universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo de Fin de Grado

SUPERVISIÓN DE UN ARMARIO DE CONTROL CON REALIDAD AUMENTADA

SUPERVISION OF A CONTROL CABINET WITH AUGMENTED REALITY

Autor: Alicia García Turiel
Tutor: Juan José Fuertes Martínez
Cotutor: Raúl González Herbón

(Julio, 2023)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA
Trabajo de Fin de Grado

ALUMNO: ALICIA GARCÍA TURIEL

TUTOR: JUAN JOSÉ FUERTES MARTÍNEZ, RAÚL GONZÁLEZ HERBÓN

TÍTULO: SUPERVISIÓN DE UN ARMARIO DE CONTROL CON REALIDAD AUMENTADA

TITLE: SUPERVISIÓN OF A CONTROL CABINET WITH AUGMENTED REALITY

CONVOCATORIA: JULIO,2023

RESUMEN:
En el proyecto "Supervisión de un Armario de Control con Realidad Aumentada" se ha desarrollado una aplicación software que permite, a través del reconocimiento de códigos QR, la supervisión del variador, el PLC y el medidor de energía de uno de los armarios de control del aula IoT Schneider Electric de la Escuela de Ingenierías de León. La aplicación usa el protocolo de comunicación Modbus TCP para lectura de las variables y conexión con los elementos. Para el desarrollo de la interfaz de la aplicación se emplea el software Unity 3D junto a la extensión de realidad aumentada de Vuforia. Los códigos que se compilen y ejecuten en Unity 3D se programan utilizando el lenguaje C#.

ABSTRACT:
The project "Supervision of a Control Cabinet with Augmented Reality" develops software app which scans QR codes, and it allows users to supervise the motor frequency driver, the PLC, and the energy measurer of one of the control cabinets located in the Schneider Electric IoT Laboratory. The app uses, as communication protocol, Modbus TCP. Unity 3D and Vuforia extension are used for the development of the application interface. The codes are written in C# language.

Palabras clave: realidad, aumentada, supervisión, Modbus, Unity, Vuforia

Firma del alumno:	**VºBº Tutor/es:F**

Resumen

El trabajo consiste en la realización de una aplicación software de Realidad Aumentada, con la plataforma Unity 3D y la extensión Vuforia, que permita la supervisión de varios elementos que componen un armario de control de Schneider Electric situado en el aula IoT de la Escuela de Ingeniería de León: el variador de un motor, el PLC y un medidor de energía. Mediante el escaneo de distintos códigos QR colocados en cada uno de los componentes que se desean supervisar, se facilitará la visualización de los distintos parámetros y variables involucradas en el proceso de control a través de la pantalla de un dispositivo móvil, en este caso utilizaremos una Tablet. Los elementos utilizarán el protocolo Modbus TCP para su comunicación, por lo que resultará imprescindible utilizar este protocolo para realizar la lectura de las variables desde la aplicación realizada con Unity. Los scripts, que se compilen y ejecuten en la interfaz de Unity 3D, se programarán en lenguaje C#; además se utilizará el IDE Visual Studio Basic para su edición y compilación. Para conocer los espacios de la memoria de cada componente en el que se almacenan las variables que se pretenden leer se utilizarán los mapas Modbus facilitados por el fabricante (Schneider Electric) en los casos del variador y el medidor de energía; para el PLC será necesario utilizar el software de programación Unity pro XL y cargar el programa desde el autómata al equipo.

Los resultados obtenidos en el trabajo ponen en manifiesto las ventajas que ofrece la implementación de sistemas de Realidad Aumentada en el ámbito de la supervisión industrial, agilizando y facilitando todas las tareas de monitorización y gestión de los procesos automatizados. Además, la aplicación desarrollada en este proyecto permite realizar la tarea de supervisión a nivel de campo, sin necesidad de desplazarse a un equipo donde este el HMI disponible. Lo cual permite aumentar la eficacia y reducir el tiempo de reacción ante un problema, en la automatización de un proceso industrial.

