



**UNIVERSIDAD POLITÉCNICA SALESIANA  
SEDE CUENCA**

**CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN**

ANÁLISIS DE LA HERRAMIENTA UNITY 3D PARA LA CREACIÓN DE  
PROCESOS VIRTUALES Y SU AUTOMATIZACIÓN CON EL PLC SIMATIC S7-1500

Trabajo de titulación previo a la obtención del  
título de Ingeniero en Electrónica

AUTOR: LAURA CAMILA ESPINOZA ENDARA  
KEVIN ALEXANDER LOJA RODAS

TUTOR: ING. JULIO CESAR ZAMBRANO ABAD, PhD.

Cuenca – Ecuador

2024

## CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Laura Camila Espinoza Endara con documento de identificación N° 0604865485 y Kevin Alexander Loja Rodas con documento de identificación N° 0106663743; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 29 de julio de 2024

Atentamente,



---

Laura Camila Espinoza Endara

0604865485



---

Kevin Alexander Loja Rodas

0106663743

**CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE  
TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA**

Nosotros, Laura Camila Espinoza Endara con documento de identificación N° 0604865485 y Kevin Alexander Loja Rodas con documento de identificación N° 0106663743, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: “Análisis de la herramienta Unity 3D para la creación de procesos virtuales y su automatización con el PLC SIMATIC S7-1500” el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 29 de julio de 2024

Atentamente,



---

Laura Camila Espinoza Endara

0604865485



---

Kevin Alexander Loja Rodas

0106663743

## CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Julio Cesar Zambrano Abad con documento de identificación N° 0301489696, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: ANÁLISIS DE LA HERRAMIENTA UNITY 3D PARA LA CREACIÓN DE PROCESOS VIRTUALES Y SU AUTOMATIZACIÓN CON EL PLC SIMATIC S7-1500, realizado por Laura Camila Espinoza Endara con documento de identificación N° 0604865485 y Kevin Alexander Loja Rodas con documento de identificación N° 0106663743, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 29 de julio de 2024

Atentamente,



---

Ing. Julio Cesar Zambrano Abad, PhD

0301489696

# AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a todas las personas que hicieron posible la realización de este proyecto sobre todo a mi familia quien a sido mi apoyo más grande, les agradezco por todo el sacrificio que han dado por mi. Primero, quiero expresar mi agradecimiento a mi tutor del proyecto, Ing. Julio Zambrano, PhD, por su guía, apoyo y paciencia a lo largo de todo el proceso. Su experiencia y conocimientos han sido esenciales para el avance y la culminación exitosa de este proyecto. No puedo dejar de agradecer a mi pareja, quien ha sido mi compañero constante durante todo este proceso. Su apoyo incondicional y su ayuda en los momentos más difíciles han sido invaluable para mí. Por ello, estaré eternamente agradecida.

**Laura Camila Espinoza Endara**

Quisiera expresar mi más sincero y profundo agradecimiento a todas las personas que han hecho posible la culminación de este trabajo de titulación. Primero, deseo manifestar mi más profundo agradecimiento al Ing. Julio Zambrano, PhD, por su invaluable guía y apoyo durante todo este proceso. Su paciencia han sido fundamentales para superar los desafíos y alcanzar el éxito en este proyecto. A mi familia, mi amor eterno y mi mayor apoyo, les debo una gratitud inmensa. Su amor incondicional, comprensión y constante ánimo han sido el pilar sobre el cual he construido cada logro. Su apoyo durante toda mi vida universitaria ha sido fundamental para alcanzar este sueño. A mi amiga Camila, compañera de tantos momentos inolvidables a lo largo de mi carrera universitaria, le agradezco por estar siempre a mi lado. Su amistad sincera y su apoyo inquebrantable han convertido cada etapa de este viaje en una experiencia más rica y significativa. A Emilia y Sara, quienes estuvieron a mi lado al final del trayecto universitario, les agradezco profundamente por su apoyo y amistad. Su presencia y aliento en los momentos críticos han sido un bálsamo para el alma y una fuente de fuerza inagotable. Finalmente, al amor de mi vida, mi gratitud es infinita. Su amor, paciencia y comprensión han sido mi refugio en los momentos de incertidumbre y mi mayor alegría a lo largo de este proceso. Su apoyo constante ha hecho que cada desafío sea más llevadero y cada éxito más dulce. A todos ustedes, gracias por ser parte fundamental de este viaje y por contribuir a la realización de este sueño. Su presencia en mi vida es el verdadero tesoro que valoro más allá de cualquier logro.

**Kevin Alexander Loja Rodas**

# DEDICATORIAS

## *Dedicatoria de Laura Camila Espinoza Endara*

Dedico esta tesis a mi querida familia, cuyos actos de amor y sacrificio incondicionales han sido el pilar fundamental en cada paso de mi vida y en este proceso académico. Su apoyo constante no solo me ha sostenido, sino que ha sido una fuente de inspiración y fortaleza inigualable. Cada palabra de aliento y cada gesto de comprensión han sido una luz en los momentos más oscuros, impulsándome a seguir adelante con renovada determinación.

A mis padres, cuyas sabias enseñanzas y amor incondicional han sido mi guía constante, por su apoyo inquebrantable y su capacidad para estar a mi lado en cada desafío. Este logro es un reflejo de su confianza y dedicación, y no habría sido posible sin su inestimable ayuda y apoyo continuo. Este trabajo es tanto mío como de ustedes, y es un testimonio del impacto profundo que han tenido en mi vida. Con todo mi amor y profunda gratitud, les dedico este logro.

## *Dedicatoria de Kevin Alexander Loja Rodas*

Dedico esta tesis con todo mi corazón a mi familia, cuyo amor incondicional y apoyo constante han sido el pilar fundamental a lo largo de mi vida. Su comprensión, paciencia y aliento me han dado la fuerza necesaria para superar cada desafío y alcanzar este importante logro. A mi pareja, a quien agradezco profundamente por ser mi mayor apoyo y fuente de inspiración. Su amor, comprensión y paciencia han sido el faro que ha iluminado mi camino durante este proceso. Su constante aliento y creencia en mí han hecho que cada paso hacia este objetivo sea más llevadero y gratificante. Esta tesis es el reflejo del esfuerzo y la dedicación de todos ustedes. Gracias por estar siempre a mi lado y por ser una parte esencial en la realización de este sueño.



# Índice general

<b>Agradecimientos</b>	<b>I</b>
<b>Dedicatorias</b>	<b>III</b>
<b>Índice General</b>	<b>V</b>
<b>Índice de figuras</b>	<b>IX</b>
<b>Resumen</b>	<b>X</b>
<b>Abstract</b>	<b>XI</b>
<b>Antecedentes o Problema de Estudio</b>	<b>1</b>
<b>Justificación</b>	<b>3</b>
<b>Objetivos</b>	<b>4</b>
Objetivo General . . . . .	4
Objetivos Específicos . . . . .	4
<b>Introducción</b>	<b>5</b>
<b>1. La herramienta Unity 3D</b>	<b>8</b>
1.1. Introducción . . . . .	8
1.2. Descarga e Instalación de Unity 3D . . . . .	9
1.2.1. Descarga de Unity 3D . . . . .	9
1.3. Ecosistema de desarrollo de Unity . . . . .	13
1.3.1. Lenguaje de Programación C# . . . . .	14

1.3.2. Herramientas y Extensiones . . . . .	16
<b>2. Implementación del proceso virtual</b>	<b>20</b>
2.1. Planificación y diseño del proceso virtual . . . . .	20
2.1.1. Funcionamiento . . . . .	20
2.1.2. Variables de entrada . . . . .	21
2.1.3. Variables de salida . . . . .	21
2.1.4. Estructura de Carpetas de Activos . . . . .	22
2.1.5. Recursos utilizados . . . . .	22
2.2. Configuración y programación del proceso virtual. . . . .	24
<b>3. Interfaz de comunicación y programación del PLC</b>	<b>35</b>
3.1. Introducción al PLC SIMATIC S7-1500 . . . . .	35
3.2. Configuración del proyecto e interfaz de comunicación. . . . .	36
3.2.1. Creación del proyecto: . . . . .	36
3.2.2. Configuración de la Interfaz de Comunicación: . . . . .	41
3.3. Programación del PLC en TIA PORTAL . . . . .	44
3.3.1. Segmento 1 . . . . .	44
3.3.2. Segmento 2 . . . . .	45
3.3.3. Segmento 3 . . . . .	46
3.3.4. Segmento 4 . . . . .	47
3.3.5. Segmento 5 . . . . .	48
3.3.6. Segmento 6 . . . . .	49
3.4. Interfaz de Comunicación con el Proceso Virtual . . . . .	49
3.5. Pruebas de funcionamiento . . . . .	56
<b>4. Conclusiones y Futuros Trabajos</b>	<b>60</b>
4.1. Conclusiones . . . . .	60
4.2. Futuros Trabajos . . . . .	61
<b>Glosario</b>	<b>63</b>
<b>Referencias</b>	<b>66</b>

# Índice de figuras

1.1. Selección de opción de descarga . . . . .	9
1.2. Proceso de descarga de Unity 3D . . . . .	10
1.3. Archivo descargado . . . . .	10
1.4. Términos de la licencia. . . . .	11
1.5. Inicio de la instalación. . . . .	11
1.6. Proceso de instalación en curso. . . . .	12
1.7. Finalización de la instalación. . . . .	12
1.8. Icono de Unity 3D instalado en el escritorio. . . . .	13
1.9. Simulación de un entorno virtual con Unity 3D . . . . .	14
1.10. Script editor de código. . . . .	15
1.11. Utilización de librerías en Unity . . . . .	16
1.12. Asset Store [16] . . . . .	17
1.13. Aplicaciones Industriales [17] . . . . .	18
1.14. Documentación . . . . .	18
2.1. Proceso Industrial en 2D . . . . .	21
2.2. Estructura de carpetas de activos en Unity 3D. . . . .	23
2.3. Estructura de carpetas de activos en Unity 3D. . . . .	24
2.4. Actuator . . . . .	25
2.5. Button Controller . . . . .	26
2.6. Camera Controller . . . . .	26
2.7. ConveyorBelt . . . . .	27
2.8. ConveyorBeltCollider . . . . .	28
2.9. ConveyorController . . . . .	28

2.10. Destroyer . . . . .	29
2.11. GaemObjectSensor . . . . .	30
2.12. GameObjectSensorInductive . . . . .	31
2.13. GameObjectSensorNonMetal . . . . .	32
2.14. IndicatorController . . . . .	32
2.15. Paths. . . . .	34
3.1. Creación de un proyecto en TIA Portal. . . . .	36
3.2. Detalles para la creación de un proyecto. . . . .	37
3.3. Interfaz Inicial de TIA Portal. . . . .	37
3.4. Adición de Dispositivos. . . . .	38
3.5. Vista de dispositivos en TIA Portal. . . . .	38
3.6. Configuración de Módulos de Potencia. . . . .	39
3.7. Incorporación de Módulos de Entradas Analógicas. . . . .	39
3.8. Incorporación de Módulos de Salidas Analógicas. . . . .	40
3.9. Adición de Módulos de Entradas Digitales. . . . .	40
3.10. Configuración de Módulos de Salidas Digitales. . . . .	40
3.11. Configuración inicial del PLC. . . . .	41
3.12. Configuración de nivel de acceso en Protección y Seguridad. . . . .	41
3.13. Configuración de mecanismos de conexión. . . . .	42
3.14. Creación de un bloque de datos. . . . .	42
3.15. Propiedades del bloque de datos. . . . .	43
3.16. Deshabilitar acceso optimizado al bloque. . . . .	43
3.17. Imagen del Bloque 1. . . . .	44
3.18. Imagen del Bloque 2. . . . .	45
3.19. Imagen del Bloque 3. . . . .	46
3.20. Imagen del Bloque 4. . . . .	47
3.21. Imagen del Bloque 5. . . . .	48
3.22. Imagen del Bloque 6. . . . .	49
3.23. Configuración de Módulos de Potencia. . . . .	50
3.24. Configuración de Variables de Clase. . . . .	51
3.25. Métodos de Inicialización. . . . .	52

3.26. Método Update y Conexión al PLC. . . . .	53
3.27. Lectura de Datos del PLC. . . . .	54
3.28. Lectura de Datos del PLC. . . . .	54
3.29. Métodos para Establecer Estados de Componentes. . . . .	55
3.30. Métodos para Establecer Estados de Componentes. . . . .	55
3.31. Métodos para Establecer Estados de Componentes. . . . .	56
3.32. Prueba de funcionamiento 1. . . . .	57
3.33. Prueba de funcionamiento 2. . . . .	57
3.34. Prueba de funcionamiento 3. . . . .	58
3.35. Prueba de funcionamiento 4. . . . .	58
3.36. Prueba de funcionamiento 5. . . . .	59
3.37. Prueba de funcionamiento 6. . . . .	59

# Resumen

Este proyecto resalta la importancia de la comunicación entre los controladores lógicos programables (PLC) SIMATIC S7-1500 y la plataforma Unity 3D a través del protocolo de comunicación Sharp 7, que optimiza la gestión de datos y la ejecución de comandos. Esta integración permite a los estudiantes del software TIA PORTAL realizar configuraciones de procesos mediante la programación en lenguaje LADDER para la manipulación de variables y la creación de condicionales que pueden visualizarse en Unity 3D.

El proyecto también aborda la necesidad de crear un entorno virtual 3D específicamente diseñado para simular procesos industriales. Como parte del proceso de validación, se llevarán a cabo pruebas físicas en los laboratorios de Redes Industriales de la Universidad Politécnica Salesiana, lo que contribuirá a mejorar las prácticas y permitirá a los estudiantes obtener una comprensión más profunda de un entorno industrial.

Este proyecto concluye que la implementación de estas tecnologías mejora la experiencia educativa de los estudiantes y los prepara mejor para los desafíos del mundo laboral moderno.

**Palabras clave:** Comunicación PLC SIMATIC S7-1500, Plataforma Unity 3D, Protocolo de comunicación Sharp 7, Programación en lenguaje LADDER, Entorno virtual 3D, Simulación de procesos industriales.

# Abstract

This project highlights the importance of communication between SIMATIC S7-1500 programmable logic controllers (PLC) and the Unity 3D platform through the Sharp 7 communication protocol, which optimizes data management and command execution. This integration allows students using the TIA PORTAL software to configure processes through programming in LADDER language for variable manipulation and creation of conditionals that can be visualized in Unity 3D.

The project also addresses the need to create a 3D virtual environment specifically designed to simulate industrial processes. As part of the validation process, physical tests will be carried out in the Industrial Networks laboratories of the Salesian Polytechnic University, which will help improve practices and allow students to gain a deeper understanding of an industrial environment.

This project concludes that the implementation of these technologies enhances the educational experience of students and better prepares them for the challenges of the modern working world.

**Keywords:** SIMATIC S7-1500 PLC Communication, Unity 3D Platform, Sharp 7 Communication Protocol, LADDER Language Programming, 3D Virtual Environment, Industrial Process Simulation.

# Antecedentes

La implementación de Entornos Virtuales en Unity 3D y controlados con PLCs es fundamental tanto en el ámbito académico como en el ámbito profesional. Esta combinación representa un avance significativo hacia la preparación para la Industria 4.0, donde la integración de sistemas físicos y virtuales proporciona una valiosa experiencia práctica y segura a estudiantes y profesionales en un entorno simulado. Esta integración permite familiarizarse con la interacción entre sistemas virtuales y físicos antes de enfrentarse a situaciones reales en el campo laboral.

El interés en la implementación de entornos virtuales en el ámbito estudiantil surge de la necesidad de ofrecer una formación práctica y actualizada en tecnologías emergentes. La implementación de entornos virtuales con Unity 3D y PLC SIMATIC S7-1500 permite a las empresas optimizar sus procesos de producción, realizar simulaciones y entrenamientos virtuales, mejorando la gestión de datos en tiempo real con el objetivo de comprobar su buen funcionamiento.

En el ámbito académico, en el año 2021 se realizó una investigación dirigida al desarrollo de un entorno virtual 3D diseñado para simular un proceso batch en los laboratorios de automatización, este entorno virtual permitía visualizar las variables del proceso en un HMI (Interfaz Hombre-Máquina) dentro del entorno virtual 3D en el cual se utilizó TIA Portal con el PLC S7-1200 mediante diagramas GRAFCET para su programación [1], [2]. Esta investigación se validó con estudiantes universitarios quienes mostraron una alta aceptación.

En una investigación realizada en el 2020, se examinó la falta de formación universitaria en controladores lógicos programables (PLCs). El estudio reveló que los sistemas de eventos discretos, realizados mediante PLCs, son una de las áreas con mayor déficit educativo. Se destacó la necesidad de preparar a los estudiantes para las



demandas del mercado laboral en el campo de la automatización industrial [3]. Con esto en mente, se reconoce la oportunidad de brindar un valor adicional en el ámbito estudiantil mediante la implementación de crear un entorno virtual con Unity 3D con base a un proceso industrial real que será automatizado.

En el año 2021 se realizó una investigación dirigida al ámbito laboral en la cual se enfocó en mejorar la capacitación en seguridad para operadores de equipos pesados en la industria minera. Este estudio utiliza tecnologías modernas, como cámaras de 360 grados, con la plataforma Unity 3D. Este proyecto se cero para desarrollar escenarios y herramientas de capacitación en seguridad [4]. Con esto en mente se tiene en cuenta la importancia de desarrollar recursos efectivos para la preparación del personal, brindando así mayor seguridad en su desempeño laboral.

Estos y otros trabajos más de investigación en cuanto al desarrollo de entornos virtuales avalan la necesidad de contar con estas herramientas, tanto en el ámbito académico como en el ámbito profesional. En el ámbito académico, estas herramientas pueden acercar más al estudiante a un entorno industrial real para realizar pruebas sobre la automatización de procesos. Eso resulta altamente necesario, ya que normalmente a nivel de universidades resulta complicado contar con plantas o procesos industriales de alta escala para hacer pruebas de laboratorio, ya sea por limitaciones económicas o limitaciones físicas. Por otra parte, a nivel industrial, un entorno virtual puede servir para capacitar a personal o también para realizar pruebas o experimentos sobre determinados procesos, sin tener la necesidad de interactuar con el proceso real, evitando daños y/o accidentes.

# Justificación

La justificación de este proyecto se basa en varios aspectos fundamentales que responden a las necesidades específicas y a la relevancia en el contexto de la investigación y la educación en tecnología. Primero, la creación de un entorno virtual utilizando Unity 3D se justifica por la necesidad de proporcionar una plataforma interactiva y visualmente atractiva para la simulación de procesos industriales. Esta elección se debe a que Unity 3D es un software libre, lo que facilita su obtención. En segundo lugar, la selección de los PLCs SIMATIC S7-1500 se basa en su disponibilidad y compatibilidad con los recursos existentes en la universidad además de que es lo último en tecnología industrial. Estos PLCs ofrecen capacidades avanzadas y son compatibles con una amplia gama de aplicaciones para la automatización de procesos industriales.

La importancia de la automatización radica en su capacidad para mejorar la eficiencia, la precisión y la seguridad en los procesos industriales, lo que conlleva a una mayor productividad y competitividad en diversos sectores. La simulación de un entorno virtual de un proceso industrial real es fundamental para proporcionar a los estudiantes una experiencia práctica y tangible que complemente su aprendizaje teórico. Esto les permite comprender mejor los conceptos y principios detrás de la automatización industrial y desarrollar habilidades prácticas relevantes para su futura carrera profesional. Además, al ser un espacio simulado en el ámbito industrial, se reduce significativamente el riesgo de accidentes, proporcionando un entorno seguro para el aprendizaje y la experimentación.

# Objetivos

## Objetivo General

- Analizar la herramienta Unity 3D para la creación de procesos virtuales y su automatización con el PLC SIMATIC S7-1500.

