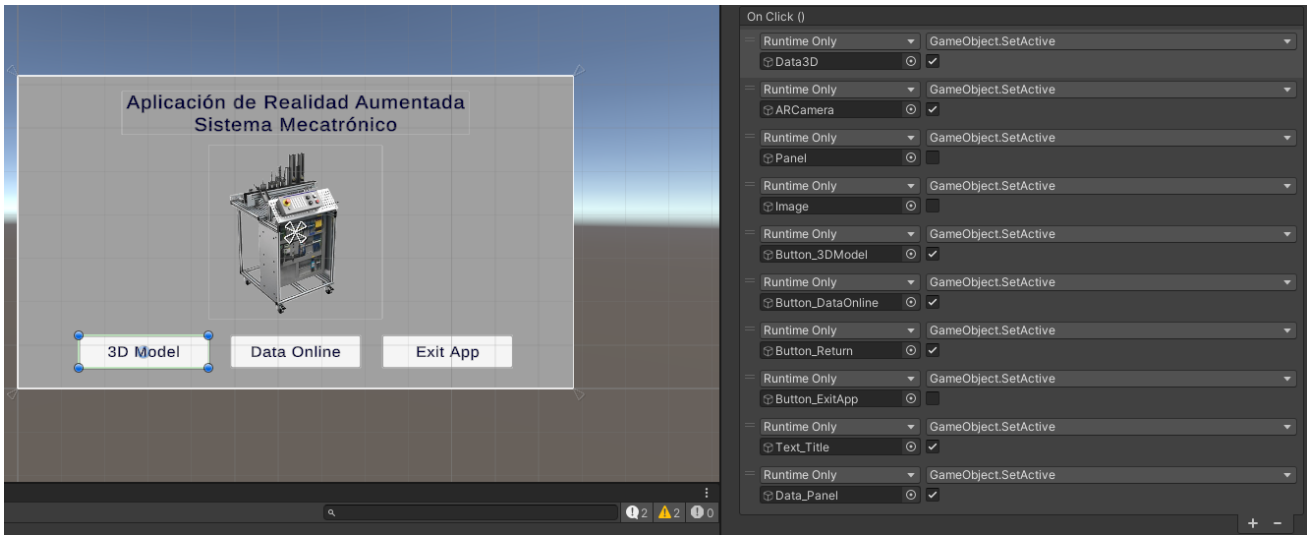


Figura 90: Configuración de visibilidad de objetos para Botón Data Online

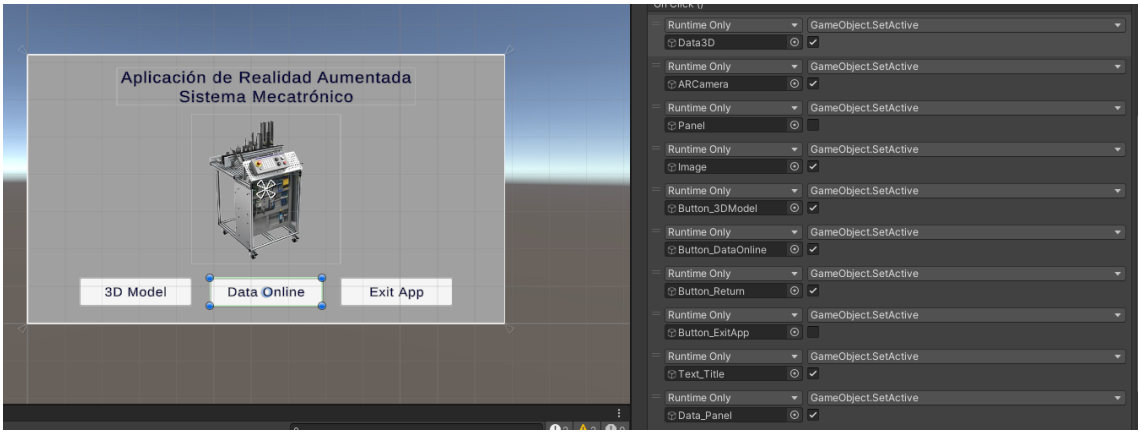
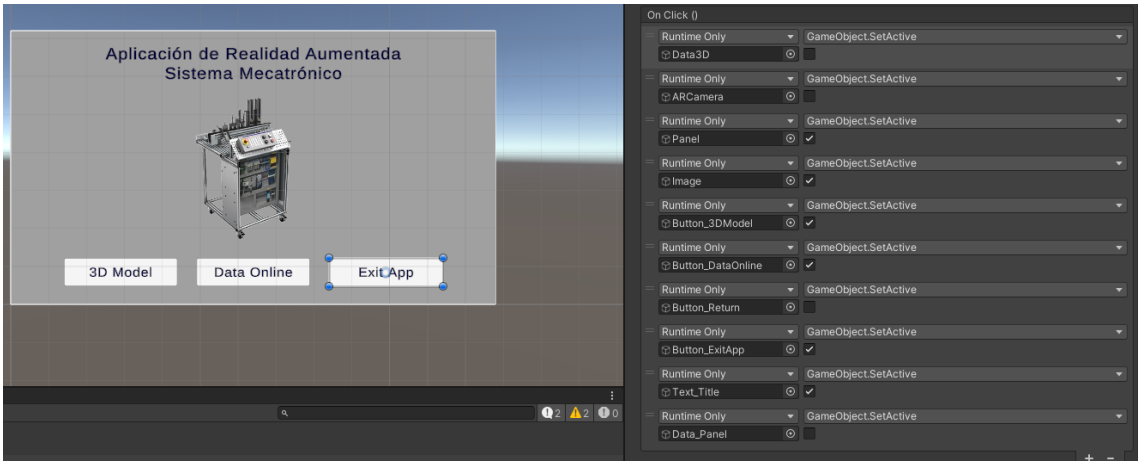


Figura 91: Configuración de visibilidad de objetos para Botón Return



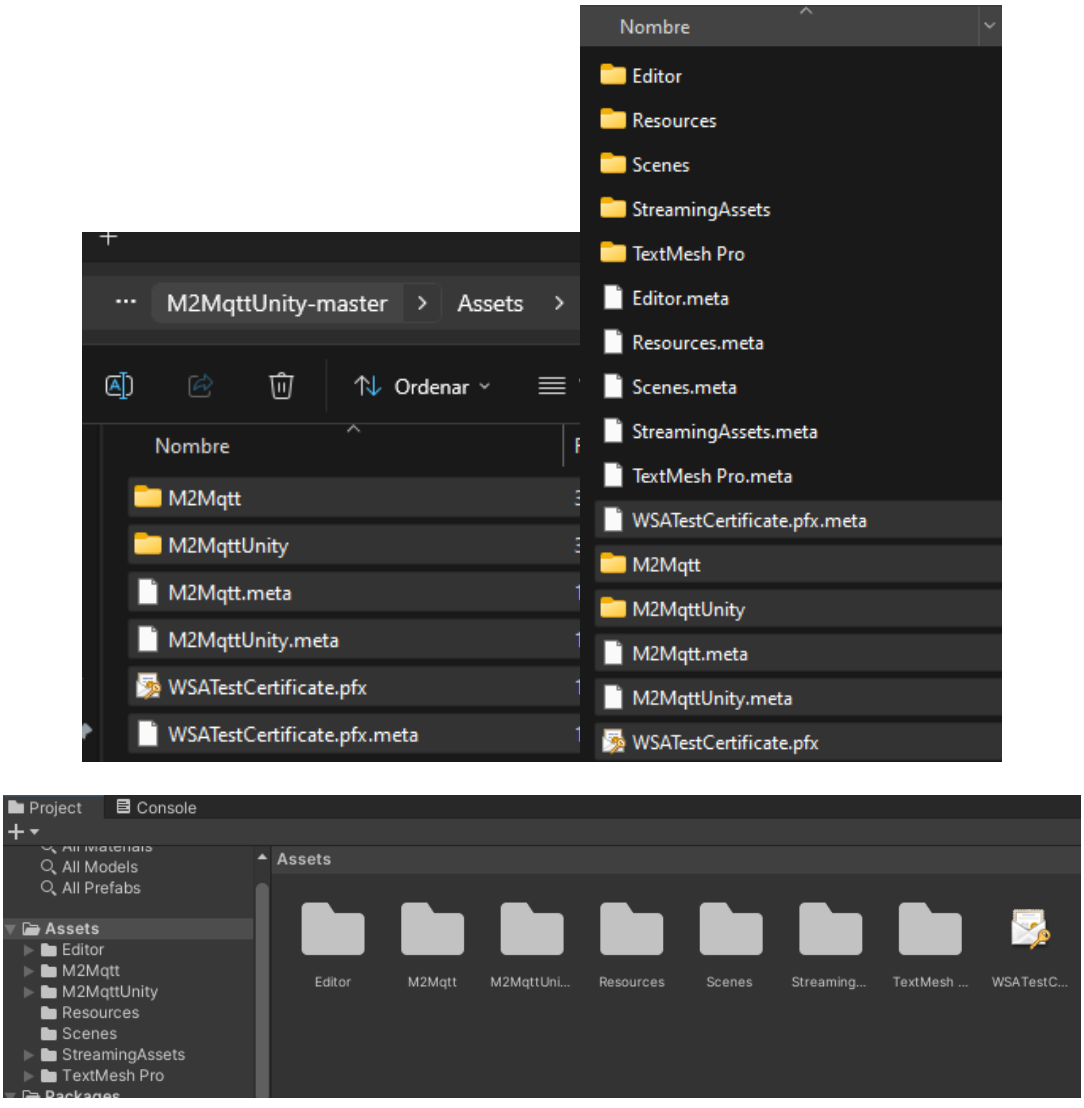
Hasta este punto se tiene la interfaz finalizada, se requiere ahora configurar los elementos de comunicación para lograr mostrar en la aplicación los datos del equipo mecatrónico publicados en el Bróker MQTT como se verificó en la Figura 31.