Abstract

The project consists of an Augmented Reality software application which is made by using the Unity 3D platform in combination with the Vuforia extension. The app allows the supervision of some elements which are in one of the Schneider Electric control cabinets installed in the IoT laboratory in the Leon Engineering School: the motor frequency driver, the PLC, and a n energy measurer. It scans QR codes which are located over the components that we want to supervise, easing the visualization of the different parameters and variables which are involved in the process. The app shows the values in the screen of a mobile device, in this case it is a tablet. The elements use the industrial communication protocol Modbus TCP, so it is necessary to use this protocol rules for the value reading via Unity interface. The scripts compiled and executed are written in C# language. Also, the Visual Studio Basic IDE is used for editing and compiling the code. Modbus maps, eased by the fabricant (Schneider Electric), are employed to know the memory spaces where the variables are stored in the case of the frequency driver and the energy measurer. In the PLC it is necessary open the programming software Unity pro XL and load the project from the PLC to the computer.

The results obtained shows the advantages that Augmented Reality implementation offers in the Industrial Supervision area, it speeds up and eases the monitoring and management tasks. Besides, the app developed in this project allows the user to supervise in the field of the process. This leads to an efficiency increase and problem response time reduction in industrial automatization.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

Se considera un sistema de control industrial a todo tipo de software, que, en combinación con el hardware necesario, permiten la monitorización, supervisión, gestión y soporte de las infraestructuras y procesos industriales; optimizando al máximo la productividad, rendimiento y calidades finales. En 1959 se logra, por primera vez, controlar una máquina a través de un ordenador. A principio de los años 60, Westinghouse y North Electric Company desarrollan el primer sistema de control supervisado aplicado a las líneas de telefonía, llamado Visicode; siendo el punto de inicio de los sistemas SCADA (Supervisory Control and Data Acquisition) actuales [1].

Desde sus inicios en 1968, de la mano de Evan Sutherland, la Realidad Aumentada (RA) ha supuesto un reto complicado, debido a la ausencia de estudios e investigación y a la pobreza de los recursos informáticos destinados [2, 3]. En la actualidad, gracias al desarrollo de software como Unity3D o Autodesk Maya [4], su popularidad ha aumentado de forma exponencial, especialmente en el desarrollo de videojuegos y para el desarrollo de aplicaciones pedagógicas en la enseñanza [5, 6].

1.1 OBJETIVO DEL TRABAJO

El objetivo final del proyecto es el desarrollo de una aplicación software que permite, a través de una cámara de realidad aumentada instalada en una Tablet, el reconocimiento y supervisión de distintos componentes que forman parte de un armario de supervisión. Para la lectura de los valores de las variables que intervienen en el proceso se utiliza el protocolo de comunicación industrial Modbus TCP.

1.2 ESTRUCTURA DEL DOCUMENTO

El documento se estructura de la siguiente manera: primero, comienza con un estado del arte que versa sobre la supervisión industrial, industria 4.0, realidad aumentada y protocolos de comunicación. Después, se realiza una explicación teórica de las herramientas, información y técnicas aplicadas para el desarrollo del proyecto, comenzando por explicar el protocolo Modbus TCP. A continuación, se describen los recursos hardware y el software utilizado. Por último, se realiza una breve guía con todos los pasos seguidos para poder realizar la aplicación y comunicación con el variador, así como la explicación de todas las líneas de código necesarias.

2. Estado del arte

2.1 CONTROL Y SUPERVISIÓN INDUSTRIAL

En un proceso automatizado en la industria resulta un aspecto fundamental realizar un seguimiento y control que garanticen un correcto funcionamiento y seguridad. Para ello es necesario combinar hardware y software, con la correspondiente conectividad de red, permitiendo alcanzar los valores de las variables deseados, según las circunstancias; y la corrección de cualquier problema o desviación que pueda ocurrir durante el proceso.

Los PLC (Controladores Lógicos Programables) llevan utilizándose en la industria por más de 40 años. Surgieron en la década de los 60 para sustituir los sistemas basados en relés. Estos primeros PLCs apenas disponían de memoria y requerían de módulos y terminales externos para poder ser programados. A partir de 1980, se crearon los primeros programas de software que permitían la programación de los PLC en un ordenador [7], desde entonces la mejora de sus características ha sido continua y su usabilidad en la industria ha incrementado hasta ser uno de los sistemas más comunes para llevar a cabo la automatización y control de un proceso. Los PLC operan de forma digital y cíclica, proporcionando tiempos de repuesta rápidos.