## Objetivos específicos:

- Diseñar un entorno virtual con Unity 3D con base a un proceso industrial real que será automatizado.
- Implementar una interfaz para la comunicación entre el entorno virtual y el PLC SIMATIC S7-1500.
- Desarrollar la automatización del proceso utilizando la herramienta informática de TIA Portal.
- Evaluar la automatización y la interacción entre el PLC y el proceso virtual mediante pruebas de laboratorio.

# Introducción

La integración de tecnologías emergentes en el ámbito de la automatización industrial ha revolucionado la forma en que se desarrollan y gestionan los procesos industriales. En este contexto, el presente proyecto destaca la importancia crucial de establecer una comunicación eficaz entre los controladores lógicos programables (PLC) SIMATIC S7-1500 y la plataforma Unity 3D mediante el protocolo de comunicación Sharp 7.

Este proyecto aborda también la necesidad de crear un entorno virtual 3D diseñado específicamente para simular procesos industriales. Las pruebas físicas que se llevarán a cabo en los laboratorios de Redes Industriales de la Universidad Politécnica Salesiana validarán la efectividad de este entorno, contribuyendo a mejorar las prácticas y proporcionando a los estudiantes una comprensión más profunda del entorno industrial en el que eventualmente trabajarán.

La supervisión y el control de todos los parámetros relevantes del proceso es de gran importancia para lograr una alta calidad constante con el menor número posible de rechazos [5].

En este contexto, el presente trabajo se propone explorar cómo la integración de tecnologías como PLC y Unity 3D puede mejorar la experiencia educativa de los estudiantes en el campo de la automatización industrial, al mismo tiempo que los prepara para enfrentar los desafíos y oportunidades que ofrece el mundo laboral moderno.

Por esta razón, es necesario desarrollar propuestas de entornos virtuales de procesos industriales que brinden la oportunidad de una interacción entre software y hardware muy aproximados a la realidad, supliendo fácilmente a los equipos físicos tradicionales [1].

En el presente documento se abordará la integración de Unity 3D con el PLC SIMATIC S7-1500, utilizando el protocolo de comunicación Sharp 7 para establecer una interacción fluida y efectiva entre software y hardware. El enfoque principal es analizar cómo Unity 3D puede ser utilizado para crear y automatizar procesos virtuales que simulen escenarios industriales reales. A lo largo de los capítulos, se explorará desde la introducción a las herramientas y funcionalidades de Unity 3D hasta la implementación detallada de la lógica de procesos virtuales y la integración con sistemas de control automático.

En el primer capítulo, se introduce Unity 3D, explorando su relevancia y aplicabilidad en la simulación de procesos industriales. Se discute el ecosistema de desarrollo que ofrece Unity, incluyendo las herramientas y características clave para el desarrollo de aplicaciones interactivas. Además, se profundiza en el lenguaje de programación C, fundamental para el scripting y la automatización dentro de Unity, y se examinan las plataformas en las que Unity permite desplegar proyectos. También se presentan casos de uso y aplicaciones de Unity en diferentes sectores, especialmente en la industria, y se proporciona una visión de la comunidad y los recursos disponibles para los desarrolladores, como documentación y tutoriales.

El segundo capítulo se centra en la implementación del proceso virtual, describiendo la planificación y el diseño del entorno virtual basado en un proceso industrial real. Se detalla el desarrollo de activos y recursos en Unity 3D necesarios para la simulación, así como la configuración y programación de la lógica del proceso virtual dentro del entorno Unity. Además, se describen los algoritmos específicos empleados para la simulación y control del proceso, asegurando que el entorno virtual se comporte de manera consistente con las expectativas del mundo real.

El tercer capítulo aborda la interfaz de comunicación y la programación del PLC SIMATIC S7-1500. Se comienza con una introducción a las capacidades del PLC, seguido por la configuración de la interfaz que permitirá la comunicación entre Unity y el PLC. Se explica detalladamente cómo se programa el PLC dentro del entorno de TIA Portal para integrarse con la simulación de Unity, enfocándose en cómo esta integración permite una comunicación efectiva y eficiente entre el software y el hardware.

En el cuarto capítulo se presentan las pruebas de funcionamiento, donde se evalúa la implementación y sincronización entre Unity y el PLC S7-1500. Finalmente, en el quinto capítulo se incluyen las conclusiones del proyecto, destacando si los objetivos inicialmente propuestos fueron alcanzados. También se ofrecen recomendaciones basadas en los resultados obtenidos y se sugieren líneas de trabajo futuro que podrían derivarse del proyecto.

# Capítulo 1

## La herramienta Unity 3D

### 1.1. Introducción

Unity3D es reconocido como un potente motor 3D multiplataforma y un entorno de desarrollo que combina facilidad de uso con robustez para satisfacer tanto a principiantes como a expertos en el desarrollo de juegos y aplicaciones. Según Unity Technologies, la compañía detrás de este software, Unity proporciona una plataforma completa para crear juegos innovadores y aplicaciones atractivas en modos 2D, 3D, Realidad Virtual (VR) y Realidad Aumentada (AR) [6].

Unity 3D es la plataforma principal utilizada para desarrollar los recursos educativos virtuales. Ofrece un conjunto completo de soluciones de software para crear contenido interactivo en tiempo real en 2D y 3D [7]. Este entorno de desarrollo integrado (IDE) incluye una variedad de herramientas que permiten a los desarrolladores diseñar, prototipar y desplegar sus proyectos con eficiencia.

El sistema utiliza el motor Unity3D, que puede ser utilizado en múltiples plataformas, como Windows, Mac, iPhone, Android, entre otras. Esta versatilidad permite a los desarrolladores crear una única vez y desplegar en múltiples dispositivos, asegurando una amplia accesibilidad y alcance [8].

Utilizando Unity 3D, se pueden crear aplicaciones para dispositivos de Realidad Aumentada (AR) y Realidad Virtual (VR) como Oculus, HTC Vive y Google Cardboard. Unity 3D soporta el desarrollo en C#, un lenguaje popular y poderoso que permite a los desarrolladores implementar lógicas complejas y personalizar el

comportamiento de los objetos en sus aplicaciones [9].

El entorno de Unity 3D permite la experiencia de realidad aumentada, también conocida como realidad híbrida. Esta tecnología puede ser utilizada para mejorar la percepción del entorno real mediante la superposición de información digital [10].

Esta herramienta propone una arquitectura de sistema de simulación visual y un método de implementación de evaluación de daños en objetivos basado en la tecnología interactiva de Unity3D. Este sistema puede realizar la evaluación de daños en dimensiones estructurales y funcionales de interfaces complejas, proporcionando una herramienta eficaz para diversas aplicaciones industriales [11].

## 1.2. Descarga e Instalación de Unity 3D

### 1.2.1. Descarga de Unity 3D

A continuación se presenta el procedimiento a seguir para descargar la herramienta Unity 3D.

1. **Acceso al sitio web oficial de Unity:** A través de un navegador web se debe ingresar al sitio web de Unity 3D tecleando el siguiente enlace: <https://unity.com/es/download>.
2. **Confirmación de la descarga:** Dentro de la página se debe presionar la opción de "Download" (véase la Figura 1.1).

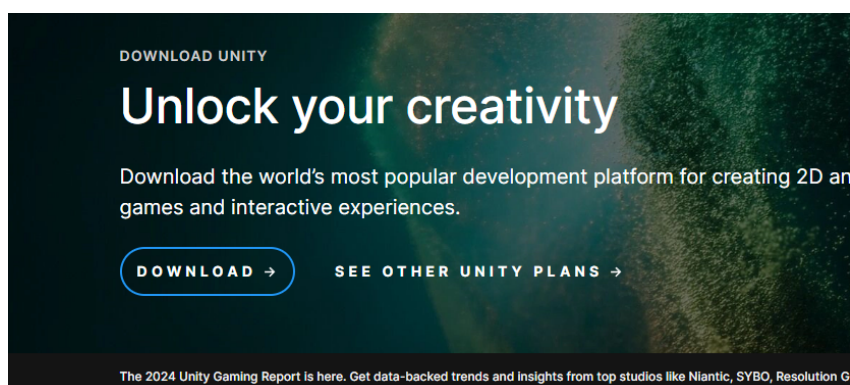


Figura 1.1: Selección de opción de descarga



3. **Inicio de la descarga:** El instalador comenzará a descargarse y el progreso podrá observarse en la sección de descargas del navegador (véase la Figura 1.2).

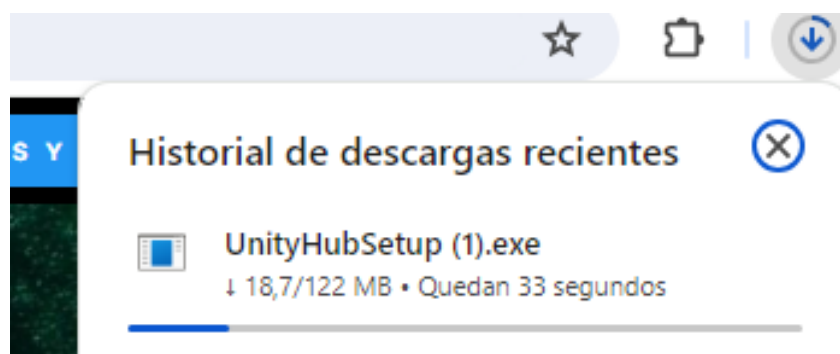


Figura 1.2: Proceso de descarga de Unity 3D

4. **Archivo descargado:** Una vez completada la descarga, se debe acceder a la carpeta donde se descargó el instalador y se debe seleccionar el archivo ejecutable (véase la Figura 1.3).

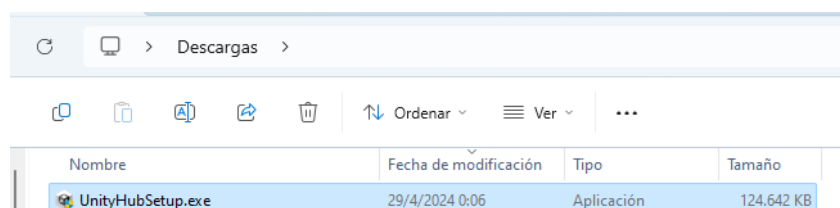


Figura 1.3: Archivo descargado

## Instalación de Unity

5. **Inicio de la instalación:** Al iniciar la instalación se debe aceptar los términos de la licencia presionando el botón "Aceptar" (véase la Figura 1.4).

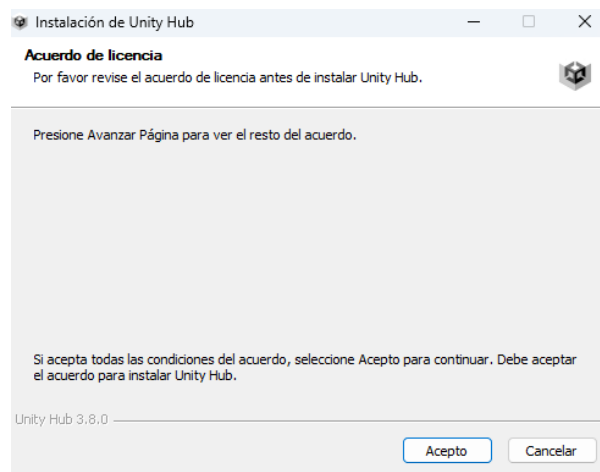


Figura 1.4: Términos de la licencia.

- Selección del directorio de instalación:** Aparecerá una ventana para seleccionar el directorio de instalación. Se debe presionar el botón "Instalar" (véase la Figura 1.5).

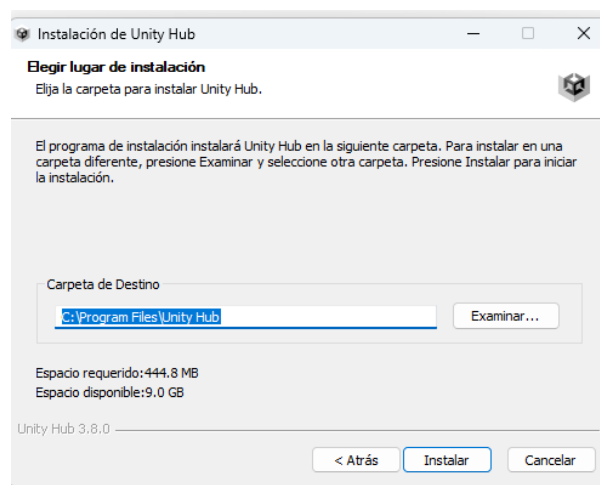


Figura 1.5: Inicio de la instalación.

7. **Proceso de instalación:** La instalación comenzará y se completará en unos minutos (véase la Figura 1.6).

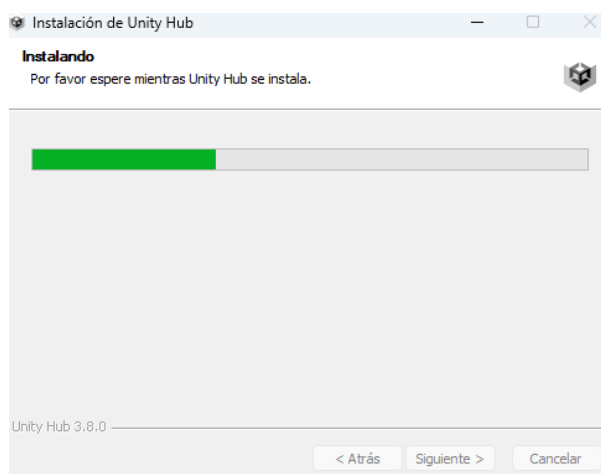


Figura 1.6: Proceso de instalación en curso.

8. **Finalización de la instalación:** Una vez completada la instalación, se mostrará una ventana en la que se deberá presionar el botón "Terminar" (véase la Figura 1.7).

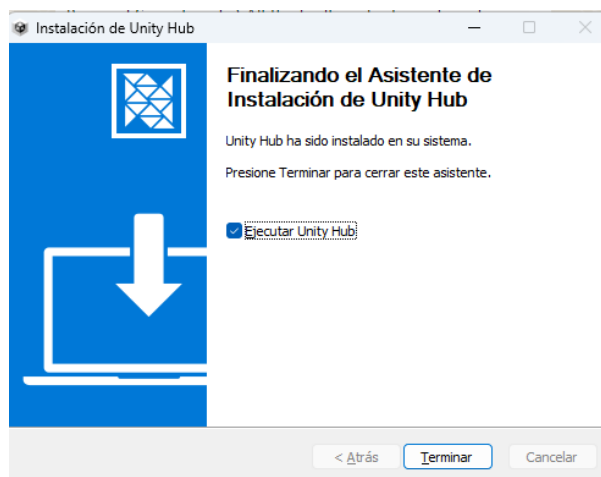


Figura 1.7: Finalización de la instalación.

9. **Comprobación de la instalación:** El icono del programa será visible en el escritorio, indicando que Unity 3D está listo para su uso (véase la Figura 1.8).



Figura 1.8: Icono de Unity 3D instalado en el escritorio.

### 1.3. Ecosistema de desarrollo de Unity

El motor de juegos Unity fue introducido por Unity Technologies en 2005 y desde entonces se ha convertido en una de las plataformas más populares para desarrollar juegos 2D y 3D. Unity no solo proporciona un entorno de desarrollo robusto, sino que también incluye una serie de herramientas y recursos que facilitan el proceso de creación de contenido interactivo [12]. Entre estas herramientas se encuentran el Editor de Escenas, el Sistema de Animación, el Editor de Scripts y la Asset Store, entre otros.

El Editor de Escenas en Unity ofrece una interfaz visual para diseñar y organizar objetos en entornos 3D o 2D. El Sistema de Animación permite crear y gestionar animaciones para personajes y objetos mediante un sistema basado en esqueletos. El Editor de Scripts proporciona un entorno para programar el comportamiento del entorno virtual en C#, controlando la lógica y las interacciones. La Asset Store es un mercado en línea donde se pueden comprar o vender recursos como modelos, texturas y scripts, facilitando la integración de contenido adicional. (véase la Figura 1.9)

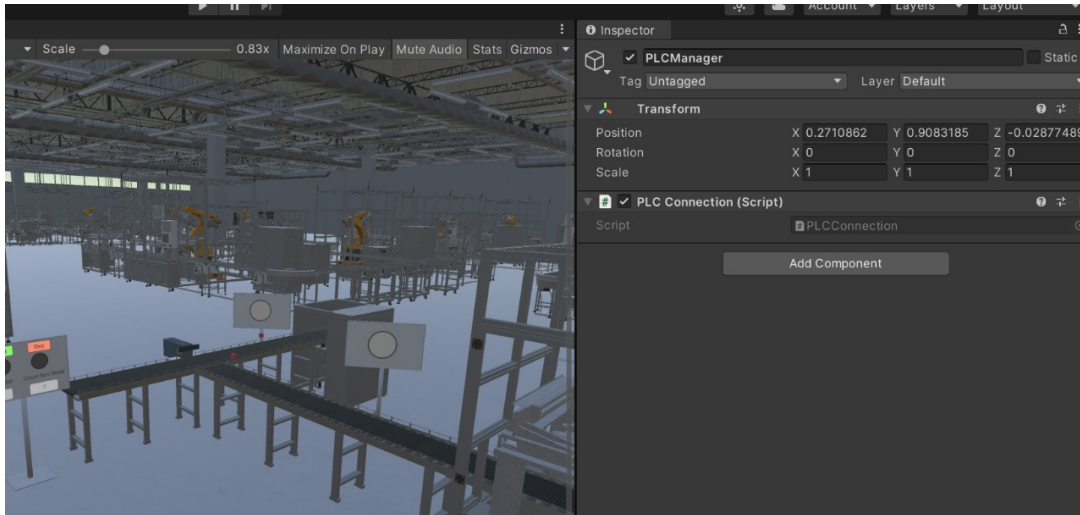


Figura 1.9: Simulación de un entorno virtual con Unity 3D

En la figura 1.9 se ilustra cómo las diversas herramientas de Unity, como el Editor de Escenas y el Sistema de Animación, son utilizadas para crear el entorno interactivo y detallado donde se ha configurado una interfaz de usuario para un entorno industrial simulado. En esta simulación, cuenta con panel de control con botones que permiten iniciar y detener un proceso, así como contadores diseñados para contabilizar objetos metálicos y no metálicos en una cinta transportadora.

### 1.3.1. Lenguaje de Programación C#

C# fue reconocido como lenguaje de programación en el año en 2023. Este logro sobresaliente es consistente con la victoria del lenguaje sobre obstáculos y su desarrollo como una potencia multifacética y multiplataforma1 [13]. Se destaca por su sintaxis clara y fácil de aprender, lo que lo hace ideal tanto para principiantes como para desarrolladores experimentados. Su integración con Unity facilita la creación de aplicaciones complejas y de alto rendimiento [9](véase la Figura 1.10)

```
private S7Client client;
private byte[] buffer = new byte[6]; // Ajusta el tamaño del buffer para leer todos los
datos necesarios
private string plcIpAddress = "192.168.0.1";
private int rack = 0;
private int slot = 1;
private float updateInterval = 0.1f;
private float nextUpdateTime = 0f;

private bool conveyor;
private bool startButton;
private bool stopButton = true; // Inicia en verdadero
private bool sensorCountMetal;
private bool sensorCountNonMetal;
private bool actuator = false; // Variable para el estado del actuador
private int indicatorMetalParts;
private int indicatorNonMetalParts;

void Start()
{
    client = new S7Client();
    ConnectToPLC();

    // Establecer el estado inicial del botón de parada en verdadero
    Invoke("InitializeStopButtonState", 1.0f); // Retraso para asegurar la conexión
antes de establecer el estado

    // Inicializar el estado del actuador en false
    Invoke("InitializeActuatorState", 1.0f); // Retraso para asegurar la conexión
antes de establecer el estado
}
```

Figura 1.10: Script editor de código.

En la figura 1.10 se presenta un ejemplo de programa en C#, mediante el cual se logra que Unity interactúe con un PLC usando la biblioteca S7Client. En el código se declara varias variables para la configuración del sistema, incluyendo la dirección IP del PLC, su ubicación física (rack y slot). En el método Start(), se crea una instancia de S7Client, se conecta al PLC y se configura el estado inicial de varios componentes.