3.7. Comunicar aplicación de RA con Equipo Industrial a través de MQTT

Para realizar la comunicación a través del protocolo MQTT, se hace uso de la librería M2Mqtt for Unity del MIT License, con el cual contamos con estructuras ya definidas a utilizar en la comunicación. Se comienza importando la librería al proyecto, luego se crean los scripts relacionados a la recepción (mqttReceiver) y luego a la conexión con las variables requeridas (mqttController), finalmente se crea el script para el botón Salir App, y se asignan las propiedades de scripts a cada objeto que requiera actualizar la información proveniente de la comunicación MQTT.

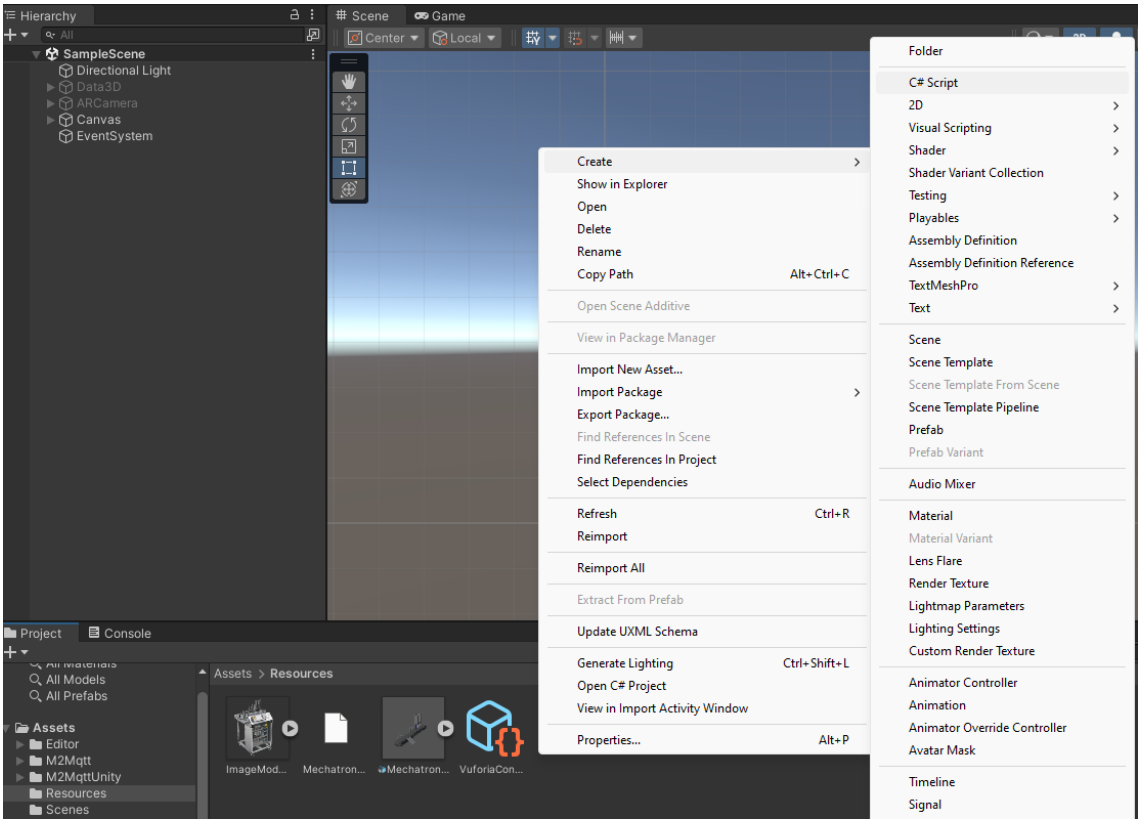
a. ***Importación de librería y creación de scripts***: copiar los elementos de la librería M2Mqtt for Unity a la carpeta Assets del proyecto (Figura 92).

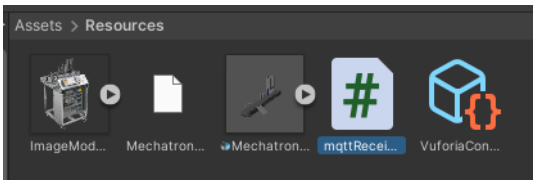
Figura 92: Cargar librería M2Mqtt for Unity a carpeta Recursos del proyecto



En la carpeta Recursos crear 4 scripts: “mqttReceiver”, ”mqttController”, ”mqttController2” y “Salir” (Figura 93)