## Bibliotecas

Unity proporciona una serie de bibliotecas estándar, junto con un concepto de tienda de recursos que es un buen modo de gestión de materiales. La disponibilidad de estas bibliotecas reduce significativamente el tiempo de desarrollo, al proporcionar soluciones listas para utilizar que pueden integrarse fácilmente en cualquier proyecto [14]. Incluyen una amplia gama de funciones predefinidas que facilitan tareas comunes en el desarrollo de programas, como la gestión de gráficos, la física, el audio y la entrada del usuario (véase la Figura 1.11).

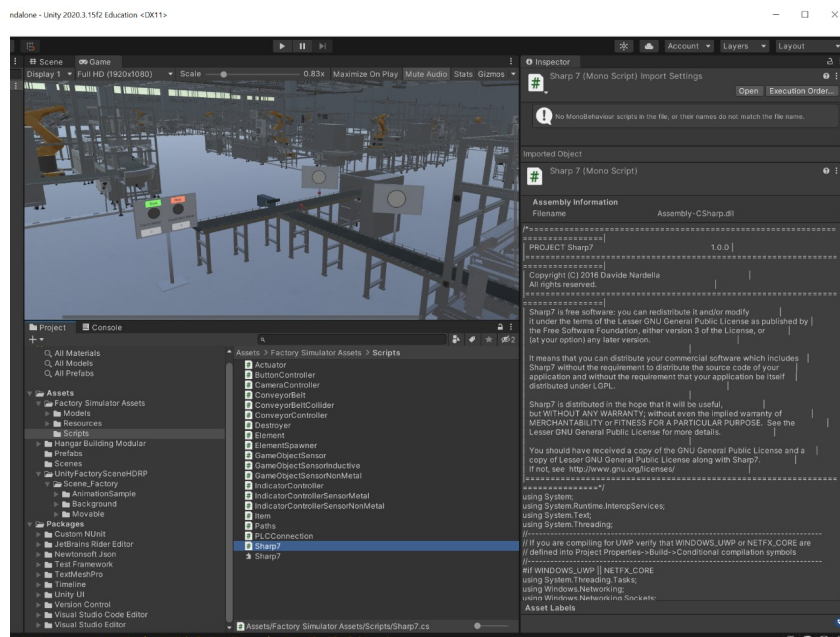


Figura 1.11: Utilización de librerías en Unity

La figura 1.11 muestra el editor de Unity con la simulación industrial, utilizando varias bibliotecas y paquetes para mejorar sus funcionalidades. Se observan bibliotecas como *TextMeshPro* y *Sharp7*. *TextMeshPro* proporciona capacidades avanzadas para el renderizado de texto, mientras que *Sharp7* es un script que emplea bibliotecas de C# como *System.Text* y *System.Threading*, manejando funciones de texto.

### 1.3.2. Herramientas y Extensiones

#### Plugins

Los plugins de Unity son extensiones diseñadas para agregar funcionalidades adicionales, proporcionando herramientas extras para la creación de contenido y la optimización del rendimiento [15]. Se puede utilizar plugins como ProBuilder para modelado 3D, Cinemachine para cámaras avanzadas, TextMesh Pro para mejorar el texto, DOTween para facilitar el manejo de las animaciones, Gaia para optimizar el control de personajes y la generación de entornos. Estos plugins están disponibles en la Unity Asset Store y otros sitios web, permitiendo adaptar y mejorar los proyectos.

La Asset Store de Unity es una herramienta clave para la colaboración, permitiendo a los desarrolladores compartir y vender activos que pueden ser

utilizados por otros en sus proyectos. En la figura 1.12 se muestra el sitio web en donde se los pueden encontrar.

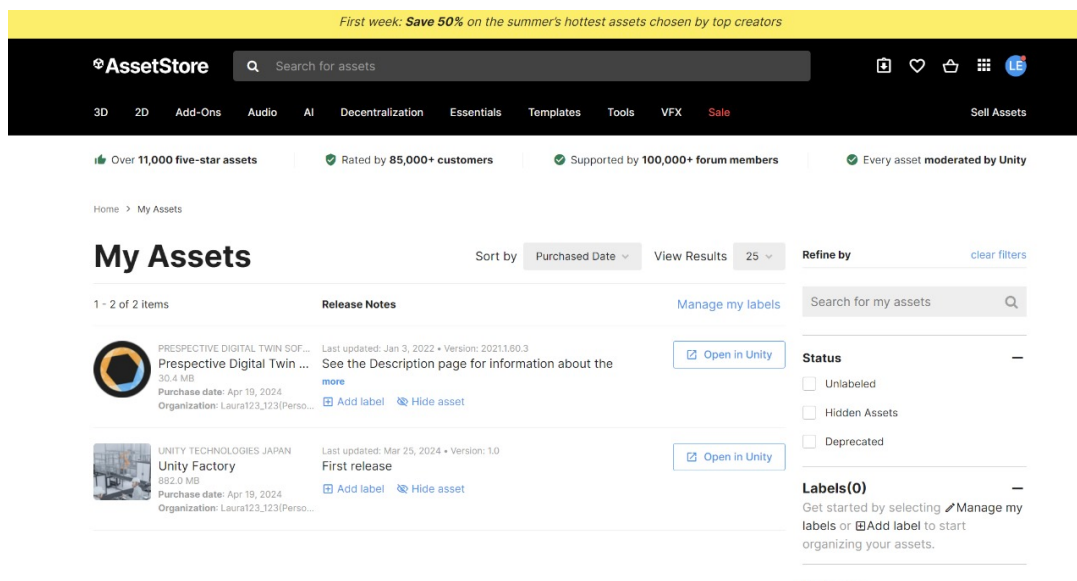


Figura 1.12: Asset Store [16]

## Simulaciones

A través de la tecnología de simulación virtual, los estudiantes pueden practicar en un escenario virtual, lo cual tiene un efecto positivo en la mejora de sus habilidades prácticas, aumentando su interés en el aprendizaje y estimulando el pensamiento innovador. Estas simulaciones permiten a los usuarios interactuar con entornos virtuales realistas, proporcionando una experiencia práctica sin los riesgos y costos asociados con las operaciones del mundo real [15]. Unity es utilizado en una amplia variedad de aplicaciones industriales, incluyendo simulaciones de entrenamiento, diseño de productos y análisis de procesos como se muestra en la figura 1.13.



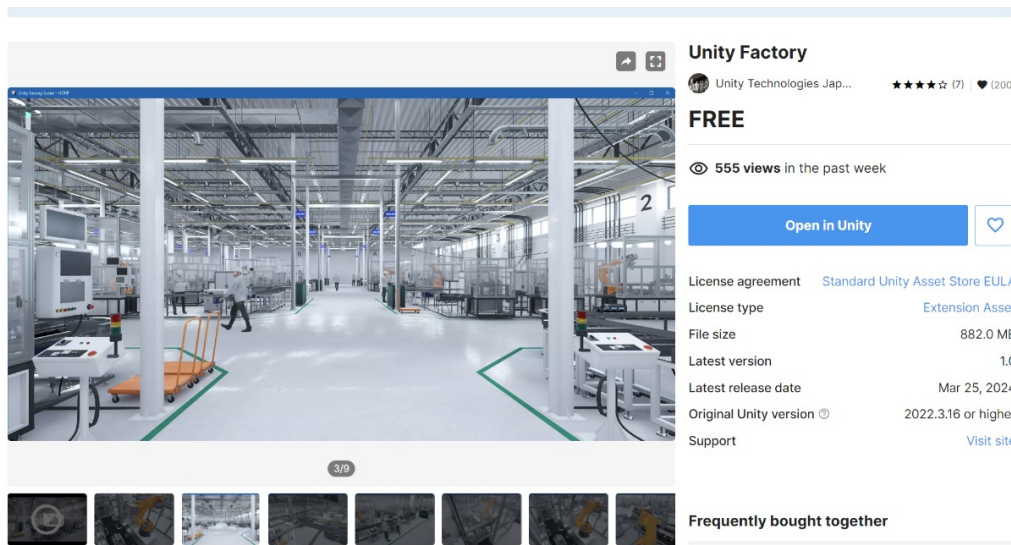


Figura 1.13: Aplicaciones Industriales [17]

## Documentación

La documentación son archivos extensos y detallados, proporcionando a los desarrolladores una gran cantidad de recursos para aprender y resolver problemas, facilitando el proceso de desarrollo. Estos recursos cubren una amplia gama de temas, desde conceptos básicos hasta técnicas avanzadas, y están diseñados para ayudar a los desarrolladores a aprovechar al máximo las capacidades de Unity [14] (véase la Figura 1.14).

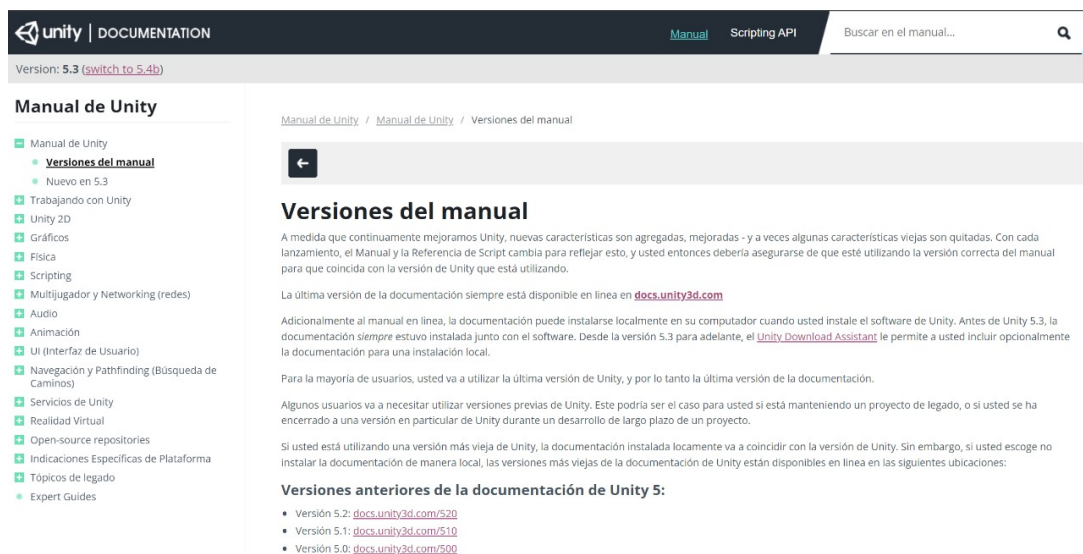


Figura 1.14: Documentación

Estos incluyen guías detalladas, ejemplos de código que facilitan la comprensión y el uso de las diversas características del motor. Los tutoriales paso a paso proporcionan una forma práctica de aprender, permitiendo a los desarrolladores seguir instrucciones claras como se observa en la figura 1.14 un manual que se puede aplicar lo que han aprendido en sus propios proyectos.

# Capítulo 2

## Implementación del proceso virtual

### 2.1. Planificación y diseño del proceso virtual

En este apartado se detalla la estructura y funcionamiento del sistema de control automatizado basado en el PLC SIMATIC S7-1500. Se especificarán las variables que interactuarán con el PLC, diferenciando entre variables de entrada y variables de salida. Además, se aborda la implementación del sistema, describiendo cómo se integran y configuran las diferentes variables y componentes para lograr un funcionamiento óptimo del proceso automatizado.

#### 2.1.1. Funcionamiento

Para el presente proyecto se plantea la implementación de un sistema de clasificación de material basado en cintas transportadoras y elementos de detección. La Figura 2.1 muestra un bosquejo en dos dimensiones del proceso industrial a implementar.

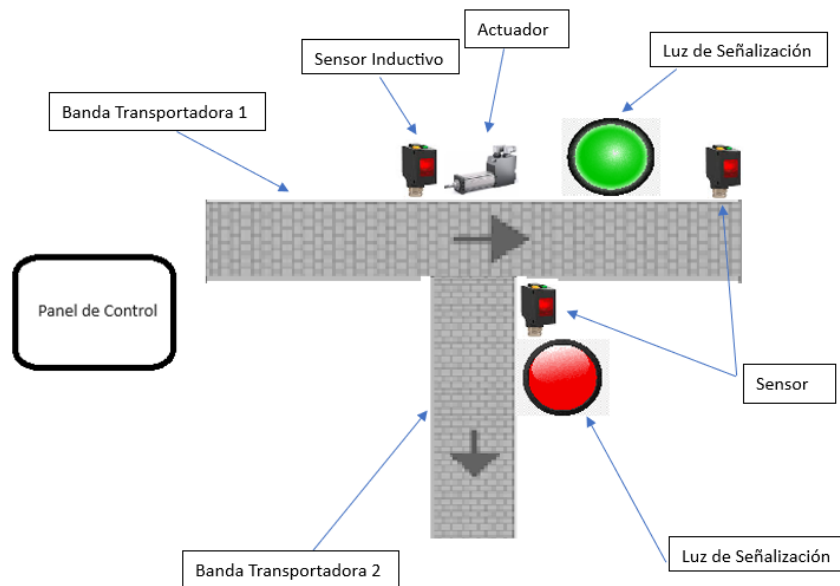


Figura 2.1: Proceso Industrial en 2D

Se diseñó un sistema de control automatizado utilizando el PLC SIMATIC S7-1500 y el software TIA Portal, donde se gestionan un total de 11 variables, divididas en 5 de entrada y 6 de salida. Las variables de entrada incluyen botones de arranque y parada, así como varios sensores que detectan distintos tipos de materiales. Por otro lado, las variables de salida controlan elementos como las bandas y el actuador.

### 2.1.2. Variables de entrada

- **Start btn 1:** Botón de arranque
- **Stop btn 1:** Botón de parada
- **Sensor Count Metal:** Sensor contador de metal
- **Sensor Count Non Metal:** Sensor contador de no metal
- **Sensor Inductive:** Sensor inductivo

### 2.1.3. Variables de salida

- **Conveyor:** Transportador
- **Indicator Green:** Indicador verde

- **Indicator Red:** Indicador rojo
- **Actuator:** Actuador
- **Indicator Metal Parts:** Indicador de partes metálicas
- **Indicator Non Metal Parts:** Indicador de partes no metálicas

La implementación del sistema se lleva a cabo utilizando el software TIA Portal, el cual permite la programación y configuración del PLC S7-1500. En este proceso, se definen las conexiones y la lógica de control necesarias para asegurar que todas las variables interactúen de manera coherente y eficiente, logrando así una automatización robusta y confiable del proceso industrial.

Al finalizar este apartado, se tendrá una visión clara de las variables involucradas y la metodología utilizada para su integración y control, lo que permitirá comprender en detalle el funcionamiento del sistema automatizado propuesto.

#### 2.1.4. Estructura de Carpetas de Activos

La carpeta Assets es la raíz de todos los activos en un proyecto de Unity. Dentro de esta carpeta, se organizan todos los subdirectorios que contienen los diferentes tipos de recursos necesarios para el desarrollo del proyecto. Para el presente proyecto se utilizaron las siguientes carpetas: Factory Simulator, Hangar Building Modular y UnityFactorySceneHDRP.

#### 2.1.5. Recursos utilizados

- **Models:** Esta subcarpeta contiene modelos 3D utilizados en el simulador de fábrica. Los modelos pueden estar en formatos como FBX, OBJ, entre otros. La correcta organización y nomenclatura de los archivos de modelos es crucial para mantener la claridad y el orden.
- **Resources:** En esta carpeta se almacenan recursos adicionales necesarios para el simulador de fábrica. Estos pueden incluir texturas, materiales, animaciones y otros archivos relacionados.

- **Scripts:** Aquí se encuentran los scripts escritos en C# que definen la lógica y el comportamiento del simulador de fábrica. Mantener los scripts organizados y bien documentados facilita su mantenimiento y actualización.
- **Prefabs:** Los prefabs son objetos preconfigurados que se pueden reutilizar en diferentes partes del proyecto. En esta subcarpeta se almacenan prefabs relacionados con la construcción modular del hangar.
- **Scenes:** Esta carpeta contiene las escenas relacionadas con la construcción modular del hangar. Las escenas en Unity representan diferentes niveles o áreas del juego.
- **AnimationSample:** Aquí se almacenan muestras de animaciones utilizadas en la escena de la fábrica. Estas animaciones pueden incluir movimientos de máquinas, personajes u otros elementos interactivos.
- **Movable:** En esta carpeta se encuentran los elementos que pueden moverse o interactuar en la escena de la fábrica. Estos pueden incluir piezas de maquinaria, objetos manipulables por el jugador, entre otros.

A continuación, en la figura 2.2 se muestra la estructura de carpetas en Unity:

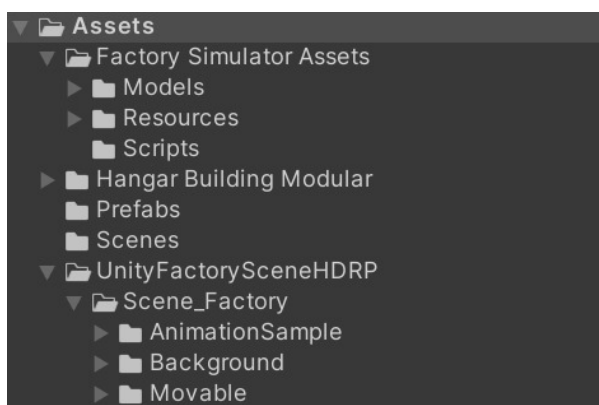


Figura 2.2: Estructura de carpetas de activos en Unity 3D.

## 2.2. Configuración y programación del proceso virtual.

Para la implementación del proyecto es necesario la creación de códigos que permitan el movimiento de cada uno de los objetos, estos van a ser explicados a continuación (véase en la figura 2.3).

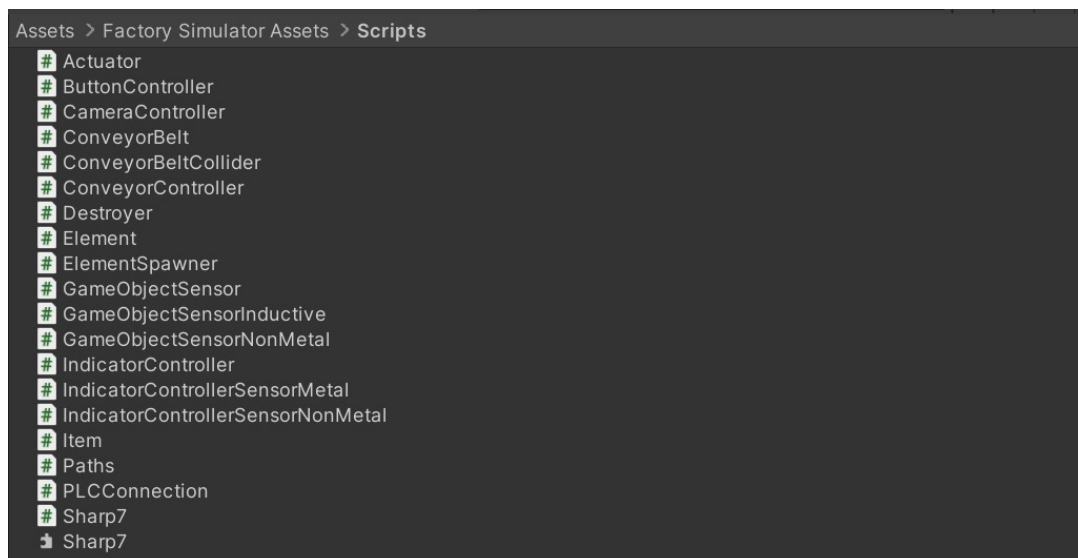


Figura 2.3: Estructura de carpetas de activos en Unity 3D.

### Actuator

En esta sección, se introduce la clase Actuator, que hereda de MonoBehaviour. Esta clase se encarga de gestionar el estado del actuador y su posición inicial a través de la variable pública state y la referencia a actuatorStartPoint. El método ChangeState cambia el estado del actuador y, si es verdadero (true), inicia una coroutine ActuatorMove que mueve el actuador a lo largo del eje z hasta un valor de 16 y luego lo regresa a su posición inicial. Si es falso (false), detiene todas las coroutines y resetea la posición del actuador a (0, 0, 0). La coroutine ActuatorMove mueve suavemente el actuador usando yield return para crear un efecto de animación continua.

El Actuator en Unity es un script personalizado que maneja comportamientos específicos de un objeto de juego. Tiene propiedades como State, que puede indicar el estado actual del actuador (por ejemplo, activo o inactivo), y Actuator Start Point, que es un punto de referencia en el espacio de la aplicación. Este script es parte de un

sistema que controla aspectos como movimientos o interacciones del objeto (véase la figura 2.4)

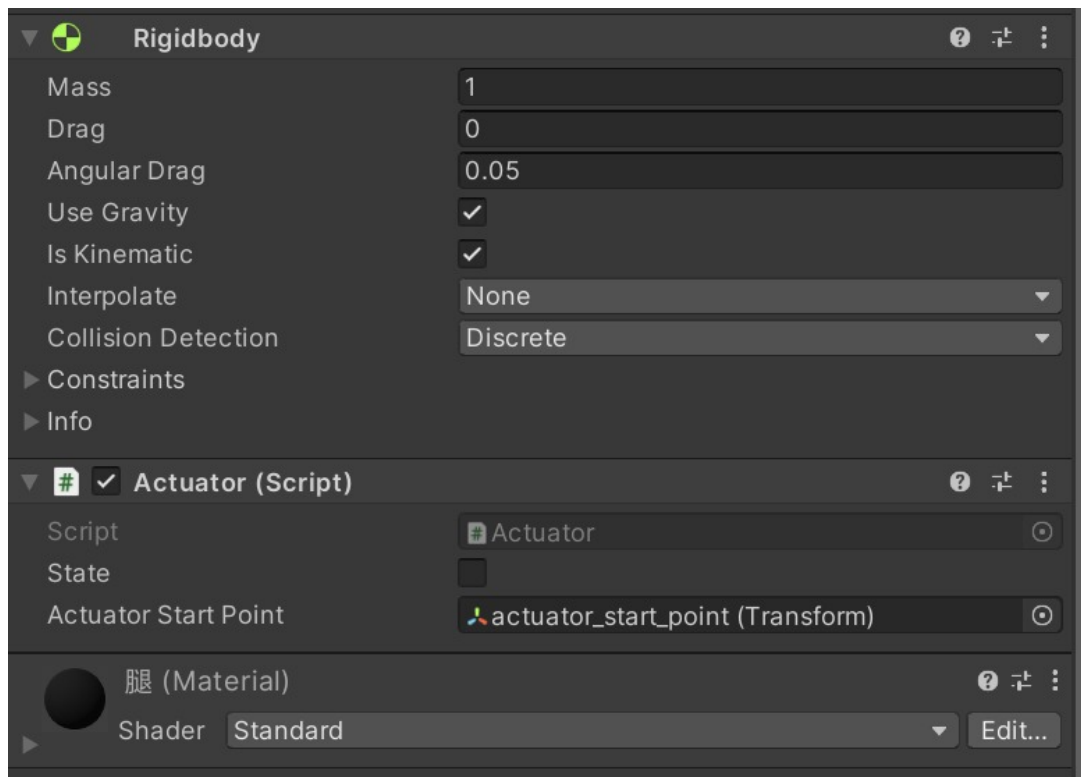


Figura 2.4: Actuator

## ButtonController

Aquí se presenta la clase `ButtonController`, que hereda de `MonoBehaviour`. Esta clase contiene referencias a `PLCConnection` y controladores de indicadores (`indicatorGreen` y `indicatorRed`), así como textos de visualización (`DisplayMetalText` y `DisplayNonMetalText`). También maneja los estados de los botones de inicio y parada, iniciando con `startButtonState` en falso y `stopButtonState` en verdadero. Los métodos `HandleStartButtonStateChanged` y `HandleStopButtonStateChanged` actualizan estos estados y los indicadores. La clase también escucha cambios en los conteos de metal y no metal, actualizando las visualizaciones correspondientes. La función `UpdateIndicatorLights` se asegura de que los indicadores reflejen correctamente los estados de los botones (véase la figura 2.5).



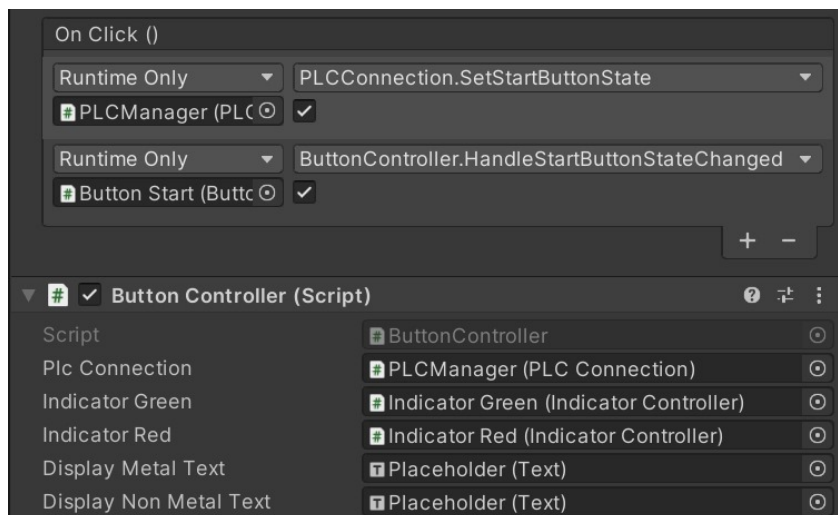


Figura 2.5: Button Controller

### CameraController

La clase `CameraController`, que hereda de `MonoBehaviour` dentro del espacio de nombres `FactorySimulator`, se describe en esta sección. Esta clase maneja la rotación de la cámara en los ejes de cabeceo, guiñada y balanceo, y permite el movimiento de la cámara en todas las direcciones y el zoom. Utiliza variables serializadas para ajustar las velocidades de giro, movimiento y zoom. En el método `Update`, la clase responde a las entradas del ratón para rotar la cámara. Cuando se mantiene presionado el botón derecho se puede realizar movimientos con el botón central y también aplicar zoom con la rueda de desplazamiento (véase la figura 2.6).

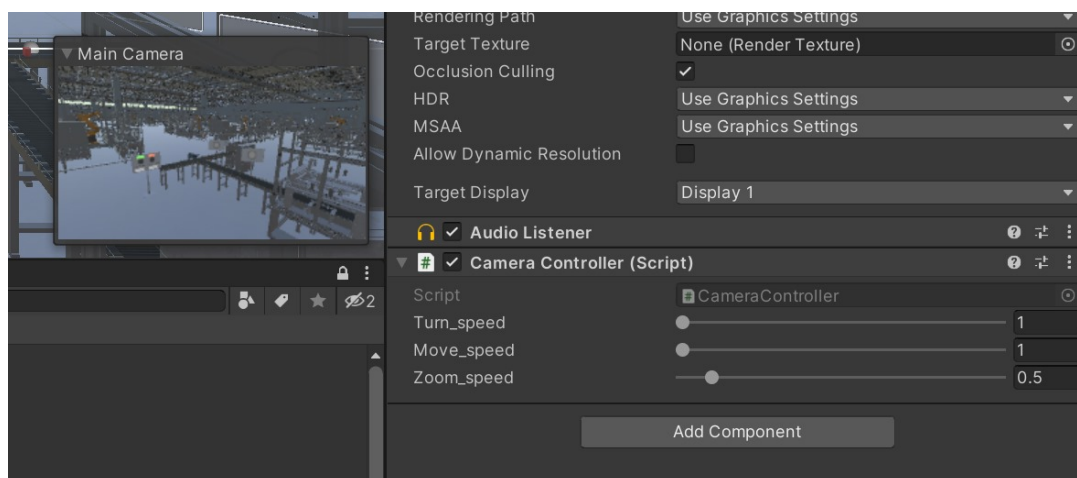


Figura 2.6: Camera Controller

## ConveyorBelt

En esta sección se explora la clase `ConveyorBelt`, que hereda de `MonoBehaviour` dentro del espacio de nombres `FactorySimulator`. La clase utiliza un array de `Transform` para definir el camino que seguirán los ítems y un `ConveyorBeltCollider` para detectar los ítems en la cinta. También permite configurar la velocidad constante y la velocidad de movimiento de los ítems. En el método `FixedUpdate`, la clase mueve cada ítem hacia el siguiente punto en el camino, ajustando la velocidad según sea constante o variable. El método privado `getClosestPathIndexToTransform` encuentra el índice del punto más cercano en el camino para cada ítem, asegurando un movimiento preciso a lo largo de la cinta (véase la figura 2.7).

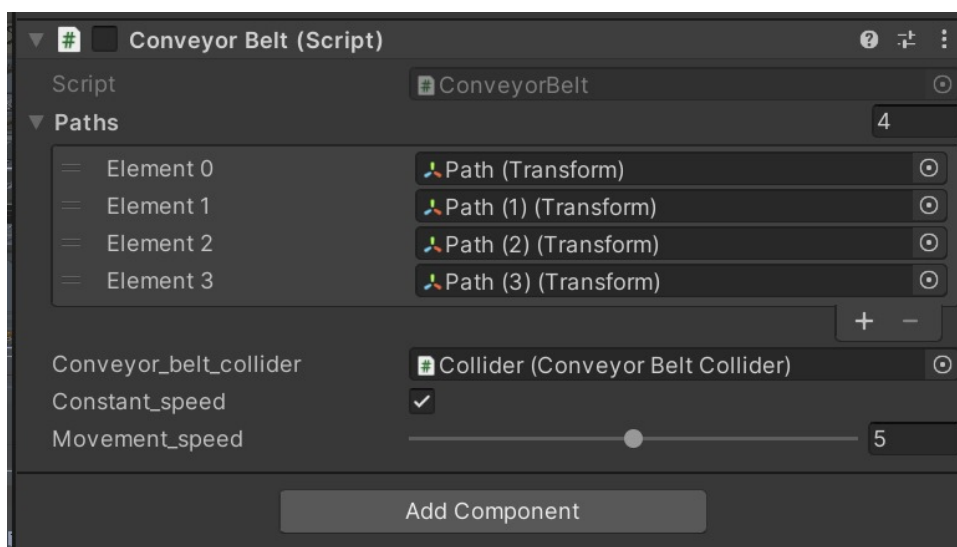


Figura 2.7: ConveyorBelt

## ConveyorBeltCollider

Se analiza la clase `ConveyorBeltCollider`, que hereda de `MonoBehaviour` dentro del espacio de nombres `FactorySimulator`. Esta clase mantiene una lista de ítems (`items`) que están en la cinta transportadora. Los métodos `OnTriggerEnter` y `OnTriggerExit` añaden o eliminan ítems de esta lista cuando entran o salen del collider, respectivamente. Además, se agrega una acción a cada ítem para removerlo de la lista antes de ser destruido, evitando referencias nulas. La función `getItems` devuelve la lista actual de ítems en la cinta. En la figura 2.8 se puede observar un

rectángulo verde que hace referencia al collider mencionado que sirve para que el objeto se mantenga sobre este.

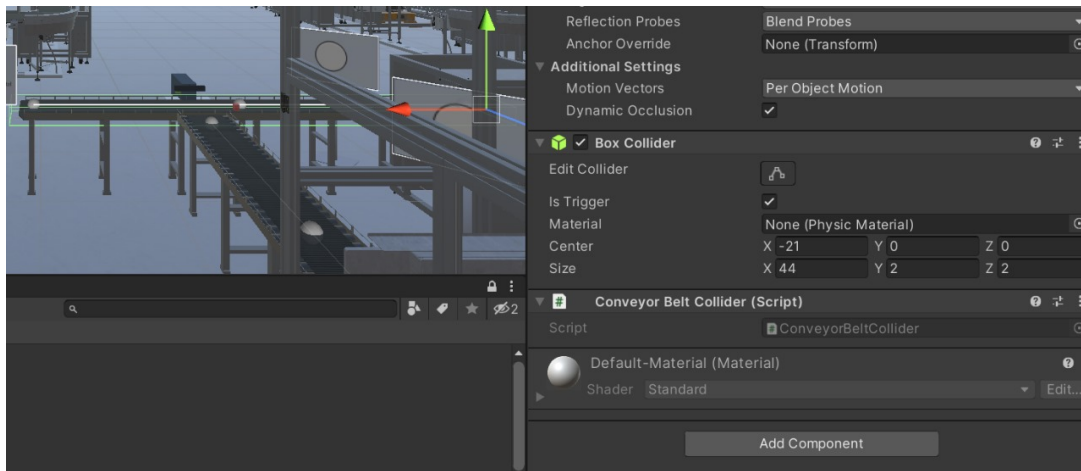


Figura 2.8: ConveyorBeltCollider

### ConveyorController

La clase `ConveyorController`, que hereda de `MonoBehaviour` dentro del espacio de nombres `FactorySimulator`, es el foco de esta sección. Esta clase controla una `ConveyorBelt` y se comunica con un `PLCConnection`. En el método `Start`, si `plcConnection` no es nulo, se suscribe al evento `OnStartButtonStateChanged` para controlar el estado de la cinta transportadora. La función `SetState` habilita o deshabilita la `ConveyorBelt` según el estado recibido. La función `ChangeSpeed` ajusta la velocidad de movimiento de la cinta, y `ChangeVal` permite cambiar esta velocidad con un valor entero sin signo. Los métodos `On` y `Off`, accesibles desde el menú de contexto, permiten habilitar o deshabilitar la cinta manualmente (véase la figura 2.9).

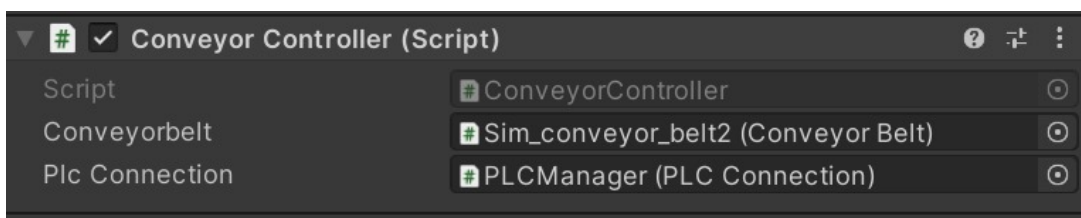


Figura 2.9: ConveyorController

## Destroyer

En este apartado se introduce la clase `Destroyer`, que hereda de `MonoBehaviour`. Esta clase utiliza un `Raycast` para detectar y eliminar objetos en una dirección específica. La variable pública `distance` determina la distancia del rayo, y `direction` especifica la dirección del mismo. En el método `FixedUpdate`, se dibuja un rayo desde la posición del objeto en la dirección dada y se verifica si colisiona con algún objeto dentro de la distancia especificada. Si el objeto colisionado tiene la misma etiqueta (`tag`) o la etiqueta "Metal", se destruye inmediatamente utilizando `DestroyImmediate`. En la figura 2.10 se puede observar un eje que hace referencia a la posición que tiene el mismo en el programa.

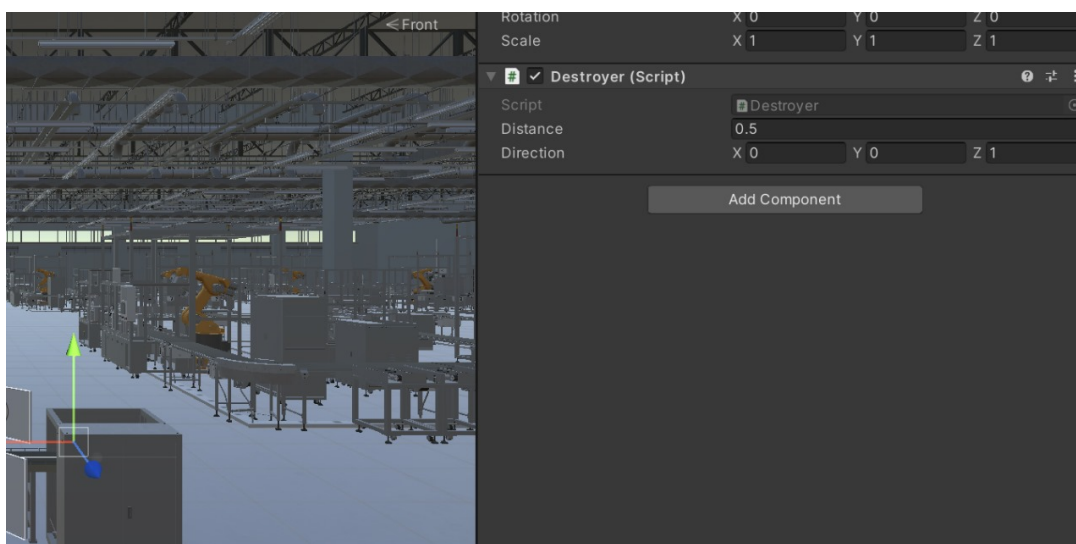


Figura 2.10: Destroyer

## Element

En esta sección, se examina la clase `Element`, que hereda de `MonoBehaviour`. La clase tiene los campos serializados: `rb`, `Rigidbody`, `pivot` y `Transform`. En el método `Start`, se inicializa `rb` obteniendo el componente `Rigidbody` adjunto al objeto. La clase también incluye un método `GetCoordinate` que devuelve la posición del `pivot` en el espacio de coordenadas del mundo.

## ElementSpawner

La clase `ElementSpawner`, que hereda de `MonoBehaviour`, se describe aquí. Esta clase contiene una referencia a un punto de generación (`spawnPoint`), un array de `Element` y una referencia al último elemento generado (`lastElement`). La variable `delta` determina la distancia mínima entre el `spawnPoint` y el `lastElement` para permitir la generación de un nuevo elemento. En el método `Update`, si la distancia entre el `spawnPoint` y el `lastElement` es mayor que `delta`, se llama al método `Create` para generar un nuevo elemento. El método `Create` instancia un nuevo `Element` aleatorio del array en el `spawnPoint` y resetea su posición local a `Vector3.zero`.

## GameObjectSensor

Se detalla la clase `GameObjectSensor`, que hereda de `MonoBehaviour`. La clase utiliza un `Raycast` para detectar objetos con una etiqueta específica (`sensorTag`, por defecto "Metal") en una dirección y distancia definidas. La clase contiene un evento `OnSensorStateChange` para notificar cambios en el estado del sensor y una referencia a `PLCConnection` para la comunicación con un PLC. En el método `Update`, si el `Raycast` detecta un objeto con la etiqueta correcta, el estado del sensor cambia a verdadero y se envía esta información al PLC. Si no se detecta ningún objeto, el estado del sensor cambia a falso. El método `SendSensorStateToPLC` envía el estado del sensor al PLC y `UpdateSensorStateFromPLC` actualiza el estado del sensor basado en el estado recibido del PLC (véase la figura 2.11).