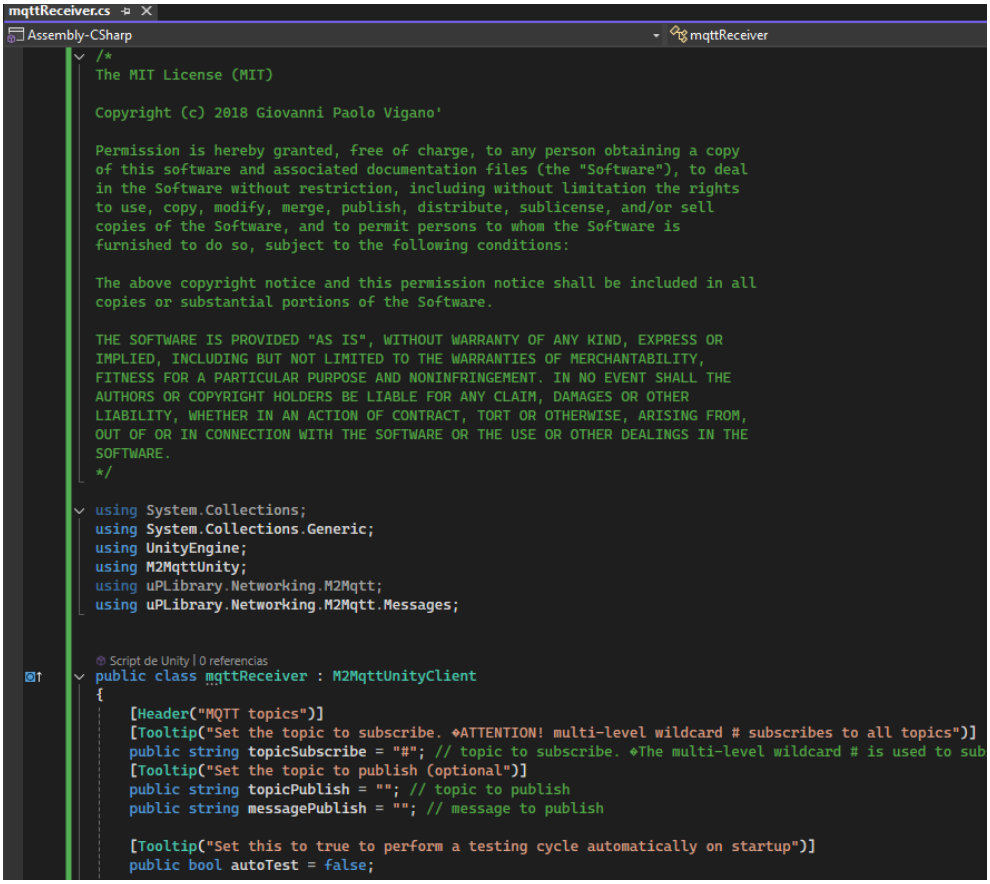
Figura 93: Crear scripts





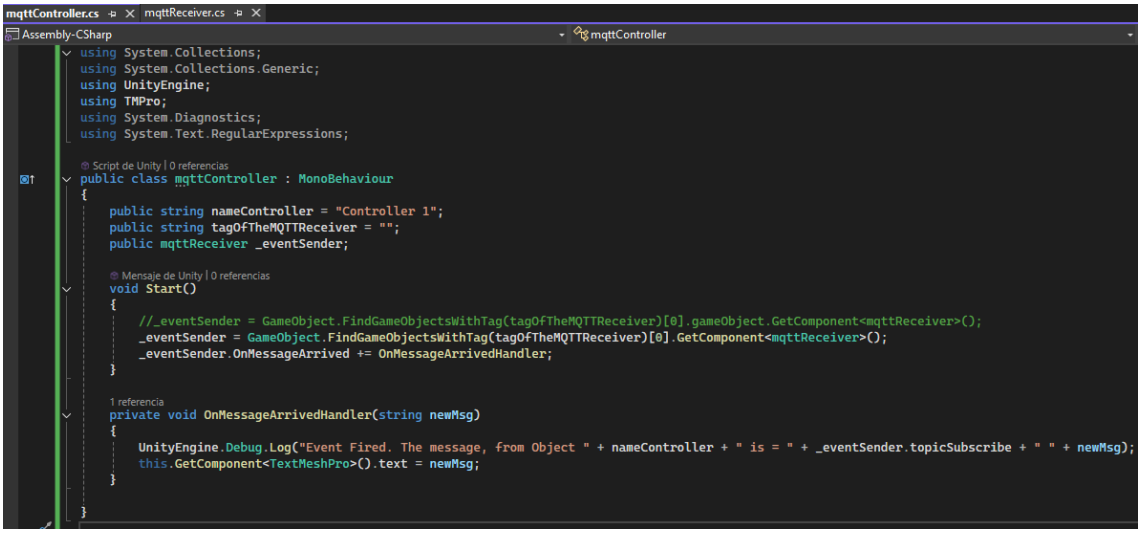
Cargar el código del Anexo I: “mqttReceiver código de programa” al archivo mqttReceiver, que permite la creación de conexión con el Bróker MQTT, en función del Tópico a subscribir (Figura 94).

Figura 94: Cargar código programa mqttReceiver



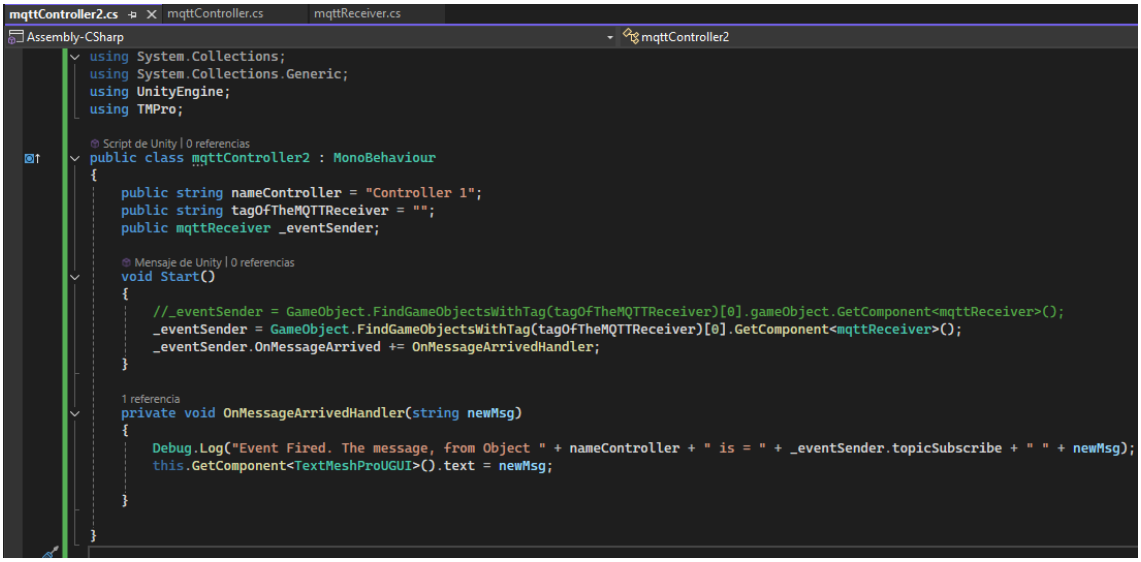
Cargar el código del Anexo II: “mqttController código de programa” al archivo mqttController, que permite el registro de la lectura del mqttReceiver respectivo hacia el objeto de texto 3D que esté relacionado mediante el tag propio (Figura 95).

Figura 95: Cargar código programa Controller



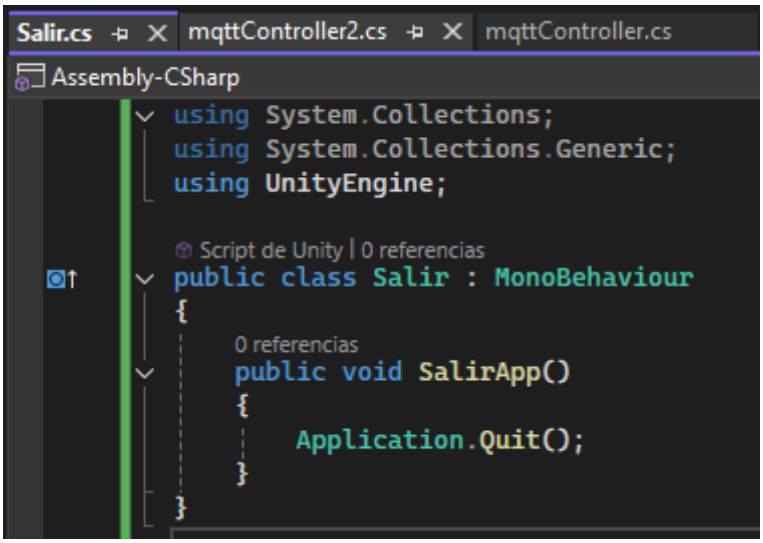
Cargar el código del Anexo III: “mqttController2 código de programa” al archivo mqttController2, que permite el registro de la lectura del mqttReceiver respectivo hacia el objeto de texto UI que esté relacionado mediante el tag propio (Figura 96).