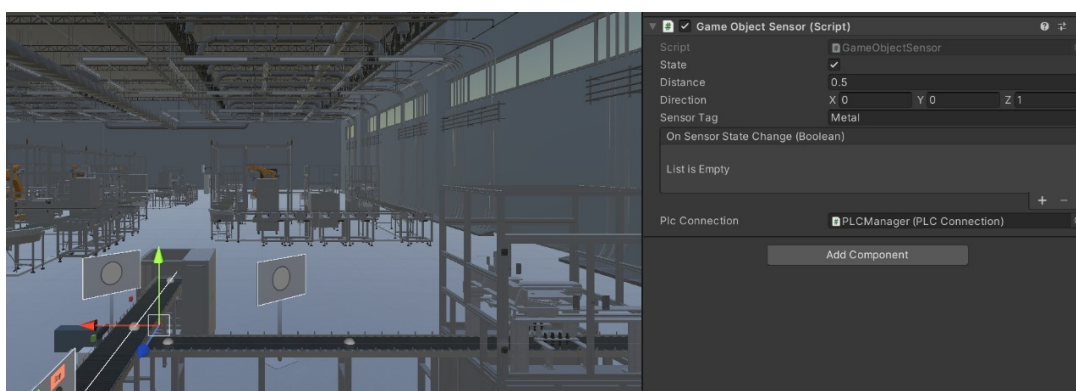


Figura 2.11: GaemObjectSensor

### GameObjectSensorInductive

Se describe la clase `GameObjectSensorInductive`, que hereda de `MonoBehaviour`. La clase utiliza un `Raycast` para detectar objetos metálicos en una dirección y distancia específicas. Tiene referencias a `PLCConnection` y `Actuator` para la comunicación y activación. En el método `Start`, se suscribe a los eventos de cambio de estado de la cinta transportadora y del actuador del `PLCConnection`. En el método `Update`, si el `Raycast` detecta un objeto metálico, cambia el estado del sensor a verdadero y envía esta información al PLC. Si no se detecta ningún objeto, el estado del sensor cambia a falso y desactiva el actuador. El método `SendSensorStateToPLC` envía el estado del sensor al PLC, `UpdateSensorStateFromPLC` actualiza el estado del sensor basado en el estado recibido del PLC, y `ActivateActuator` cambia el estado del actuador según el estado recibido (véase la figura 2.12).

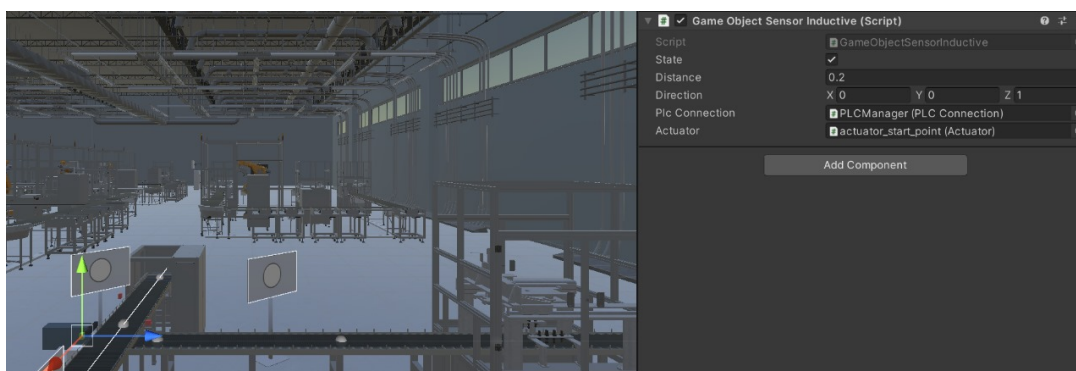


Figura 2.12: GameObjectSensorInductive

### GameObjectSensorNonMetal

En esta parte, se aborda la clase `GameObjectSensorNonMetal`, que hereda de `MonoBehaviour`. La clase utiliza un `Raycast` para detectar objetos en una dirección y distancia específicas. Contiene un evento `OnSensorStateChange` para notificar cambios en el estado del sensor y una referencia a `PLCConnection` para la comunicación con un PLC. En el método `Start`, si `plcConnection` no es nulo, se suscribe al evento `OnConveyorStateChanged` para actualizar el estado del sensor desde el PLC. En el método `Update`, si el `Raycast` detecta un objeto, el estado del sensor cambia a verdadero, se envía esta información al PLC y se invoca el evento. Si no se detecta ningún objeto, el estado del sensor cambia a falso y se invoca el

evento. El método `SendSensorStateToPLC` envía el estado del sensor al PLC y `UpdateSensorStateFromPLC` actualiza el estado del sensor basado en el estado recibido del PLC. En la figura 2.13 se puede observar el posicionamiento de dicho sensor.

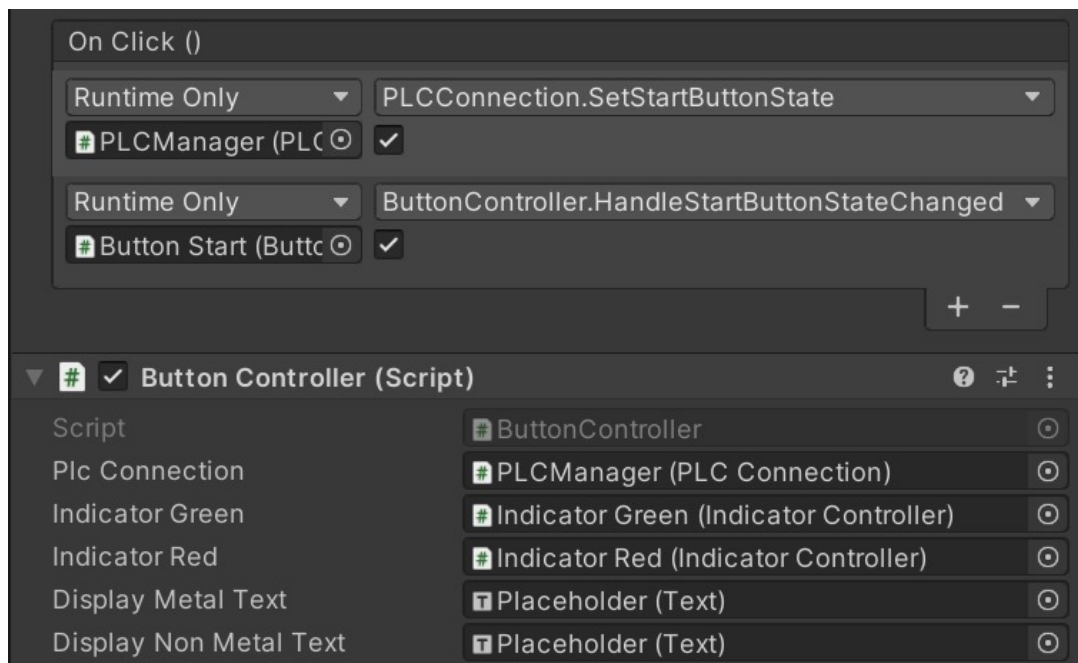


Figura 2.13: GameObjectSensorNonMetal

## IndicatorController

La clase `IndicatorController`, que hereda de `MonoBehaviour`, es el tema de esta sección. La clase contiene una referencia a una imagen (`indicatorImage`) y un método `SetColor` que establece el color de dicha imagen. Si `indicatorImage` no es nulo, el método `SetColor` cambia el color de la imagen al color especificado (véase la figura 2.14).

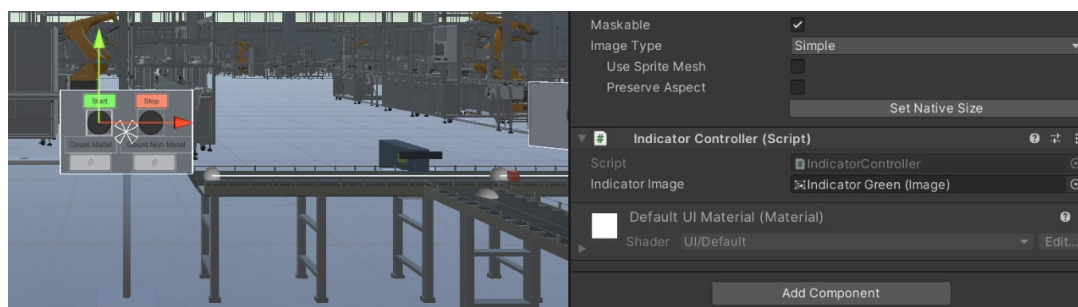


Figura 2.14: IndicatorController

### **IndicatorControllerSensorMetal**

Se examina aquí la clase `IndicatorControllerSensorMetal`, que hereda de `MonoBehaviour`. La clase contiene una referencia a una imagen (`indicatorImage`) y a un sensor (`GameObjectSensor`). En el método `Start`, si el sensor no es nulo, se suscribe al evento `OnSensorStateChange` del sensor. El método `HandleSensorStateChange` cambia el color de la imagen a amarillo cuando el estado del sensor es verdadero y a gris cuando es falso. El método `SetColor` establece el color de la imagen si `indicatorImage` no es nulo.

### **IndicatorControllerSensorNonMetal**

En esta sección se describe la clase `IndicatorControllerSensorNonMetal`, que hereda de `MonoBehaviour`. La clase contiene una referencia a una imagen (`indicatorImage`) y a un sensor (`GameObjectSensorNonMetal`). En el método `Start`, si el sensor no es nulo, se suscribe al evento `OnSensorStateChange` del sensor. El método `HandleSensorStateChange` cambia el color de la imagen a amarillo cuando el estado del sensor es verdadero y a gris cuando es falso. El método `SetColor` establece el color de la imagen si `indicatorImage` no es nulo.

### **Item**

La clase `Item`, que hereda de `MonoBehaviour` dentro del espacio de nombres `FactorySimulator`, se introduce aquí. Esta clase requiere componentes `Rigidbody` y `BoxCollider`. Contiene un array de `GameObject` llamado `item_stages` que representa las diferentes etapas del ítem durante el proceso de transformación. La propiedad `faulty` indica si el ítem es defectuoso, y un `GameObject` `missing_item` representa una parte que faltará en los ítems defectuosos. La clase incluye una acción `action_before_destroyed` que se invoca antes de destruir el ítem. En el método `Update`, se actualizan los estados de los ítems y su condición de defectuosos. El método `incrementItemStage` avanza el ítem a la siguiente etapa. Al destruir el ítem, se invoca la acción `action_before_destroyed` si está definida.



## Paths

La clase *Paths*, que hereda de *MonoBehaviour* dentro del espacio de nombres *FactorySimulator*, se describe en esta sección. La clase contiene el método *OnDrawGizmos*, que se utiliza para dibujar gizmos en el editor de Unity. Este método obtiene todos los componentes *Transform* hijos del objeto y dibuja esferas en sus posiciones. Además, dibuja líneas conectando cada *Transform* hijo con el siguiente, visualizando el camino que seguirán los objetos dentro del entorno (véase la figura 2.15).

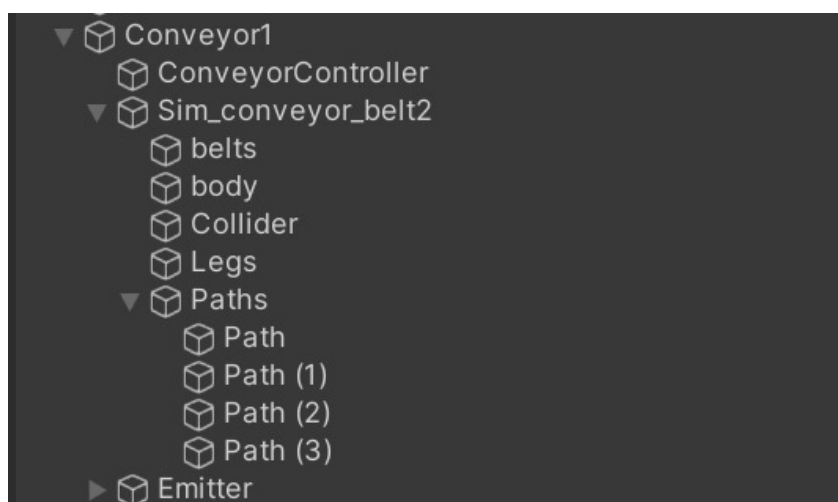


Figura 2.15: Paths.

En el proyecto, la jerarquía de la escena muestra un objeto llamado *Paths* dentro de la estructura de *Sim\_conveyor\_belt2*, como se puede ver en la figura 2.15. Este objeto *Paths* contiene varios componentes hijos, cada uno etiquetado como *Path* con números consecutivos, como *Path (1)*, *Path (2)*, y *Path (3)*. Estos componentes representan puntos específicos en el espacio que delimitan el recorrido que un objeto seguirá en el entorno de la simulación. La clase *Paths* probablemente usa estos componentes para definir visualmente las rutas, utilizando el método *OnDrawGizmos* para dibujar esferas en las posiciones de cada *Path* y líneas conectando estos puntos. Esto facilita la visualización del trayecto en el editor de Unity, ayudando en el diseño y depuración del sistema de transporte simulado, como una cinta transportadora o similar.

# Capítulo 3

## Interfaz de comunicación y programación del PLC

Esta interfaz sirve como el punto de conexión entre el entorno de desarrollo de Unity 3D y el PLC, permitiendo la transferencia de datos y comandos para controlar y monitorear el proceso virtual desde el PLC y viceversa. Su diseño y funcionalidad son clave para garantizar una comunicación efectiva y una programación eficiente del PLC, optimizando así la automatización de los procesos virtuales desarrollados en Unity 3D.

### 3.1. Introducción al PLC SIMATIC S7-1500

El Controlador Lógico Programable (PLC) es un componente esencial en los sistemas de automatización industrial, utilizado para controlar procesos de producción, máquinas y equipos, así como para la adquisición y procesamiento de datos [18]. Su alta precisión, eficiencia, programabilidad y estabilidad lo convierten en una herramienta indispensable para lograr procesos industriales más eficientes y seguros.

El SIMATIC S7-1500 es una línea de PLCs diseñada para aplicaciones industriales complejas. Su arquitectura modular y escalable permite el control de una amplia gama de dispositivos, desde maquinaria simple hasta grandes sistemas de producción [19]. Este PLC se distingue por su alta velocidad de procesamiento,

interfaces avanzadas y capacidades de diagnóstico integrado, lo que facilita la configuración, monitoreo y resolución de problemas [20].

Se utilizó el PLC SIMATIC S7-1500 debido a su tecnología avanzada y sus capacidades superiores en el control de sistemas industriales modernos. Este controlador destaca por su alta velocidad de procesamiento y su arquitectura modular, que permite una integración flexible y escalable con diversos componentes y sistemas, incluye entradas y salidas digitales y analógicas, módulos de comunicación y tecnología avanzada, permitiendo su adaptación a diversas necesidades de automatización.

## 3.2. Configuración del proyecto e interfaz de comunicación.

### 3.2.1. Creación del proyecto:

1. **Inicio del Proyecto:** Ingresar a TIA Portal y seleccionar la opción crear proyecto (véase la figura 3.1).

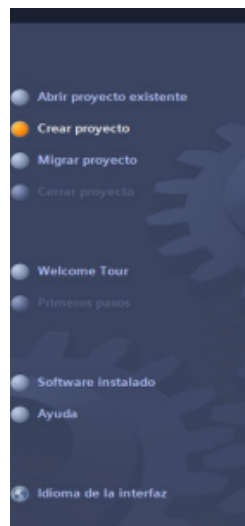


Figura 3.1: Creación de un proyecto en TIA Portal.

2. **Detalles del Proyecto:** Especificar el nombre y la ruta donde se guardará el proyecto creado (véase la figura 3.2).

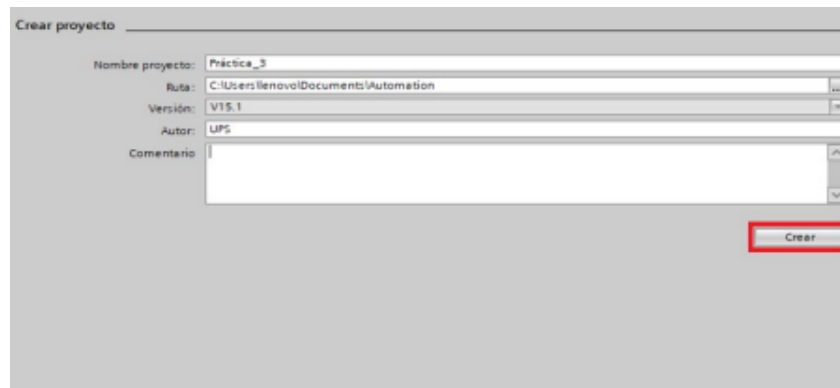


Figura 3.2: Detalles para la creación de un proyecto.

3. **Interfaz Inicial:** Seleccionar el botón “Crear”, se mostrará automáticamente la ventana “Primeros Pasos”. Aquí, seleccionar “Configurar un dispositivo” (véase la figura 3.3).

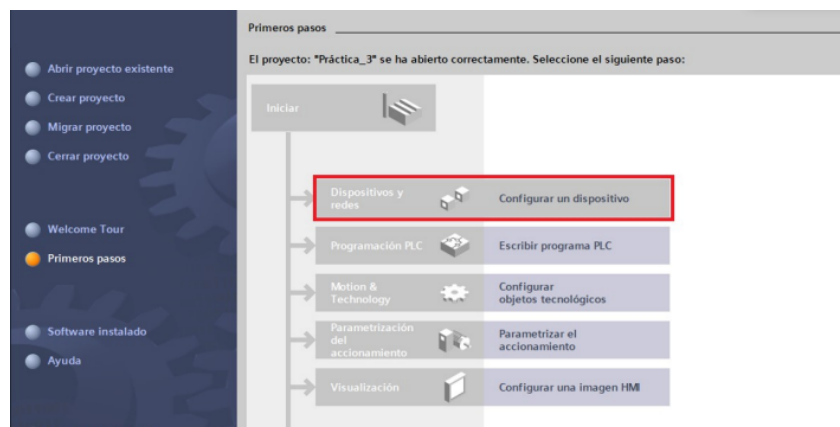


Figura 3.3: Interfaz Inicial de TIA Portal.

4. **Adición de Dispositivos:** En la ventana “Dispositivos y Redes”, se debe hacer clic en “Agregar Dispositivo”. Elegir “Controladores”, luego “SIMATIC S7-1500”, seguido de “CPU”, y seleccionar el modelo del CPU y la versión del firmware correspondiente. En este caso se adicionará un CPU 1516-3 PN/DP con el código 6ES7 516-3AN01-0AB0, versión V2.6. Por defecto se agregará un PLC con el nombre “PLC\_1” (véase la figura 3.4).

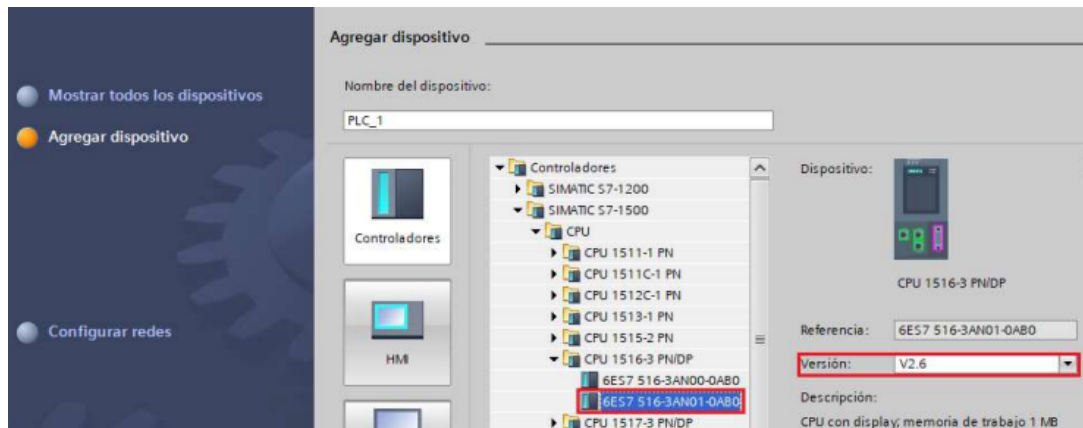


Figura 3.4: Adición de Dispositivos.

5. **Visualización de Redes:** Tras agregar el dispositivo, se abrirá la ventana “Vista de Dispositivos” mostrando el dispositivo añadido (véase la figura 3.5).