Figura 96: Cargar código programa Controller2



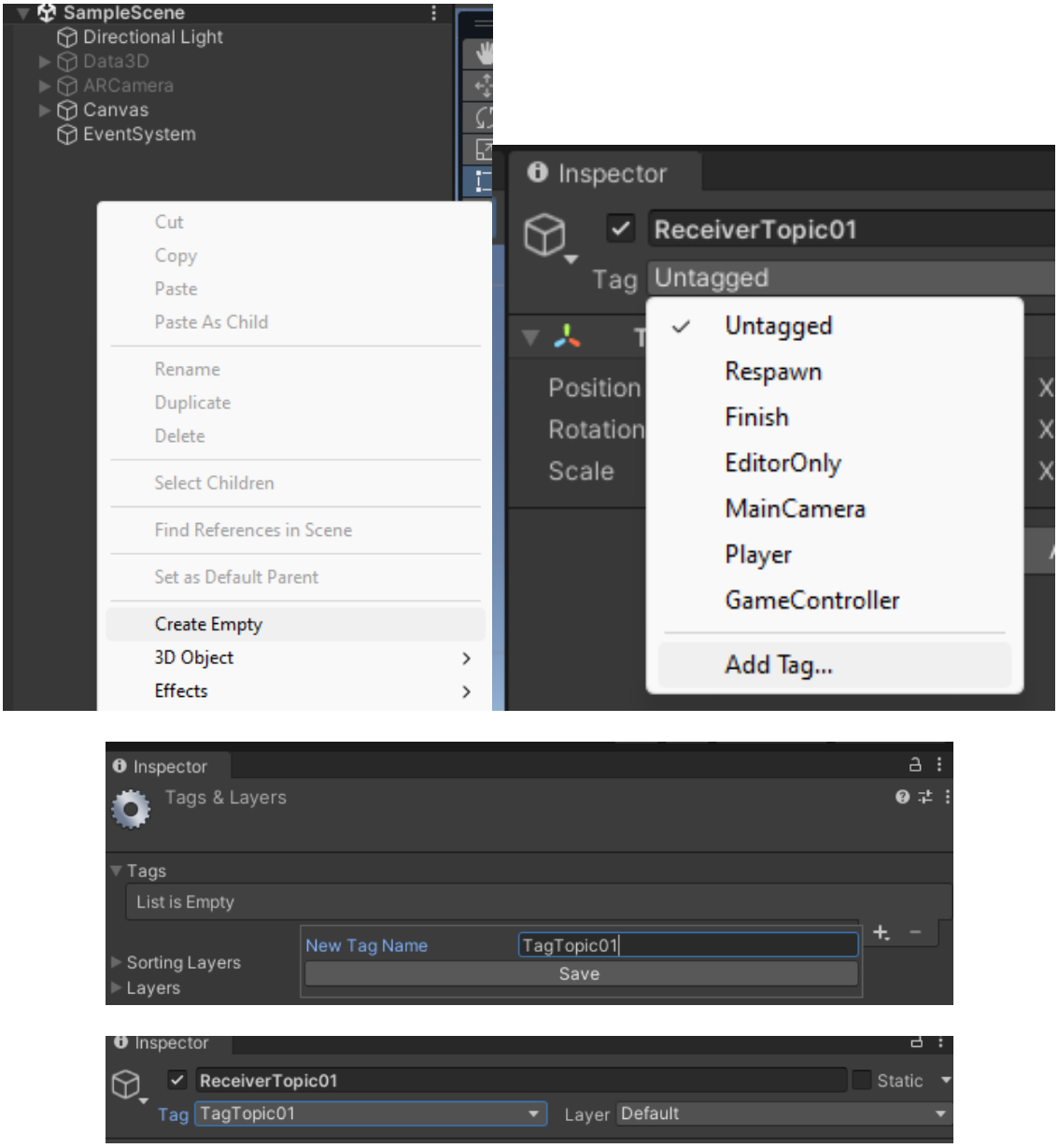
Cargar el código del Anexo IV: “Salir código de programa” al archivo “Salir”, que permite el cerrar la aplicación desde el botón que llame la función “SalirApp()” (Figura 97).

Figura 97: Cargar código programa Salir



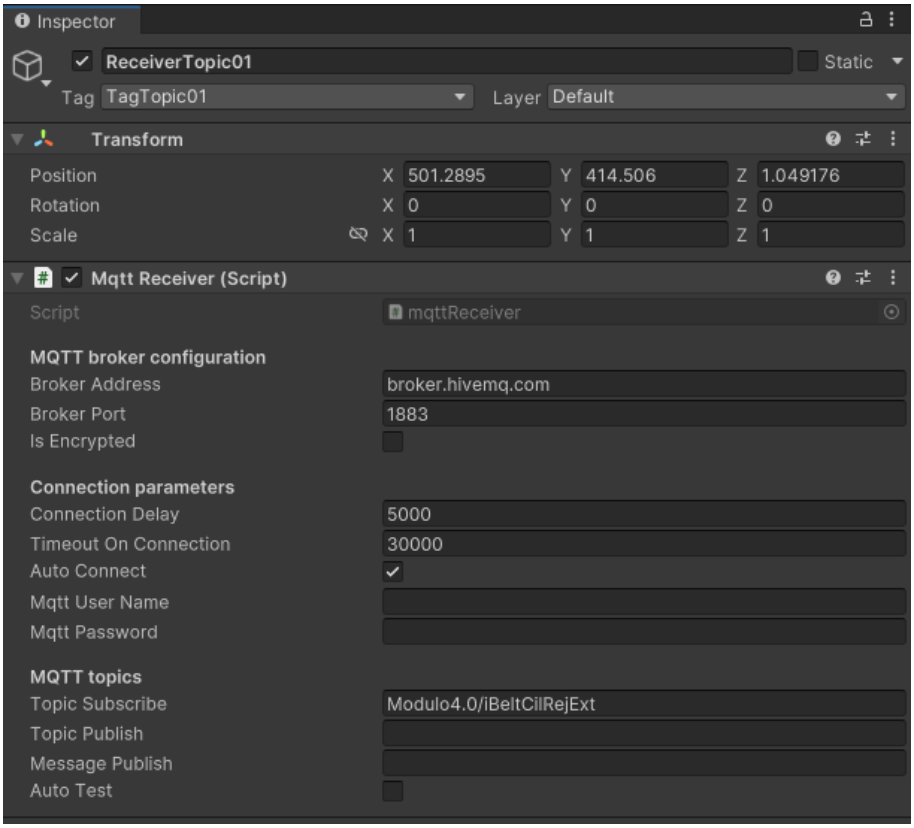
b. ***Creación de objetos, asignación de propiedades***: Realizar la creación de un objeto vacío, asignar de nombre “ReceiverTopic01” asignarle un tag “TagTopic01” según lo mostrado en la Figura 98.

Figura 98: Cargar código programa Salir



Desde la carpeta Recursos, arrastrar y soltar al objeto ReceiverTopic01, el srcipt “mqttReceiver”, en la sección Inspector aparece los campos de parámetros a configurar para la conexión MQTT (Figura 99).

Figura 99: Cargar parámetros para conexión MQTT



Insertar 14 objetos vacíos adicionales (Figura 100) y configurarlos de forma similar que en la (Figura 99), asignar los tópicos según la Tabla 7.

Figura 100: Crear más objetos Receiver

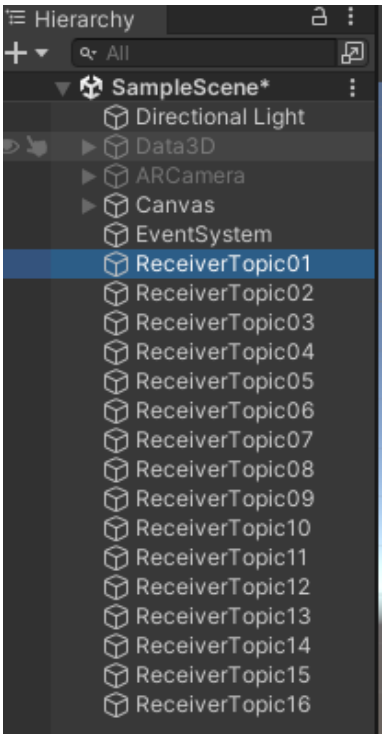


Tabla 7: Lista de tópicos asociados a las variables del equipo mecatrónico – Parte 3

Ítem	Variable	Tópico asociado	Object Asociado
1	Sensor de descarte material	Modulo4.0/iBeltCilRejExt	Text_3D_Reject_mqtt
2	Sensor de barrera de fin de banda	Modulo4.0/iBeltSensorBarr	Text_3D_Barrier_mqtt
3	Sensor capacitivo	Modulo4.0/iBeltSensorCap	Text_3D_CapacS_mqtt
4	Sensor inductivo	Modulo4.0/iBeltSensorInd	Text_3D_InductS_mqtt
5	Sensor óptico	Modulo4.0/iBeltSensorOpt	Text_3D_OpticS_mqtt
6	Sensor testeo	Modulo4.0/qBeltCilTestExt	Text_3D_CilTest_mqtt
7	Sensor despacho magazine 1	Modulo4.0/iMag1CilExt	Text_3D_CilMag1_mqtt
8	Sensor despacho magazine 2	Modulo4.0/iMag2CilExt	Text_3D_CilMag2_mqtt
9	Parada de emergencia	Modulo4.0/iEmrgSTOP	Text_EStop_mqtt
10	Lote de producción	Modulo4.0/cCONT_Lote	Text_Batch_mqtt
11	Producción actual	Modulo4.0/iNROProd	Text_ActualProd_mqtt
12	Disposición de producto	Modulo4.0/iDisposicion	Text_Disp_mqtt
13	Material de producto	Modulo4.0/iMaterial	Text_Material_mqtt
14	Confirmación de base solicitada	Modulo4.0/iPedidoBASE_OK	Text_Base_mqtt
15	Confirmación de tapa solicitada	Modulo4.0/iPedidoTAPA_OK	Text_Tapa_mqtt
16	Otro caso	Modulo4.0/Hola	

Ahora se enlaza al objeto con el Receiver Topic Texto respectivo, arrastrar el script “mqttController” para los Textos 3D y “mqttController2” para los Textos UI, relacionar con el TagTopic (Figura 101) respectivo según la Tabla 7.