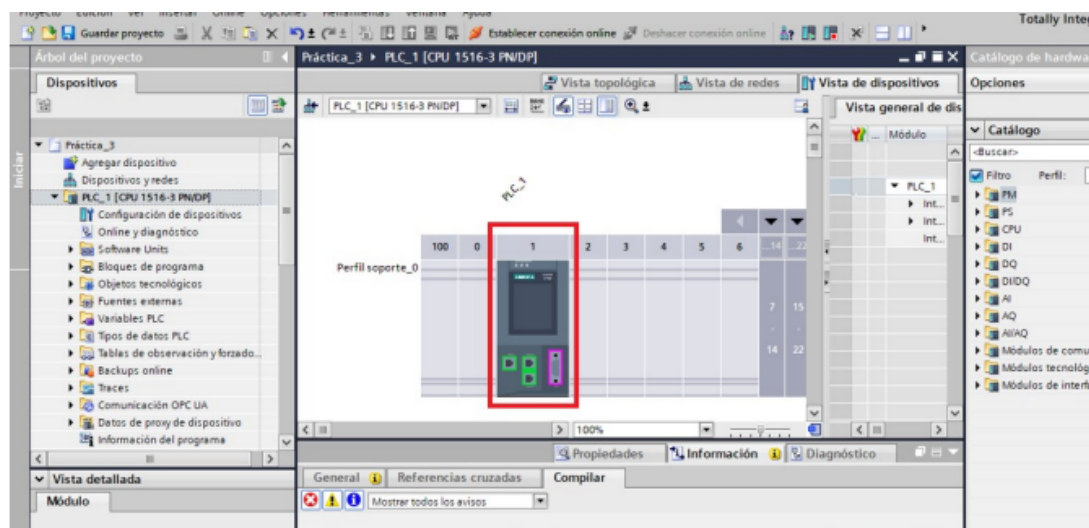


Figura 3.5: Vista de dispositivos en TIA Portal.

6. **Configuración de Módulos de Potencia:** En la ventana del proyecto, dirigirse a “Catálogo de Hardware” y seleccionar la fuente de alimentación. Este caso se añadirá una fuente de alimentación con la ruta “PM” / “PM 190W 120/230VAC” / “6EP1333-4BA00”. Un doble clic sobre la fuente añadirá el módulo automáticamente (véase la figura 3.6).

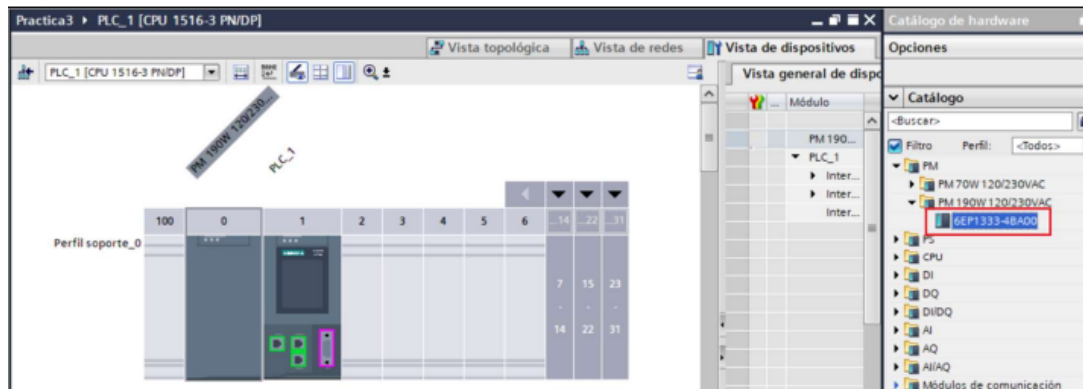


Figura 3.6: Configuración de Módulos de Potencia.

7. **Incorporación de Módulos de Entradas Analógicas:** Si bien, el proyecto no utiliza entradas y salidas analógicas, el PLC físico que se utilizará para el proyecto contiene este tipo de entradas y salidas. Para evitar conflictos con diferencias en el hardware se agregan los módulos correspondientes. En el “Catálogo de Hardware”, localizar el módulo “AI” / “8xU/I/RTD/TC ST” / “6ES7 531-7KF00-0AB0”, escoger la versión “V2.2”, y mediante un doble clic, añadir el módulo automáticamente (véase la figura 3.7).

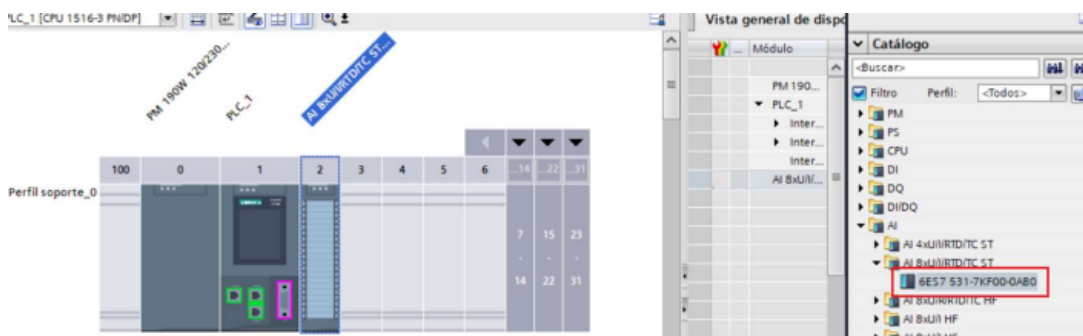


Figura 3.7: Incorporación de Módulos de Entradas Analógicas.

8. **Incorporación de Módulos de Salidas Analógicas:** En el “Catálogo de Hardware”, buscar el módulo “AQ” / “AQ 4xU/I ST” / “6ES7 532-5HD00-0AB0”, seleccionar la versión “V2.2”, y con un doble clic, incorporar el módulo (véase la figura 3.8).



Figura 3.8: Incorporación de Módulos de Salidas Analógicas.

9. **Adición de Módulos de Entradas Digitales:** Localizar el módulo “DI” / “DI 32x24VDC HF” / “6ES7 521-1BL00-0AB0”, elegir la versión “V2.1” en el “Catálogo de Hardware”, y mediante doble clic, añadir el módulo (véase la figura 3.9).

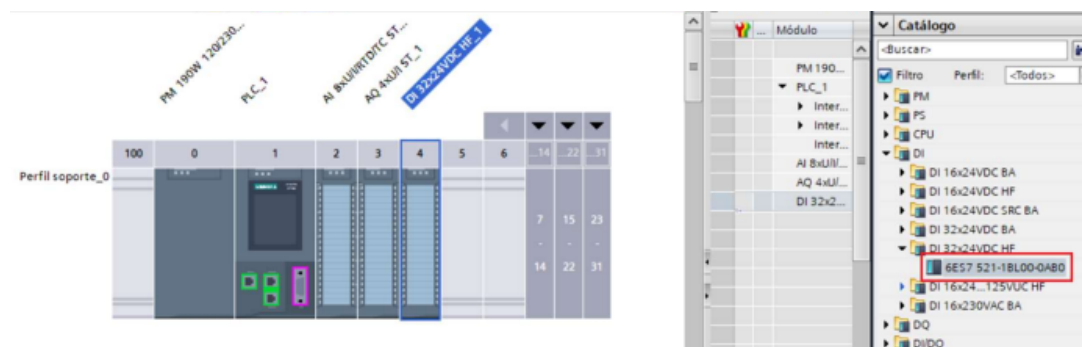


Figura 3.9: Adición de Módulos de Entradas Digitales.

10. **Configuración de Módulos de Salidas Digitales:** Dirigirse a “Catálogo de Hardware”, seleccionar el módulo “DQ” / “DQ 32x24VDC/0.5A HF” / “6ES7 522-1BL01-0AB0”, optar por la versión “V1.1”, y con un doble clic, finalizar la adición del módulo (véase la figura 3.10).

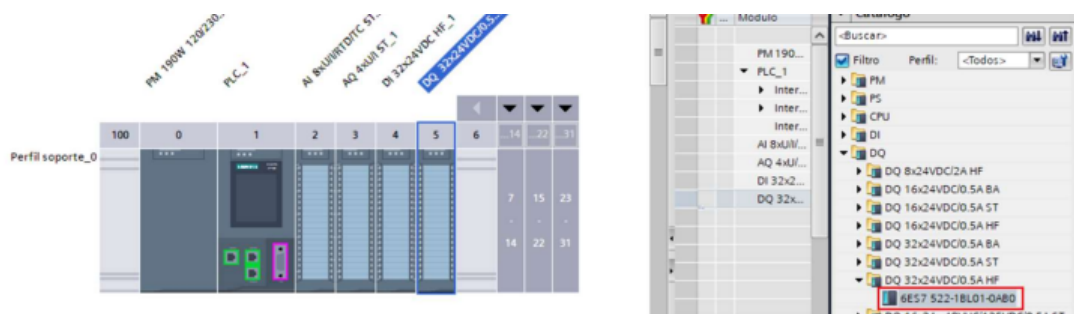


Figura 3.10: Configuración de Módulos de Salidas Digitales.

Una vez agregado todo el hardware, el sistema de automatización debe verse como se ilustra en la figura 3.11

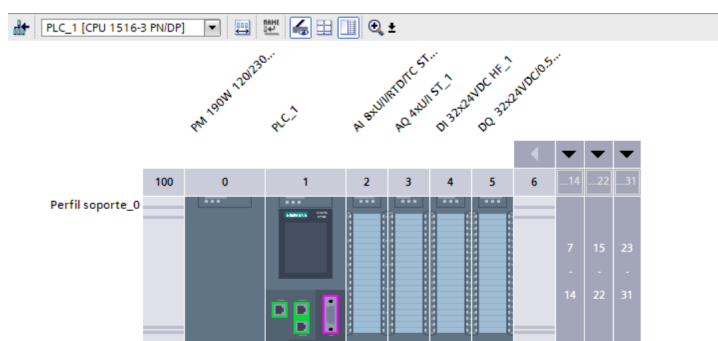


Figura 3.11: Configuración inicial del PLC.

### 3.2.2. Configuración de la Interfaz de Comunicación:

1. Configuración de Protección y Seguridad del PLC: Acceder a las propiedades del PLC en la sección de “Protección y Seguridad” y seleccionar la opción de Acceso completo (sin protección) en “Nivel de acceso” (véase en la figura 3.12).

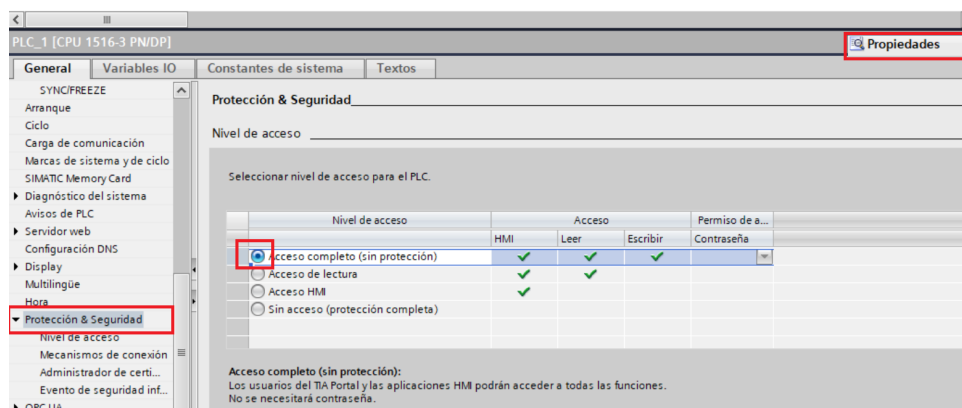


Figura 3.12: Configuración de nivel de acceso en Protección y Seguridad.

2. Configuración de Mecanismos de Conexión: Seleccionar la opción de “Permitir acceso completo a todos los recursos de la red” dentro de Mecanismos de conexión (véase en la figura 3.13).



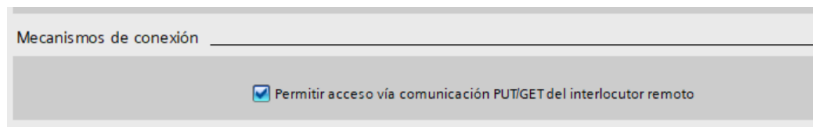


Figura 3.13: Configuración de mecanismos de conexión.

3. Creación de un Bloque de Datos: Crear un bloque de datos para el envío y recepción de datos (véase en la figura 3.14).

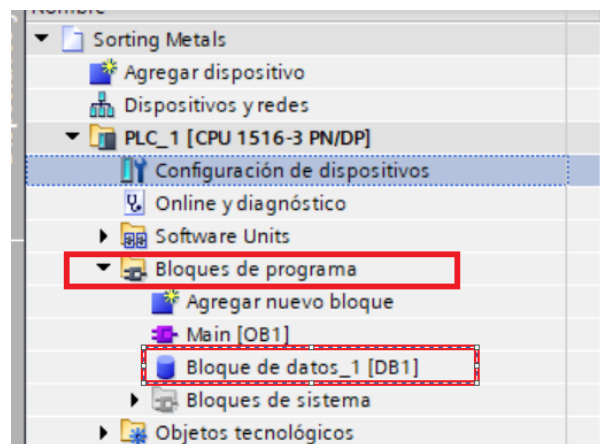


Figura 3.14: Creación de un bloque de datos.

4. Propiedades del Bloque de Datos: Seleccionar la opción de propiedades del bloque de datos (véase en la figura 3.15).

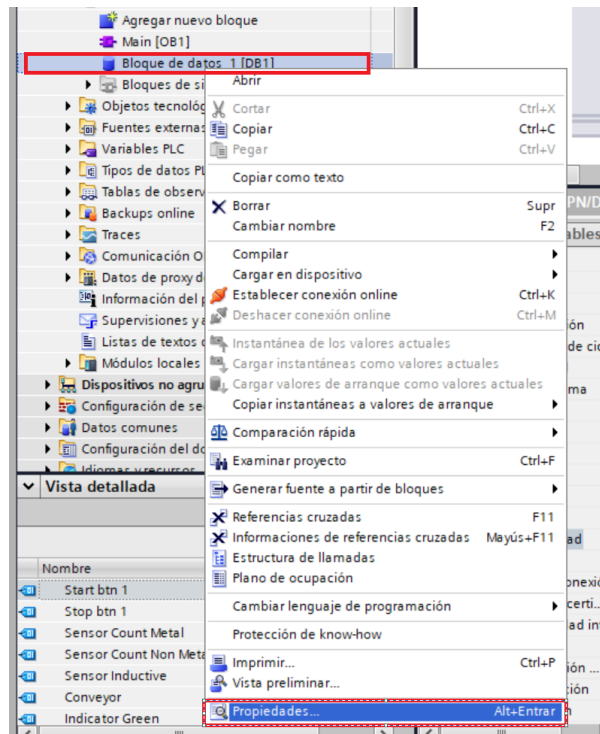


Figura 3.15: Propiedades del bloque de datos.

5. Deshabilitar Acceso Optimizado al Bloque: Seleccionar la opción de “Acceso optimizado al bloque” para deshabilitarlo (véase en la figura 3.16).

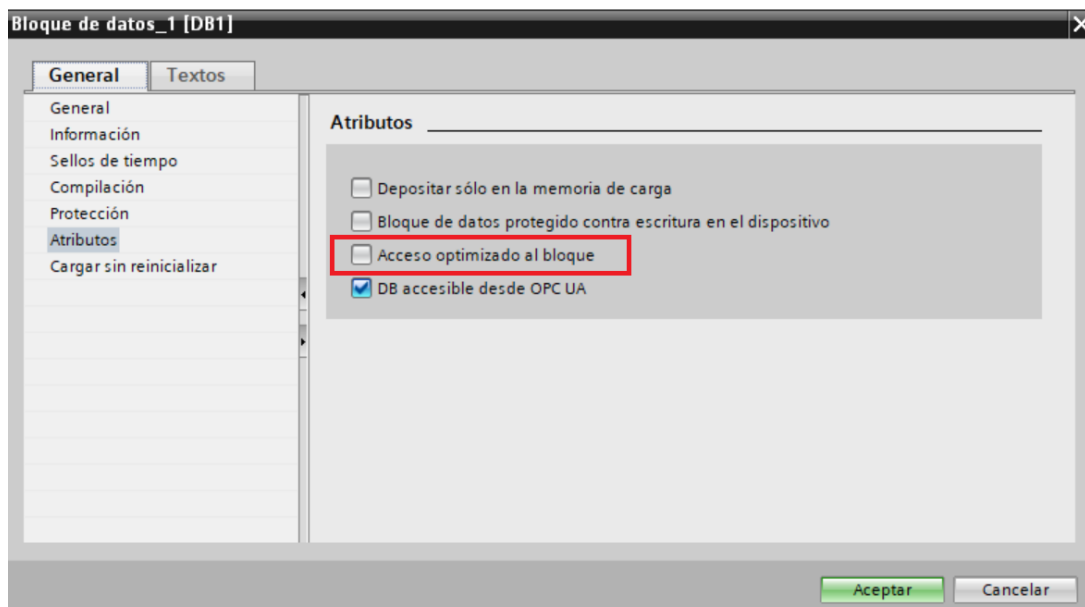


Figura 3.16: Deshabilitar acceso optimizado al bloque.

### 3.3. Programación del PLC en TIA PORTAL

#### 3.3.1. Segmento 1

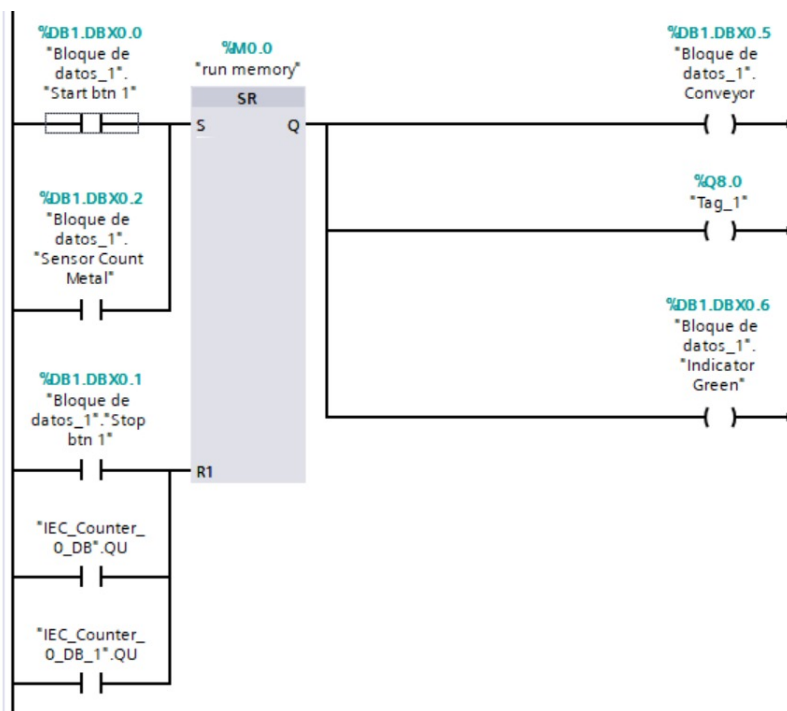


Figura 3.17: Imagen del Bloque 1.

#### Funcionamiento

La figura 3.17 representa un sistema de control básico. Al presionar el botón de inicio (%DB1.DBX0.0) o cuando el sensor de conteo metálico (%DB1.DBX0.2) detecte una pieza se pone en marcha el programa. La memoria de ejecución (%M0.0) se activa y mantiene el sistema en funcionamiento. El botón de parada (%DB1.DBX0.1) puede ser presionado en cualquier momento para detener el sistema. Mientras el sistema está en funcionamiento, la cinta transportadora (%DB1.DBX0.5) se activa para mover las piezas a lo largo del sistema. El indicador verde (%DB1.DBX0.6) se enciende para mostrar que el sistema está operando correctamente. Además, se utilizan contadores (IEC\_Counter\_0\_DB.QU) para realizar el seguimiento de eventos específicos dentro del sistema, asegurando un control preciso y eficiente.

### 3.3.2. Segmento 2

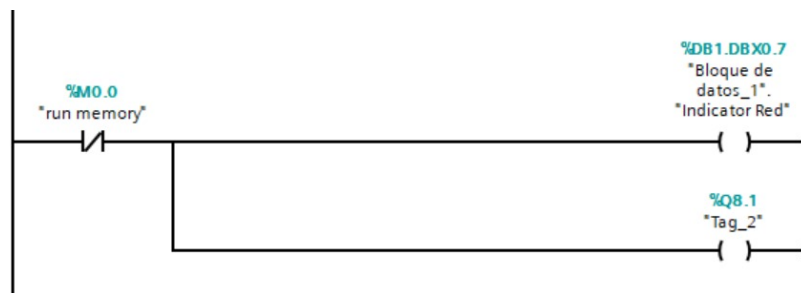


Figura 3.18: Imagen del Bloque 2.