Al botón ExitApp agregar un evento On Click, que relacione el script “Salir” asignar el objeto “Button_ExitApp” y seleccionar la función Salir.SalirApp, con lo que configura el cerrar la aplicación al pulsar este botón (Figura 102).

Figura 101: Asignar script Controller a cada objeto de texto

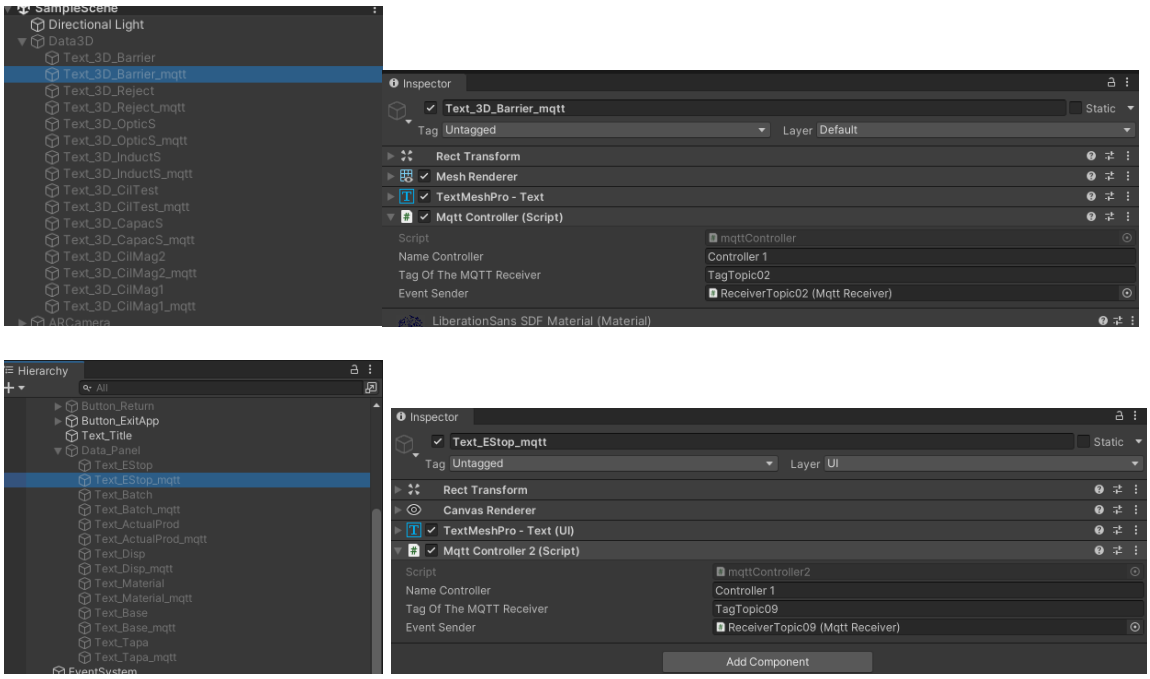
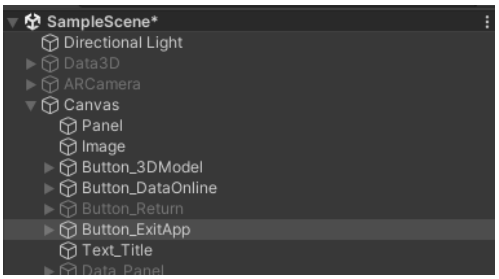
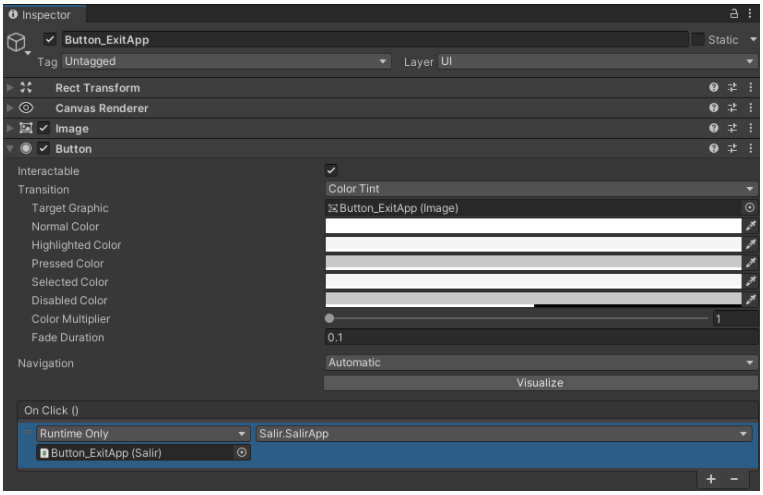


Figura 102: Asignar Script a Boton “Exit App”



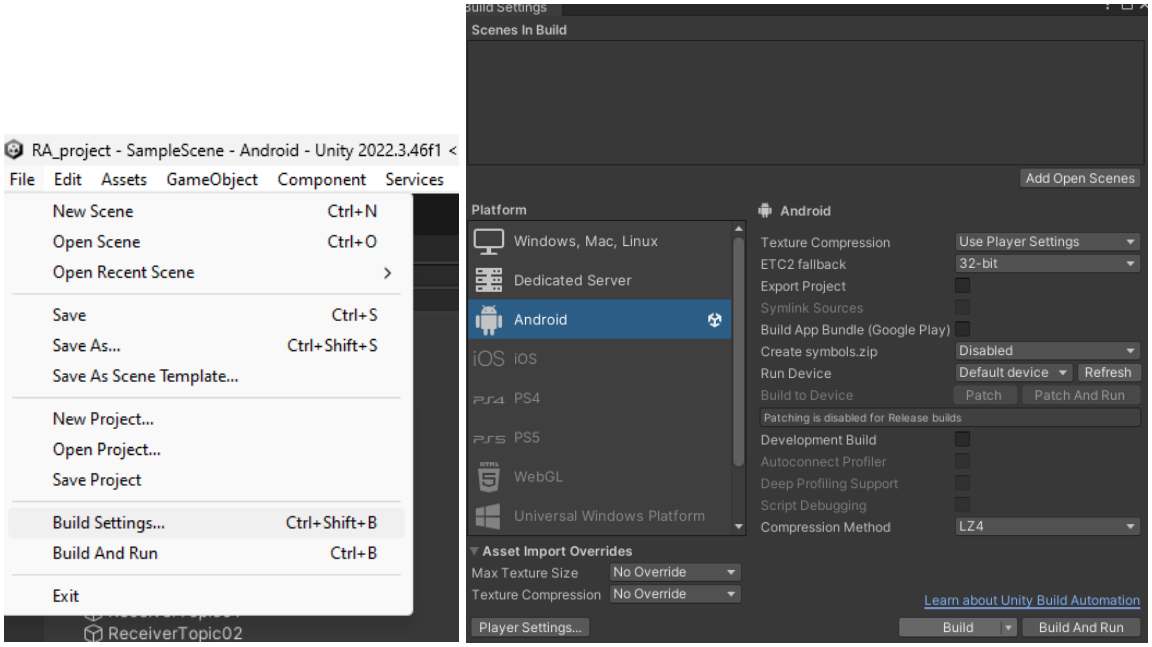


Ahora se tiene la aplicación terminada, lista para generar el archivo .apk con el cual podrá instalarse en un dispositivo móvil.