#### Funcionamiento

El diagrama de escalera mostrado en la figura 3.18 representa una parte del sistema de control donde se utilizan indicadores luminosos y etiquetas para mostrar el estado del sistema. Cuando el registro de memoria (%M0.0) “run memory” está activado, las siguientes acciones ocurren:

- La salida (%DB1.DBX0.7) “Indicator Red” se activa, encendiendo un indicador rojo. Esto muestra que el sistema está detenido.
- La salida (%Q8.1) “Tag\_2” también se activa, para poder observar su funcionamiento en el PLC real.

Este bloque de programa asegura que cuando el sistema está en funcionamiento (%M0.0 activado), se proporciona una indicación visual mediante el indicador rojo y se actualiza el estado a través de la etiqueta (%Q8.1).

### 3.3.3. Segmento 3

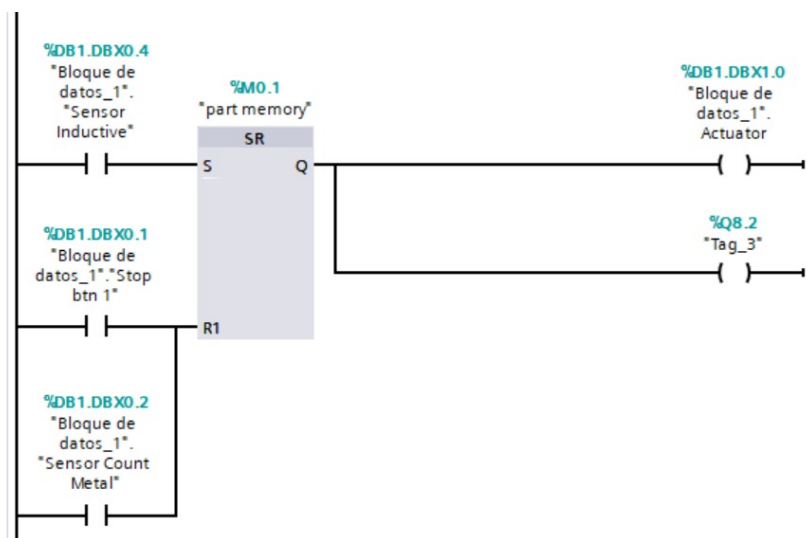


Figura 3.19: Imagen del Bloque 3.

#### Funcionamiento:

En la figura 3.19 el circuito inicia cuando alguna de las entradas (%DB1.DBX0.4) "Sensor Inductivo" o %DB1.DBX0.2 "Sensor Contador de Metales") se activan. Si el sensor inductivo detecta un objeto metálico (%DB1.DBX0.4) o el sensor de conteo de metales (%DB1.DBX0.2) cuenta una pieza metálica, y el botón de parada (%DB1.DBX0.1) no está presionado, entonces la memoria de ejecución (%M0.1) "part memory" se activa (bit S del bloque SR).

Cuando la memoria (%M0.1) está activada, se habilitan dos salidas:

- (%DB1.DBX1.0) "Actuador": Se activa, lo que podría significar que el actuador comienza a trabajar.
- (%Q8.2) "Tag\_3": Se activa, indicando el estado específico del sistema.

### 3.3.4. Segmento 4

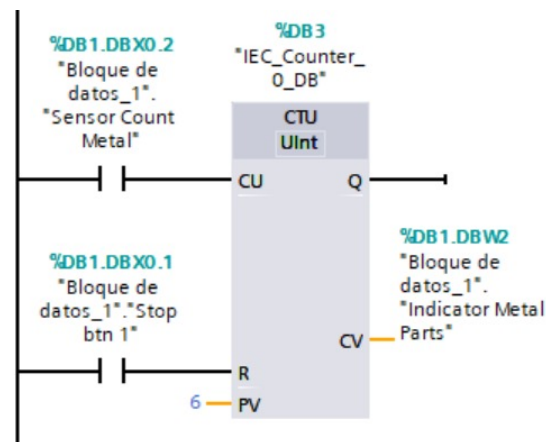


Figura 3.20: Imagen del Bloque 4.

#### Funcionamiento

El bloque de contador ascendente (CTU) en el programa de PLC se utiliza para contar el número de piezas metálicas detectadas por un sensor. Este contador incrementa su valor cada vez que el sensor detecta una pieza, representado por la activación del bit (%DB1.DBX0.2). Adicionalmente, el contador puede reiniciarse a través de un botón de parada, asociado al bit (%DB1.DBX0.1), que pone el contador a cero. El contador tiene un valor de preajuste (PV) configurado en 6; cuando el valor actual del contador (CV) alcanza este preajuste, se activa la salida Q, reflejada en (%DB1.DBW2), indicando que se han contado seis piezas metálicas. Este sistema permite un conteo preciso y la posibilidad de reiniciar el proceso en cualquier momento, facilitando el control y monitoreo de las piezas metálicas en el proceso de producción (véase en la figura 3.20).

## 3.3.5. Segmento 5

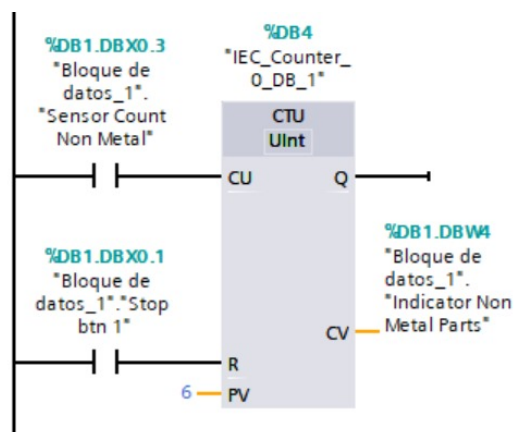


Figura 3.21: Imagen del Bloque 5.

**Funcionamiento**

El bloque de contador ascendente (CTU) en el programa de PLC se utiliza para contar el número de piezas no metálicas detectadas por un sensor. Este contador incrementa su valor cada vez que el sensor detecta una pieza, representado por la activación del bit (%DB1.DBX0.3). Además, el contador puede reiniciarse mediante un botón de parada, asociado al bit (%DB1.DBX0.1), que restablece el contador a cero. El contador tiene un valor de preajuste (PV) configurado en 6; cuando el valor actual del contador (CV) alcanza este preajuste, se activa la salida Q, reflejada en (%DB1.DBW4), indicando que se han contado seis piezas no metálicas. Este sistema permite un conteo preciso y la posibilidad de reiniciar el proceso en cualquier momento, facilitando el control y monitoreo de las piezas no metálicas en el proceso de producción (véase en la figura 3.21).

### 3.3.6. Segmento 6

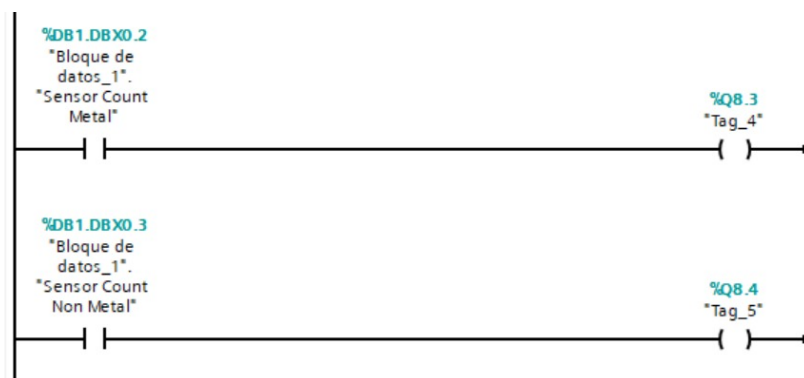


Figura 3.22: Imagen del Bloque 6.

#### Funcionamiento:

El esquema mostrado representa dos líneas de programación en un PLC utilizadas para activar salidas en función de la detección de piezas metálicas y no metálicas. La primera línea activa la salida (%Q8.3) cuando el sensor detecta una pieza metálica, representada por la activación del bit (%DB1.DBX0.2). La segunda línea activa la salida (%Q8.4) cuando el sensor detecta una pieza no metálica, representada por la activación del bit (%DB1.DBX0.3). Estas salidas son utilizadas para poder visualizar en el PLC real (véase en la figura 3.22).

## 3.4. Interfaz de Comunicación con el Proceso Virtual

En el siguiente apartado se detallará la integración de los códigos necesarios para establecer una comunicación efectiva entre Unity y el PLC. A continuación, se presenta un desglose ordenado de las secciones del código que deben ser implementadas secuencialmente para configurar la conexión y la interacción con el PLC.



## Declaraciones de Espacio de Nombres y Clase

```
using UnityEngine;
using Sharp7;
using System;

Script de Unity (1 referencia de recurso) | 5 referencias
public class PLCConnection : MonoBehaviour
{
    public event Action<bool> OnConveyorStateChanged;
    public event Action<bool> OnStartButtonStateChanged;
    public event Action<bool> OnStopButtonStateChanged;
    public event Action<int> OnMetalCountChanged;
    public event Action<int> OnNonMetalCountChanged;
    public event Action<bool> OnActuatorStateChanged;
}
```

Figura 3.23: Configuración de Módulos de Potencia.

En la figura 3.23 se define lo siguiente:

- **Using Directives:** Importa las bibliotecas necesarias para trabajar con Unity, y Sharp7, una librería para comunicación con el PLC.
- **Public Class PLCConnection: MonoBehaviour:** Define una clase llamada PLCConnection que hereda de MonoBehaviour, lo que permite que se adjunte a un objeto en Unity.
- **public event Action<...>:** Declara varios eventos que se activarán cuando los estados del PLC cambien.

## Variables de Clase

```
private S7Client client;
private byte[] buffer = new byte[6]; // Ajusta el tamaño del buffer para leer todos los datos necesarios
private string plcIpAddress = "192.168.0.1";
private int rack = 0;
private int slot = 1;
private float updateInterval = 0.1f;
private float nextUpdateTime = 0f;

private bool conveyor;
private bool startButton;
private bool stopButton = true; // Inicia en verdadero
private bool sensorCountMetal;
private bool sensorCountNonMetal;
private bool actuador = false; // Variable para el estado del actuador
private int indicatorMetalParts;
private int indicatorNonMetalParts;
```

Figura 3.24: Configuración de Variables de Clase.

En la figura 3.24 se define lo siguiente:

- **S7Client client:** Cliente para comunicarse con el PLC.
- **byte[] buffer:** Buffer para leer y escribir datos.
- **string plcIpAddress:** Dirección IP del PLC.
- **int rack** y **int slot:** Ubicación del PLC en el rack.
- **float updateInterval** y **float nextUpdateTime:** Intervalo de actualización y tiempo para la próxima actualización, Variables booleanas y enteras para los estados de diferentes componentes del PLC.

## Método Start y Métodos de Inicialización

```
void Start()
{
    client = new S7Client();
    ConnectToPLC();

    // Establecer el estado inicial del botón de parada en verdadero
    Invoke("InitializeStopButtonState", 1.0f); // Retraso para asegurar la conexión antes de establecer el estado

    // Inicializar el estado del actuador en false
    Invoke("InitializeActuatorState", 1.0f); // Retraso para asegurar la conexión antes de establecer el estado
}

0 referencias
void InitializeStopButtonState()
{
    SetStopButtonState(true);
}

0 referencias
void InitializeActuatorState()
{
    SetActuatorState(false);
}
```

Figura 3.25: Métodos de Inicialización.

En la figura 3.25 se define lo siguiente:

- **void Start():** Método de Unity que se ejecuta al iniciar el script, crea una nueva instancia de `S7Client`, llama al método `ConnectToPLC` para establecer la conexión con el PLC, utiliza `Invoke` para llamar a métodos de inicialización con un retraso de 1 segundo. `InitializeStopButtonState()` y `InitializeActuatorState()`: Métodos para establecer los estados iniciales del botón de parada y el actuador.

## Método Update y Conexión al PLC

```

Mensaje de Unity | 0 referencias
void Update()
{
    if (Time.time >= nextUpdateTime)
    {
        nextUpdateTime = Time.time + updateInterval;
        ReadPLCData();
    }
}

3 referencias
void ConnectToPLC()
{
    int connectionResult = client.ConnectTo(plcIpAddress, rack, slot);
    if (connectionResult == 0)
    {
        Debug.Log("Connection ok");
    }
    else
    {
        Debug.LogError("Connection error: " + client.ErrorText(connectionResult));
        Invoke("ReconnectToPLC", 5);
    }
}

0 referencias
void ReconnectToPLC()
{
    ConnectToPLC();
}

```

Figura 3.26: Método Update y Conexión al PLC.

En la figura 3.26 se define lo siguiente:

- **void Update():** Método de Unity que se llama una vez por cuadro. Comprueba si es tiempo de la próxima actualización y, si es así, llama a ReadPLCData.
- **ConnectToPLC():** Establece la conexión con el PLC. Si la conexión es exitosa, registra un mensaje de éxito. Si falla, registra un mensaje de error y programa un intento de reconexión en 5 segundos.
- **ReconnectToPLC():** Llama de nuevo a ConnectToPLC para intentar reconectar.

## Lectura de Datos del PLC

```
void ReadPLCData()
{
    if (client != null && client.Connected)
    {
        int readResult = client.DBRead(1, 0, buffer.Length, buffer);
        if (readResult == 0)
        {
            Debug.Log("DB1 Read ok");

            bool newStartButtonState = S7.GetBitAt(buffer, 0, 0); // Offset 0.0, bit 0
            if (startButton != newStartButtonState)
            {
                startButton = newStartButtonState;
                OnStartButtonStateChanged?.Invoke(startButton);
            }

            bool newStopButtonState = S7.GetBitAt(buffer, 0, 1); // Offset 0.1, bit 1
            if (stopButton != newStopButtonState)
            {
                stopButton = newStopButtonState;
                OnStopButtonStateChanged?.Invoke(stopButton);
            }

            bool newSensorInductiveState = S7.GetBitAt(buffer, 0, 4); // Offset 0.4, bit 4
            if (conveyor != newSensorInductiveState)
            {
                conveyor = newSensorInductiveState;
                OnConveyorStateChanged?.Invoke(conveyor);
            }

            bool newActuatorState = S7.GetBitAt(buffer, 1, 0); // Offset 1.0, bit 0
            if (actuator != newActuatorState)
            {
                actuator = newActuatorState;
                OnActuatorStateChanged?.Invoke(actuator);
            }
        }
    }
}
```

Figura 3.27: Lectura de Datos del PLC.

```
        int newIndicatorMetalParts = S7.GetUIntAt(buffer, 2); // Offset 2.0
        if (indicatorMetalParts != newIndicatorMetalParts)
        {
            indicatorMetalParts = newIndicatorMetalParts;
            OnMetalCountChanged?.Invoke(indicatorMetalParts);
        }

        int newIndicatorNonMetalParts = S7.GetUIntAt(buffer, 4); // Offset 4.0
        if (indicatorNonMetalParts != newIndicatorNonMetalParts)
        {
            indicatorNonMetalParts = newIndicatorNonMetalParts;
            OnNonMetalCountChanged?.Invoke(indicatorNonMetalParts);
        }
    }
    else
    {
        Debug.LogError("DB1 read error: " + client.ErrorText(readResult));
        Invoke("ReconnectToPLC", 5);
    }
}
else
{
    Debug.LogWarning("PLC not connected, attempting to reconnect...");
    ConnectToPLC();
}
}
```

Figura 3.28: Lectura de Datos del PLC.

En las figura 3.27 y 3.28 se define lo siguiente:

- **ReadPLCData()**: Lee datos del PLC. Si el cliente está conectado, intenta leer datos del DB1 del PLC. Si la lectura es exitosa, actualiza los estados de los componentes

y activa los eventos correspondientes si hay cambios. Si la lectura falla, registra un error y programa un intento de reconexión.

## Métodos para Establecer Estados de Componentes

```

public void SetConveyorState(bool state)
{
    if (client != null && client.Connected)
    {
        S7.SetBitAt(ref buffer, 0, 5, state); // Offset 0.5, bit 5
        int writeResult = client.DBWrite(1, 0, buffer.Length, buffer);
        if (writeResult != 0)
        {
            Debug.LogError("DB1 write error: " + client.ErrorText(writeResult));
        }
    }
}

2 referencias
public void SetStartButtonState(bool state)
{
    if (client != null && client.Connected)
    {
        S7.SetBitAt(ref buffer, 0, 0, state); // Offset 0.0, bit 0
        int writeResult = client.DBWrite(1, 0, buffer.Length, buffer);
        if (writeResult != 0)
        {
            Debug.LogError("DB1 write error: " + client.ErrorText(writeResult));
        }

        if (state)
        {
            SetStopButtonState(false); // Asegurar que el botón de parada esté en falso
        }

        OnStartButtonStateChanged?.Invoke(state);
    }
}

```

Figura 3.29: Métodos para Establecer Estados de Componentes.

```

public void SetStopButtonState(bool state)
{
    if (client != null && client.Connected)
    {
        S7.SetBitAt(ref buffer, 0, 1, state); // Offset 0.1, bit 1
        int writeResult = client.DBWrite(1, 0, buffer.Length, buffer);
        if (writeResult != 0)
        {
            Debug.LogError("DB1 write error: " + client.ErrorText(writeResult));
        }

        if (state)
        {
            SetStartButtonState(false); // Asegurar que el botón de inicio esté en falso
        }

        OnStopButtonStateChanged?.Invoke(state);
    }
}

3 referencias
public void SetSensorState(bool state, int bitIndex)
{
    if (client != null && client.Connected)
    {
        S7.SetBitAt(ref buffer, 0, bitIndex, state); // Usar bitIndex para el offset correcto
        int writeResult = client.DBWrite(1, 0, buffer.Length, buffer);
        if (writeResult != 0)
        {
            Debug.LogError("DB1 write error: " + client.ErrorText(writeResult));
        }
    }
}

```

Figura 3.30: Métodos para Establecer Estados de Componentes.

```
public void SetActuatorState(bool state)
{
    if (client != null && client.Connected)
    {
        S7.SetBitAt(ref buffer, 1, 0, state); // Offset 1.0, bit 0
        int writeResult = client.DBWrite(1, 0, buffer.Length, buffer);
        if (writeResult != 0)
        {
            Debug.LogError("DB1 write error: " + client.ErrorText(writeResult));
        }

        actuator = state;
        OnActuatorStateChanged?.Invoke(actuator);
    }
}
```

Figura 3.31: Métodos para Establecer Estados de Componentes.

En la figura 3.29, 3.30 y 3.31 se definen los métodos SetConveyorState, SetStartButtonState, SetStopButtonState, SetSensorState, y SetActuatorState permiten establecer el estado de diferentes componentes del PLC al escribir datos en la base de datos del PLC, utilizando S7.SetBitAt para establecer bits específicos en el buffer. Luego, llaman a client.DBWrite para escribir el buffer actualizado en el PLC. Si la escritura es exitosa, actualizan los estados locales y activan los eventos correspondientes.

### 3.5. Pruebas de funcionamiento

La figura 3.32 muestra una vista panorámica de l Unity3D, evidenciada por los mensajes en consola "Connection ok" y "DB1 Read ok". La conexión es estable y los datos se presentan de manera clara y precisa, indicando que no hay errores en la comunicación.

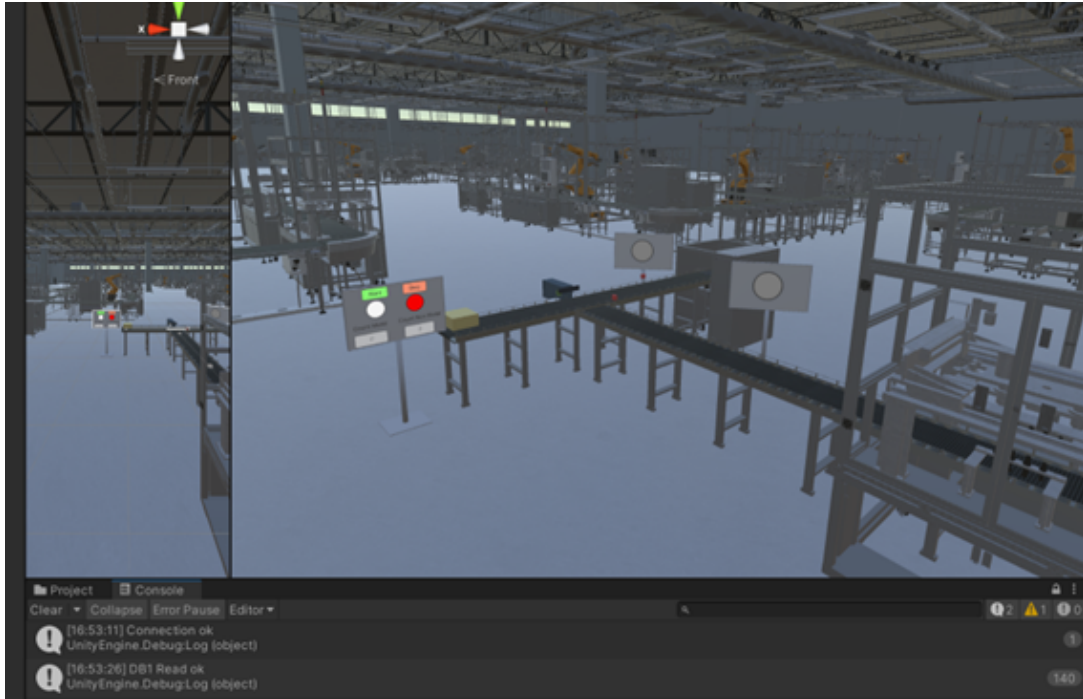


Figura 3.32: Prueba de funcionamiento 1.

La figura 3.33 muestra un sistema de conteo automatizado en un entorno industrial. En el panel de control, los botones “Start” y “Stop” permiten iniciar y detener el proceso. Los contadores digitales indican la cantidad de objetos metálicos (1) y no metálicos (4) detectados. La cinta transportadora con un sensor clasifica los objetos, actualizando los conteos en tiempo real. Este sistema, simulado en Unity3D, demuestra una operación eficiente y precisa.

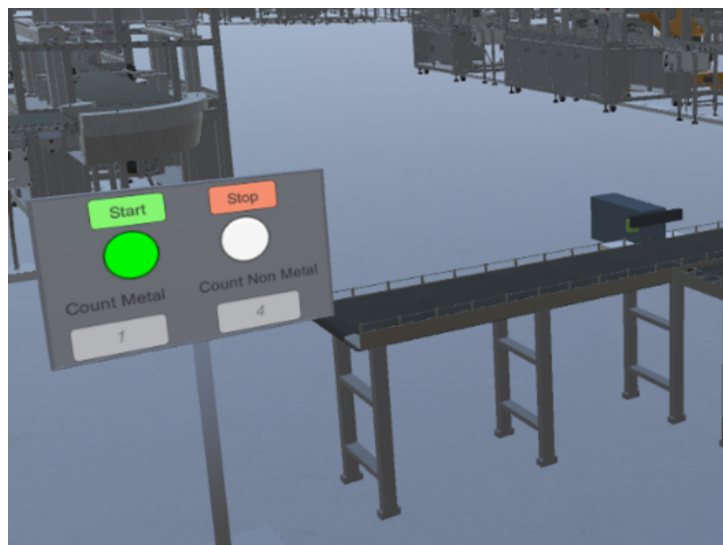
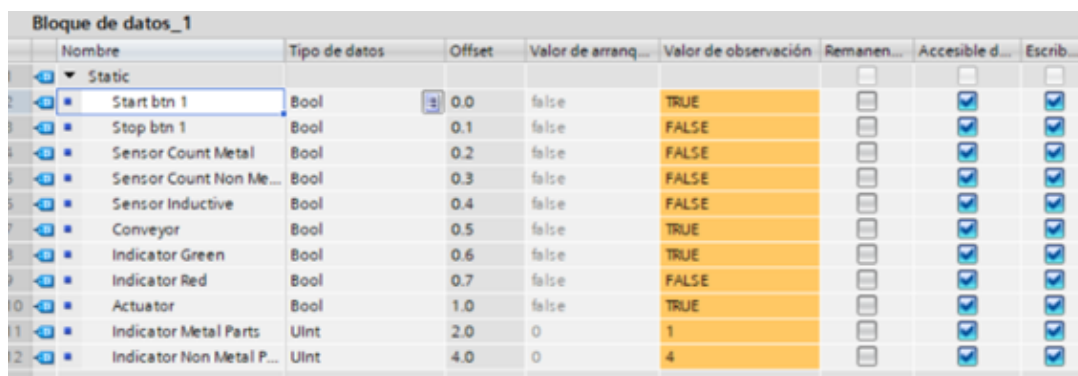


Figura 3.33: Prueba de funcionamiento 2.



La figura 3.34 se puede observar las diferentes variables del bloque de datos que fueron agregados en TIA PORTAL y visualizar el estado de cada una de estas, a su vez se puede comprobar que el valor de cada uno de los contadores coincide con el mostrado en la figura de la prueba de funcionamiento 2.



Nombre	Tipo de datos	Offset	Valor de arranq...	Valor de observación	Remanen...	Accesible d...	Escrib...
Static							
Start btn 1	Bool	0.0	false	TRUE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Stop btn 1	Bool	0.1	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sensor Count Metal	Bool	0.2	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sensor Count Non Me...	Bool	0.3	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sensor Inductive	Bool	0.4	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Conveyor	Bool	0.5	false	TRUE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indicator Green	Bool	0.6	false	TRUE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indicator Red	Bool	0.7	false	FALSE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Actuator	Bool	1.0	false	TRUE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indicator Metal Parts	UInt	2.0	0	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indicator Non Metal P...	UInt	4.0	0	4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 3.34: Prueba de funcionamiento 3.

La figura 3.35 muestra una consola de depuración de Unity3D con mensajes de log que indican la detección de objetos. Los mensajes son:

- Non Metal Detected: Un objeto no metálico fue detectado.
- Metal Detected: Un objeto metálico fue detectado.

Estos registros indican que el sistema está monitoreando y detectando correctamente diferentes tipos de materiales, registrando las detecciones con sus respectivas marcas de tiempo.



Figura 3.35: Prueba de funcionamiento 4.

En la figura 3.36 adjunta se puede observar la dirección IP utilizada, la cual es 192.168.0.1. Esta dirección se encuentra dentro del código PLCConnection de Unity3D, de esta manera se comprueba su conexión con el PLC real.



Figura 3.36: Prueba de funcionamiento 5.

La figura 3.37 muestra el entorno de trabajo con un controlador programable SIMATIC S7-1500 conectado a una computadora portátil mediante un cable ethernet. La pantalla de la computadora muestra software de programación y monitoreo TIA Portal y Unity 3D. Al lado derecho del PLC se pueden observar las variables de salida del mismo.

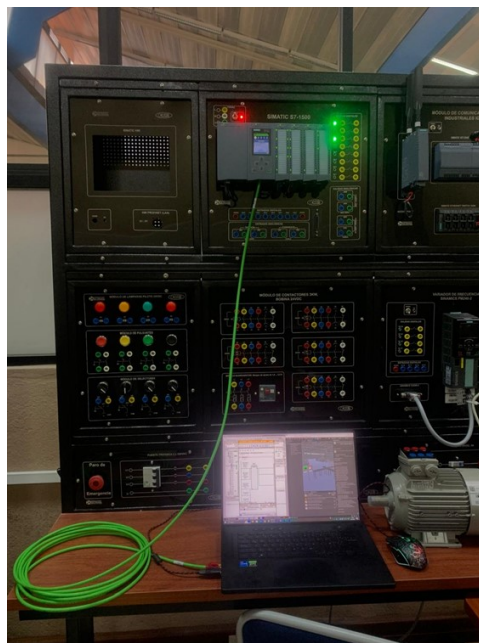


Figura 3.37: Prueba de funcionamiento 6.

# Capítulo 4

## Conclusiones y Futuros Trabajos

### 4.1. Conclusiones

- La implementación del entorno virtual en Unity 3D y su conexión con el PLC SIMATIC S7-1500 a través de Sharp 7 demuestra una integración efectiva de todos los componentes que integran el sistema virtual automatizado. A través de Sharp 7 se ha podido leer y escribir variables del PLC interactuando con C#. Esto abre un abanico de posibilidades para poder integrar el mundo de la automatización industrial con Unity 3D para la creación y automatización de todo tipo de procesos virtuales.
- El diseño del entorno virtual en Unity 3D ha permitido que un PLC real pueda interactuar con un proceso virtual que cumple una determinada función. Esta interacción ofrece al mundo académico y al mundo industrial una herramienta poderosa para el entrenamiento en el aprendizaje de la programación de PLCs y para probar cambios en la automatización de sistemas industriales, evitando la interacción con maquinaria real, lo cual, podría exponer a los operarios o comprometer su funcionalidad.
- La automatización del proceso utilizando el TIA Portal se ha completado con éxito, con valores que coinciden entre el bloque de datos en TIA Portal y los contadores en Unity 3D. La correcta detección y registro de objetos metálicos y no metálicos en el entorno virtual confirma la precisión del sistema.

- La utilización de librerías y elementos previamente creados facilita el desarrollo de aplicaciones. No obstante, se debe considerar que el desarrollo de aplicaciones de gran escala podría requerir una cantidad de tiempo considerable dado que la creación de entornos virtuales en Unity no es una tarea sencilla, pues, como se ha visto a lo largo de este documento, se requiere tener en cuenta varios aspectos.

## 4.2. Futuros Trabajos

- El presente trabajo deja las bases necesarias para poder crear un interfaz de comunicación entre Unity y un PLC real de la familia S7-1500. De aquí en adelante se podría experimentar con la automatización de procesos más complejos que involucren mayor cantidad de maquinaria y se generen otro tipo de variables, como por ejemplo variables reales para el tratamiento de señales analógicas.
- Para complementar la arquitectura se podría incorporar un panel HMI para que el usuario pueda interactuar de mejor manera con el proceso industrial. Esto también podría abordarse bajo el nuevo contexto de la industria 4.0, incorporando dashboards o tableros virtuales de control sobre páginas web, para realizar un monitoreo remoto del proceso.
- Se podría evaluar o comparar otros métodos de comunicación como por ejemplo OPC clásico u OPC UA. Esto en aras de tener diversas alternativas de conexión entre el software y los controladores industriales.

# Glosario

**Actuator** Actuador – Dispositivo que mueve o controla un mecanismo o sistema.

**Asset Store** Tienda de Recursos – Plataforma de Unity donde los desarrolladores pueden comprar y vender activos digitales como modelos 3D, texturas, sonidos y scripts.

**Assets** Activos – Recursos necesarios para el desarrollo de un proyecto en Unity 3D, incluyendo modelos 3D, texturas, scripts y más.

**Buffer** Búfer – Espacio de memoria utilizado para almacenar datos temporalmente mientras se transfieren entre dos lugares.

**C#** Lenguaje de programación utilizado en Unity 3D para escribir scripts que definen el comportamiento de los objetos y la lógica del juego.

**Digital Input** Entrada Digital – Señal digital que entra en el PLC desde un sensor u otro dispositivo.

**Digital Output** Salida Digital – Señal digital que sale del PLC para activar un actuador u otro dispositivo.

**FBX** Formato de archivo utilizado para almacenar modelos 3D que pueden ser importados a Unity.

**Gemelo Digital** Digital Twin – Representación virtual de un sistema físico que se utiliza para el monitoreo y la optimización de procesos en tiempo real.

**PLC** Controlador Lógico Programable – Dispositivo utilizado para el control automático de procesos industriales, máquinas y equipos.

**Prefabs** Objetos preconfigurados en Unity que pueden ser reutilizados en diferentes partes del proyecto para mantener la consistencia y eficiencia en el desarrollo.

**Raycast** Técnica utilizada en Unity para detectar objetos en una dirección y distancia específicas, comúnmente utilizada para la detección de colisiones y la interacción con el entorno.

**S7Client** Cliente utilizado en el código de Unity para establecer una conexión y comunicarse con el PLC.

**Script** Archivo de código que define la lógica y el comportamiento de los objetos en un proyecto de Unity 3D.

**Sensor** Dispositivo que detecta cambios en el entorno físico y envía esa información al PLC para su procesamiento.

**Sharp7** Librería utilizada para la comunicación con PLCs de Siemens, permitiendo la transferencia de datos entre Unity y el PLC.

**SIMATIC S7-1500** Familia de controladores programables de Siemens diseñados para la automatización de tareas en entornos industriales, conocidos por su alta velocidad de procesamiento y capacidades avanzadas.

**TIA Portal** Software de Siemens utilizado para la programación y configuración de PLCs, incluyendo el SIMATIC S7-1500.

**Unity 3D** Potente motor de desarrollo multiplataforma utilizado para crear juegos, simulaciones y aplicaciones en 2D, 3D, Realidad Virtual (VR) y Realidad Aumentada (AR).

# Referencias

- [1] M. F y O. L, «Entorno de realidad virtual 3D que simule un proceso batch mediante “Hardware In the Loop”, orientado al entrenamiento de PLC en los laboratorios de Automatización y control de la Universidad Técnica de Cotopaxi,» págs. 1-77, dic. de 2021. dirección: <http://repositorio.espe.edu.ec/handle/21000/30961>.
- [2] F. Pereira y C. Felgueiras, «Learning Automation from Remote Labs in Higher Education,» Association for Computing Machinery, oct. de 2020, págs. 558-562, ISBN: 9781450388504. DOI: 10.1145/3434780.3436689.
- [3] S. C. Rowe, C. I. Samson y D. E. Clough, «A Framework to Guide the Instruction of Industrial Programmable Logic Controllers in Undergraduate Engineering Education,» *Education for Chemical Engineers*, vol. 31, págs. 76-84, abr. de 2020, ISSN: 1749-7728. DOI: 10.1016/j.ece.2020.03.001.
- [4] L. Zujovic, V. Kecojevic y D. Bogunovic, «Interactive mobile equipment safety task-training in surface mining,» *International Journal of Mining Science and Technology*, vol. 31, págs. 743-751, 4 jul. de 2021, ISSN: 20952686. DOI: 10.1016/j.ijmst.2021.05.011.
- [5] E. Riedel, «MQTT protocol for SME foundries: Potential as an entry point into industry 4.0, process transparency and sustainability,» vol. 105, Elsevier B.V., 2022, págs. 601-606. DOI: 10.1016/j.procir.2022.02.100.
- [6] S. J. Pazmiño Jackson, «DESARROLLO DE UN ENTORNO VIRTUAL 3D PARA EL CONTROL DE NIVEL DE AGUA,» *Education for Chemical Engineers*, vol. 31, págs. 76-129, abr. de 2021.
- [7] M. Yu, Z. Zhan, Z. Lin, Y. Feng, Z. Chen y Q. Li, «The Design and Application of C-STEAM Instructional Resources Based on Unity 3D Blending Physical and Virtual Manipulatives,» en *2022 14th International Conference on Education*

- Technology and Computers (ICETC)*, ACM, Barcelona, Spain, 2022, págs. 150-157, ISBN: 978-1-4503-9776-6. DOI: 10.1145/3572549.3572574.
- [8] D. Luo, Y.-w. Luo, X.-l. Deng y G.-x. Wang, «Design and Implementation of Virtual Examination System Based on Unity 3D,» en *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, ACM, Dublin, Ireland, 2019, págs. 1-8, ISBN: 978-1-4503-7202-2. DOI: 10.1145/3358331.3358404.
- [9] S. Jangra, S. Angra, G. Singh, B. Sharma y A. Mantri, «Interactivity Development Using Unity 3D Software and C# Programming,» en *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Delhi, India, 2023, págs. 1-8, ISBN: 979-8-3503-3509-5. DOI: 10.1109/ICCCNT56998.2023.10308030.
- [10] M. Chaudhary, G. Singh, L. Gaur, N. Mathur y S. Kapoor, «Leveraging Unity 3D and Vuforia Engine for Augmented Reality Application Development,» en *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, IEEE, 2023, págs. 1139-1144. DOI: 10.1109/ICTACS59847.2023.10390072. dirección: <https://ieeexplore.ieee.org/document/10390072>.
- [11] S. Shi, X. Zhang, C. Ling y Q. Meng, «A Visual Simulation System Architecture and Implementation Method of Target Damage Assessment Based on Unity 3D Interactive Technology,» en *2023 4th International Conference on Intelligent Design (ICID)*, IEEE, 2023, págs. 37-40. DOI: 10.1109/ICID60307.2023.10396688. dirección: <https://ieeexplore.ieee.org/document/10396688>.
- [12] L. Ivanov, «Game Development with Unity in the Computer Science Curriculum,» en *2018 CCSC Northwestern Conference*, Consortium for Computing Sciences in Colleges, New Rochelle, NY, USA, 2018, ISBN: 715 North Avenue, New Rochelle, NY 10801.
- [13] J. Krishnareddy, «C#: el lenguaje de programación del año 2023,» *C# Corner*, ene. de 2024. dirección: <https://www.c-sharpcorner.com/article/c-sharp-language-of-the-year-2023/>.
- [14] K. Yang, J. Jie y S. Haihui, «Study on the Virtual Natural Landscape Walkthrough by Using Unity 3D,» en *2011 IEEE International Symposium on Virtual Reality Innovation*, IEEE, Singapore, 2021, págs. 235-238, ISBN: 978-1-4577-0054-5. DOI: 10.1109/ISVRI.2011.5759640.



- [15] J. Mou, «Construction of a virtual simulation teaching system for aeroengine maintenance based on Unity 3D,» en *2023 International Conference on Information Education and Artificial Intelligence (ICIEAI 2023)*, Xiamen, China: ACM, 2023, págs. 1-7. DOI: 10.1145/3660043.3660144.
- [16] Unity Technologies, *Pie de página de la tienda de activos de Unity*, Recuperado de <https://assetstore.unity.com>, 2024. dirección: <https://assetstore.unity.com>.
- [17] *Tienda de activos de Unity*, <https://assetstore.unity.com>, Búsqueda de activos: 3D, 2D, Complementos, Audio, AI, Descentralización, Lo esencial, Plantillas, Herramientas, Efectos visuales, Venta, Vender activos. Más de 11.000 activos de cinco estrellas. Calificado por más de 85.000 clientes. Con el apoyo de más de 100.000 miembros del foro. Todos los activos moderados por Unity. Descripción del activo: Demostración fotorrealista de una fábrica de producción virtual con elementos como máquinas, cintas transportadoras, mercancías, brazos robóticos y trabajadores. Compatible con HDRP. Tamaño del archivo: 882,0 MB. Última versión: 1.0. Última fecha de lanzamiento: 25 de marzo de 2024. Versión original de Unity: 2022.3.16 o superior., 2024.
- [18] Z. Wen y W. Huang, «Design and Implementation of a Control System for Atmospheric Pressure Microwave Plasma Torch Based on PLC,» en *2023 the 6th International Conference on Robot Systems and Applications (ICRSA)*, Wuhan, China: ACM, 2023, pág. 5. DOI: 10.1145/3655532.3655550.
- [19] J. L. A. Serrano y C. J. G. Toledo, «Diseño e Implementación de un Módulo de Entrenamiento con PLC S7-1500 para Aplicaciones de Control de AVG,» Trabajo de Titulación, Carrera de Ingeniería Electrónica, Universidad Politécnica Salesiana, Guayaquil, Ecuador, 2021.
- [20] IEEE, «Título del Artículo,» en *Proceedings of the Conference*, 2019, págs. 547-551, ISBN: 978-1-7281-3936-4. DOI: 10.1109/XXX.2019.XXXXXX.