3.8. Generar APK e instalar en dispositivo móvil

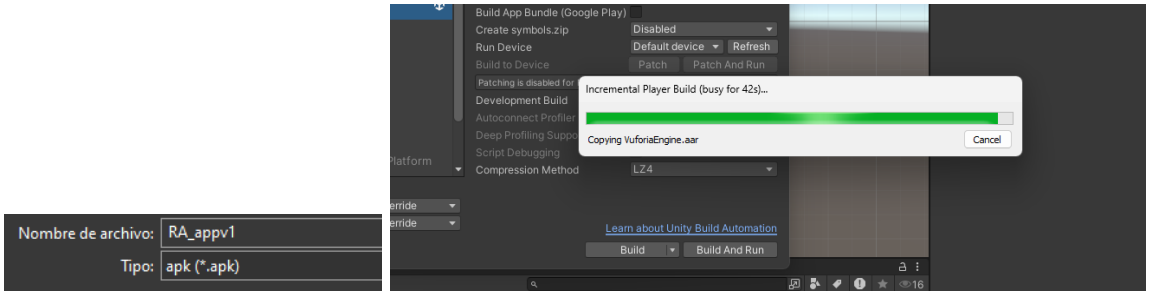
Para la generación de la apk con el cuál se podrá instalar en un dispositivo móvil, para ello primero asegurarse de guardar cualquier cambio tal como se mostró en la Figura 71, luego dirigirse a la sección “File -> Build Settings…”, en la interfaz, darle al botón “Build”.

Figura 103: Generar .apk del proyecto



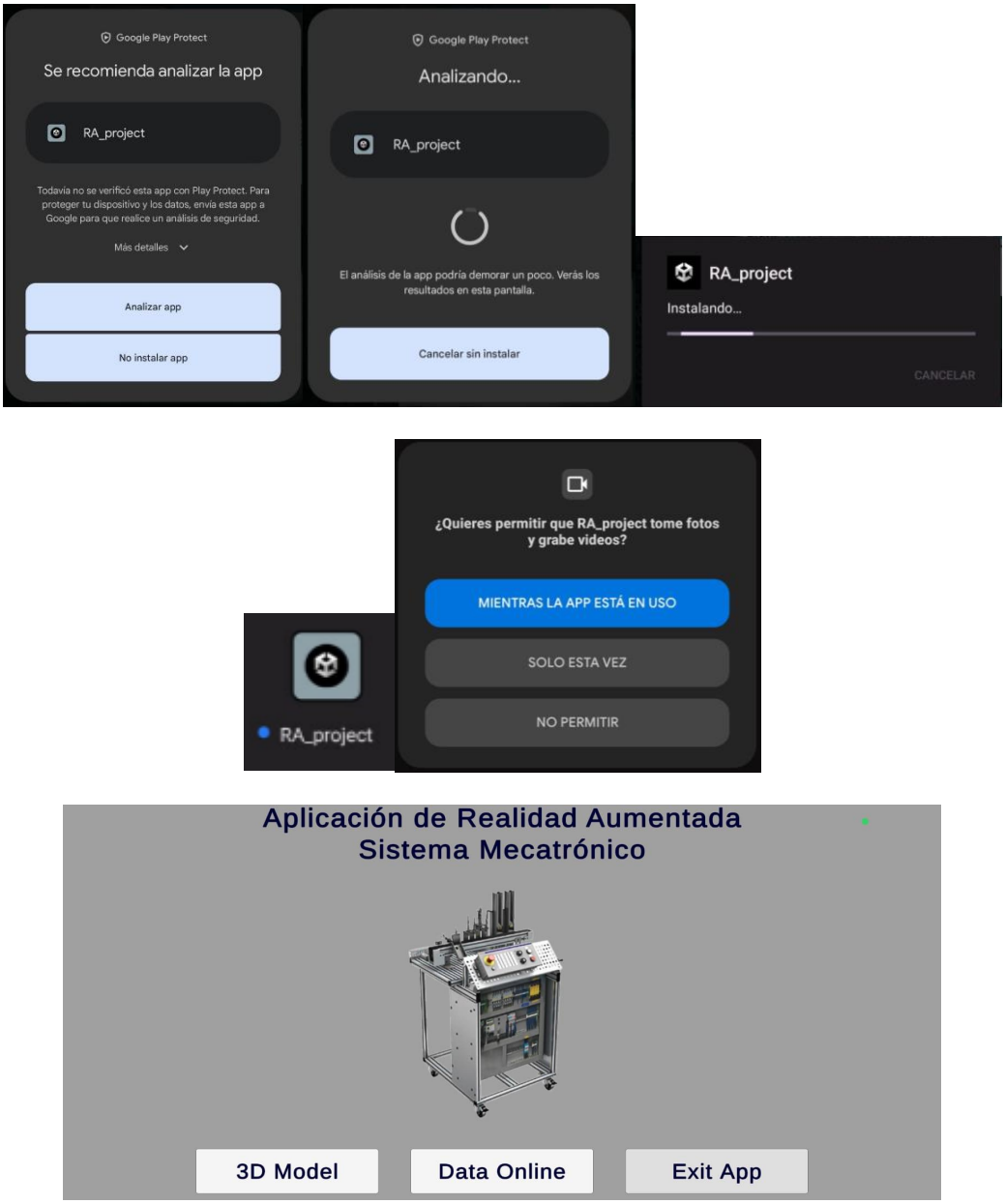
Asignar un nombre a la aplicación a generar, y luego de la carga el archivo será creado (Figura 104).

Figura 104: Creación de .apk



Pasar la aplicación al dispositivo móvil y realizar la instalación de esta, dar permisos de análisis y uso de la cámara principal con lo que la aplicación queda instalada.

Figura 105: Instalación de .apk en dispositivo móvil



Con estos pasos ya se cuenta con la aplicación instalada y lista para ser utilizada en el dispositivo móvil.

CAPÍTULO IV. Resultados

4.1. Análisis de resultados

La aplicación de realidad aumentada muestra al iniciar, la pantalla inicial, con los botones de navegación “3D Model”, “Data Online”, “Exit App” (Figura 106).

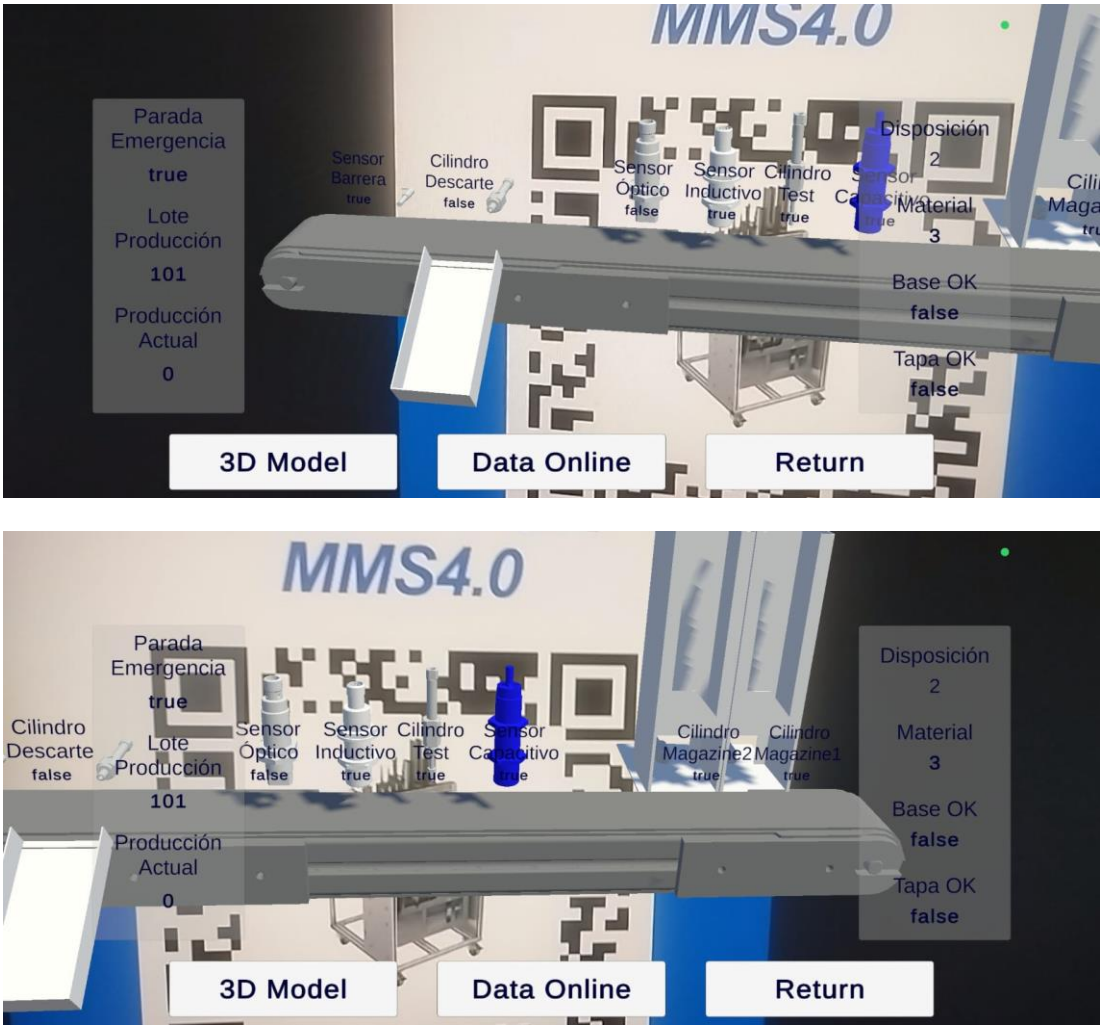
Figura 106: Ingreso a aplicación móvil

El botón “3D Model” permite ingresar a la sección de escaneo del Image Target creado, para proyectar el modelo en 3D elaborado, se debe proyectar sobre la imagen creada como Image Target, con la cámara encendida del dispositivo móvil proyectará sobre la imagen el objeto 3D desarrollado (Figura 107).

Figura 107: Sección “3D Model”

El botón “Data Online” permite ingresar a la sección visualización de la data preparada a mostrar proveniente de la comunicación MQTT desde el equipo mecatrónico. Asimismo, sigue proyectando el objeto 3D del sistema, se logra verificar el estado de las señales, conformen se dan los cambios en el equipo mecatrónico (Figura 108).

Figura 108: Sección “Data Online”



El botón “Return” permite regresar a la ventana de presentación de la aplicación, y el botón “Exit App” permite salir y cerrar la aplicación (Figura 109).

Figura 109: Sección “Exit App”

La presente aplicación la logran desarrollar los estudiantes con el apoyo de los softwares descritos, esta introduce a los mismos en la aplicación directa de tecnologías de virtualización de información con uso de software de licencia libre, lo que permite una réplica sencilla y de fácil entendimiento por los estudiantes.

Conclusiones

Se ha desarrollado el procedimiento para llegar desde tener un equipo mecatrónico implementado, a una aplicación de realidad aumentada conectada a través de internet con el equipo mecatrónico aplicando protocolos de comunicación estándar

Se propuso un diseño de desarrollo de la aplicación de Realidad Aumentada, mostrada en la Figura 16, en la cual se busca comunicar el equipo mecatrónico con un servidor-PC a través de protocolo de comunicación OPC-UA y luego compartir la información de este a internet a través del protocolo de comunicación MQTT, para finalmente crear la aplicación de realidad aumentada y a través de la comunicación MQTT poder acceder a los datos compartidos del equipo mecatrónico.

Se consiguió realizar la comunicación del equipo mecatrónico a internet a través del protocolo MQTT con apoyo de la plataforma NODE-RED Figura 31), los datos se vienen publicando en los tópicos del bróker MQTT mostrados en la Tabla 4

Se elaboró una aplicación móvil de realidad aumentada con el apoyo de la plataforma Unity, así como también del servicio Vuforia Engine, con lo cual se verifica que los tanto el objeto 3D (Figura 107), como la información en línea del equipo, permiten el acceso a los datos del equipamiento mecatrónico (Figura 108).

Se logra la comunicación de la aplicación de Realidad Aumentada con equipo mecatrónico a través del protocolo MQTT, asignando la actualización de los datos (Figura 108) a textos especificados en Tabla 7, cerrando así los objetivos planteados para el presente proyecto.

Recomendaciones

Para la comunicación de un equipo industrial que no contenga el protocolo de comunicación OPC-UA, se debe considerar la aplicación de interfaces tipo IoT Gateway, que permitan realizar la transferencia correcta de información, incluso hacia un protocolo MQTT, con lo que permite centralizar el procesamiento realizado en la primera parte de este trabajo, dentro de un dispositivo.

Para una aplicación industrial, el protocolo OPC-UA debe desarrollarse con un nivel de seguridad suficiente de tal forma que evite la intrusión de agentes desconocidos en la información que provienen de los equipos. De manera similar considerar la seguridad en la comunicación del protocolo MQTT, el cual también permite asignar niveles de seguridad a fin de evitar intrusiones en el traslado a través de internet de los datos.

Respecto al desarrollo de la aplicación móvil, considerar las versiones de Android donde se desea que corra la aplicación. Asimismo, no es recomendable utilizar la última versión de Unity para el desarrollo de la aplicación, debido a que el soporte de posibles incompatibilidades no será eficiente, mientras que, con versiones más antiguas, el soporte y resolución de fallos ya han sido levantados.

Las funciones desarrolladas en los scripts en C# de la aplicación de realidad aumentada, deberá optimizarse con la finalidad de poder asignar más información, lo que facilita la carga de la aplicación en dispositivos móviles de prestaciones limitadas.

Referencias

Aranda, E. y Gómez E. (2022) Aplicación móvil con realidad aumentada para mejorar el proceso de capacitación en el área de producción de la empresa OPPFILM S.A. [Tesis de Titulación, Universidad Autónoma del Perú]. https://repositorio.autonoma.edu.pe/handle/20.500.13067/1954

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787-2805.

Azuma, R. (1997). A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments. 6(4), 355-385. https://doi.org/10.1162/pres.1997.6.4.355

Berman, B. (2012). 3-D printing: The new industrial revolution. Business Horizons, 55(2), 155-162. https://doi.org/10.1016/j.bushor.2011.11.003

Berryman, D. (2012). Realidad aumentada: una revisión. Medical Reference Services Quarterly. 31 (2), 212-218. https://doi.org/10.1080/02763869.2012.670604

Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. Computers in Industry, 101, 1-12.

Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. Computers in Industry, 101, 1-12. https://doi.org/10.1016/j.compind.2018.04.015

Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. Computers in Industry, 101, 1-12. https://doi.org/10.1016/j.compind.2018.04.015

Chicaiza, A. y Palacios B. (2024) Realidad aumentada para el monitoreo del estado operativo de diferentes dispositivos comúnmente usados en el MPS [Trabajo de titulación, Universidad Politécnica Salesiana]. http://dspace.ups.edu.ec/handle/123456789/27973

Chui, M., Manyika, J., & Miremadi, M. (2016). Where machines could replace humans—and where they can't (yet). McKinsey Quarterly, 1-9.

Cruz, J. (2023) Implementación de un sistema SCADA en el ámbito de la Industria 4.0 e IOT [Trabajo de titulación, Universidad Técnica de Ambato]. https://repositorio.uta.edu.ec/handle/123456789/39216

Gracia, R. (2020) Aplicación de la realidad aumentada para el mantenimiento de equipos industriales [Trabajo de Fin de Grado, Universidad Pontificia Comillas]. https://repositorio.comillas.edu/jspui/handle/11531/41255

Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE) (pp. 791-798). IEEE.

Kagermann, H., Wahlster, W., & Helbig, J. (2013). Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Final report of the Industrie 4.0 Working Group. acatech.

Kimura Y., Manabe S., Ikeda S., Kimura A. and Shibata F. (2019) Can Transparent Virtual Objects Be Represented Realistically on OST-HMD's?. IEEE Conference on Virtual Reality and 3D User Interfaces (VR) https://doi.org/10.1109/VR.2019.8798001

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. Business & Information Systems Engineering, 6(4), 239-242

Lee, J., Kao, H. A., & Yang, S. (2014). Service innovation and smart analytics for Industry 4.0 and big data environment. Procedia CIRP, 16, 3-8. https://doi.org/10.1016/j.procir.2014.02.001

Mahmoud, R., Yousuf, T., Aloul, F., & Zualkernan, I. (2020). Internet of Things (IoT) security: Current status, challenges, and countermeasures. International Journal of Distributed Sensor Networks, 16(4), 1-17.

Mineraud, J., Mazhelis, O., Su, X., & Tarkoma, S. (2016). A gap analysis of Internet-of-Things platforms. Computer Communications, 89-90, 5-16.

Muñoz-Saavedra L., Miró-Amarante L., Domínguez-Morales M. (2020) Augmented and Virtual Reality Evolution and Future Tendency. Applied Sciences 2020, 10(1), 322; https://doi.org/10.3390/app10010322

Pallavicini, F., Pepe, A., & Minissi, M. E. (2019). Gaming in augmented reality: A study of the effects of Pokémon Go on embodied cognition. International Journal of Human–Computer Interaction, 35(11), 976–987.

Pereira, J., Dias, P., & Santos, B. S. (2020). An evaluation of Vuforia Target image detection capabilities. International Journal of Online and Biomedical Engineering (iJOE), 16(1), 4–13

Schmalstieg D., Höllerer T. (2016). Augmented reality: Principles and practice. Addison-Wesley Professional. https://ptgmedia.pearsoncmg.com/images/9780321883575/samplepages/9780321883575.pdf

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. IEEE Transactions on Industrial Informatics, 15(4), 2405-2415. https://doi.org/10.1109/TII.2018.2873186

Wang, X., Ong, S. K., & Nee, A. Y. C. (2016). A comprehensive survey of augmented reality assembly research. Advances in Manufacturing, 4(1), 1-22. https://doi.org/10.1007/s40436-015-0115-7

Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: State of the art and future trends. International Journal of Production Research, 56(8), 2941-2962. https://doi.org/10.1080/00207543.2018.1444806

Anexos

ANEXO 1: mqttReceiver código de programa

```
/*
The MIT License (MIT)

Copyright (c) 2018 Giovanni Paolo Vigano'

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using M2MqttUnity;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;


public class mqttReceiver : M2MqttUnityClient
{
    [Header("MQTT topics")]
    [Tooltip("Set the topic to subscribe. ?ATTENTION! multi-level wildcard # subscribes to all topics")]
    public string topicSubscribe = "#"; // topic to subscribe. ?The multi-level wildcard # is used to
subscribe to all the topics. Attention i if #, subscribe to all topics. Attention if MQTT is on data plan
    [Tooltip("Set the topic to publish (optional")]
    public string topicPublish = ""; // topic to publish
    public string messagePublish = ""; // message to publish

    [Tooltip("Set this to true to perform a testing cycle automatically on startup")]
    public bool autoTest = false;

    // using C# Property GET/SET and event listener to reduce Update overhead in the controlled objects
    private string m_msg;
    public string msg
    {
        get
        {
            return m_msg;
        }
```

```
        set
        {
            if (m_msg == value) return;
            m_msg = value;
            if (OnMessageArrived != null)
            {
                OnMessageArrived(m_msg);
            }
        }
    }

    public event OnMessageArrivedDelegate OnMessageArrived;
    public delegate void OnMessageArrivedDelegate(string newMsg);

    // using C# Property GET/SET and event listener to expose the connection status
    private bool m_isConnected;

    public bool isConnected
    {
        get
        {
            return m_isConnected;
        }
        set
        {
            if (m_isConnected == value) return;
            m_isConnected = value;
            if (OnConnectionSucceeded != null)
            {
                OnConnectionSucceeded(isConnected);
            }
        }
    }
    public event OnConnectionSucceededDelegate OnConnectionSucceeded;
    public delegate void OnConnectionSucceededDelegate(bool isConnected);

    // a list to store the messages
    private List<string> eventMessages = new List<string>();

    public void Publish()
    {
        client.Publish(topicPublish, System.Text.Encoding.UTF8.GetBytes(messagePublish),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
        Debug.Log("Test message published");
    }
    public void SetEncrypted(bool isEncrypted)
    {
        this.isEncrypted = isEncrypted;
    }
    protected override void OnConnecting()
    {
        base.OnConnecting();
    }
    protected override void OnConnected()
    {
```

```
        base.OnConnected();
        isConnected = true;

        if (autoTest)
        {
            Publish();
        }
    }
    protected override void OnConnectionFailed(string errorMessage)
    {
        Debug.Log("CONNECTION FAILED! " + errorMessage);
    }
    protected override void OnDisconnected()
    {
        Debug.Log("Disconnected.");
        isConnected = false;
    }
    protected override void OnConnectionLost()
    {
        Debug.Log("CONNECTION LOST!");
    }
    protected override void SubscribeTopics()
    {
        client.Subscribe(new string[] { topicSubscribe }, new byte[] {
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
    }
    protected override void UnsubscribeTopics()
    {
        client.Unsubscribe(new string[] { topicSubscribe });
    }
    protected override void Start()
    {
        base.Start();
    }
    protected override void DecodeMessage(string topic, byte[] message)
    {
        // The message is decoded
        msg = System.Text.Encoding.UTF8.GetString(message);

        //Debug.Log("Received: " + msg);
        //Debug.Log("from topic" + m_msg);
        //Debug.Log("from topic" + topic);

        StoreMessage(msg);
        if (topic == topicSubscribe)
        {
            if (autoTest)
            {
                autoTest = false;
                Disconnect();
            }
        }
    }
    private void StoreMessage(string eventMsg)
    {
```

```
        if (eventMessages.Count > 50)
        {
            eventMessages.Clear();
        }
        eventMessages.Add(eventMsg);
    }
    protected override void Update()
    {
        base.Update();
    }

    protected void OnDestroy()
    {
        Disconnect();
    }
    private void OnValidate()
    {
        if (autoTest)
        {
            autoConnect = true;
        }

    }
}
```

ANEXO 2: mqttController código de programa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using System.Diagnostics;
using System.Text.RegularExpressions;

public class mqttController : MonoBehaviour
{
    public string nameController = "Controller 1";
    public string tagOfTheMQTTReceiver = "";
    public mqttReceiver _eventSender;

    public void Start()
    {
        //_eventSender =
GameObject.FindGameObjectsWithTag(tagOfTheMQTTReceiver)[0].gameObject.GetComponent<mqttRe
ceiver>();
        _eventSender =
GameObject.FindGameObjectsWithTag(tagOfTheMQTTReceiver)[0].GetComponent<mqttReceiver>();
        _eventSender.OnMessageArrived += OnMessageArrivedHandler;
    }

    private void OnMessageArrivedHandler(string newMsg)
    {
        UnityEngine.Debug.Log("Event Fired. The message, from Object " + nameController + " is = " +
_eventSender.topicSubscribe + " " + newMsg);
        this.GetComponent<TextMeshPro>().text = newMsg;
    }

}
```

ANEXO 3: mqttController2 código de programa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class mqttController2 : MonoBehaviour
{
    public string nameController = "Controller 1";
    public string tagOfTheMQTTReceiver = "";
    public mqttReceiver _eventSender;

    public void Start()
    {
        //_eventSender =
GameObject.FindGameObjectsWithTag(tagOfTheMQTTReceiver)[0].gameObject.GetComponent<mqttRe
ceiver>();
        _eventSender =
GameObject.FindGameObjectsWithTag(tagOfTheMQTTReceiver)[0].GetComponent<mqttReceiver>();
        _eventSender.OnMessageArrived += OnMessageArrivedHandler;
    }

    private void OnMessageArrivedHandler(string newMsg)
    {
        Debug.Log("Event Fired. The message, from Object " + nameController + " is = " +
_eventSender.topicSubscribe + " " + newMsg);
        this.GetComponent<TextMeshProUGUI>().text = newMsg;

    }

}
```

ANEXO 4: Salir código de programa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Salir : MonoBehaviour
{
    public void SalirApp()
    {
        Application.Quit();
    }
}
```