Los sistemas de control distribuido (DCS) se utilizan en procesos industriales grandes y complejos, como pueden ser la industria petroquímica o la metalurgia. Se desarrollaron a mediados de los años 70 y han evolucionado desde 1980 para dar lugar unidades de gran concentración de información [8]. A través de uno o más controladores permite comunicar y monitorear el sistema, que se distribuye en cada uno de los componentes (o subsistemas). Proporcionan una mayor escalabilidad.

En los últimos años se han desarrollado los sistemas SCADA (Control Supervisado y Adquisición de Datos) que permiten supervisar y controlar las distintas variables implicadas en un proceso o planta. El sistema de supervisión de las líneas telefónicas Visicode, creado a principios de los años 60 se considera el predecesor de los futuros sistemas SCADA [1]. El sistema facilita la comunicación con todos los elementos de campo (como lo son sensores, actuadores o autómatas) a través de la pantalla de un ordenador que puede ser configurada y modificada por el usuario fácilmente. Permite controlar el proceso en tiempo real desde la propia pantalla del ordenador, y, además, provee con la información que se genera durante el proceso productivo [9].

2.2 INDUSTRIA 4.0 Y REALIDAD AUMENTADA

La industria 4.0 es considerada como la "cuarta revolución industrial" debido a sus beneficios y potencial en la autonomía, innovación e integración de los procesos. Contempla la introducción de las tecnologías industriales en el ambiente de la industria de la fabricación. Esto quiere decir la introducción de tecnologías como el Internet de las Cosas (IoT), Big Data, Computación en la Nube, redes de sensores inalámbricos o dispositivos móviles entre otros [10]. La Industria 4.0 da lugar a un nuevo modelo de organización y de control del ciclo de vida del producto, respaldado por las tecnologías de la información. Los productos inteligentes se basan en disponer de electrónica, software y conectividad que, en conjunto, les dota de nuevas capacidades, funciones y características [11].

Por otra parte, la aplicación de RA en el ámbito industrial comienza a ser una realidad, sobre todo en el campo del montaje y ensamblaje, donde ha quedado comprobado que, además de reducir la carga de trabajo, disminuye el número de errores cometidos, dando como resultado una disminución notable en el tiempo empleado [12]. Podemos encontrar algunos ejemplos de aplicación de RA en nano fabricación [13], en las industrias de automoción y aeroespacial [8], o en la industria naval [14]. También ha quedado demostrado que la RA resulta una herramienta eficaz de cara a detectar incompatibilidades entre los modelos de CAD y las piezas fabricadas [15].

A raíz de la introducción de la RA en el ambiente industrial y en combinación con el Internet de las Cosas, ha surgido el IoT-RA para facilitar las tareas de control remoto y mantenimiento en ambientes donde es difícil visualizar el proceso, o donde las condiciones resultan desfavorables [16, 17]. Esto da lugar a que los trabajadores puedan acceder a los datos del IoT, en el propio campo donde ocurre el proceso, dando un sentido y contexto real a dicha información.

Uno de los principales problemas que presentan los procesos de automatización en la I.4.0 es que los operarios deben supervisar visualmente que, el proceso realizado por las máquinas y el proyectado en el SCADA, coinciden. Esta división del foco de atención resulta poco eficiente, con el inconveniente añadido de que la información que reciben los trabajadores carece de un contexto real, al quedar aislada del entorno donde se produce el trabajo real. Hay estudios que respaldan que la aplicación de RA en combinación al protocolo actual IEC 61850, pone solución a dicho problema, integrando los datos del proceso y el lugar de trabajo

en un mismo espacio [18]. Otra solución consiste en aplicar tecnología de RA en los propios equipos de retransmisión donde la información pasa a formar parte del espacio físico [19].

La inclusión de RA en la industria I.4.0 y en combinación con el IoT, da como resultado un trabajo más fluido, donde los sensores pueden detectar y anticipar los problemas transmitir estos datos a los operarios, creando un ambiente virtual, en el que la información y el lugar de trabajo se combinan en un único espacio real [20].

2.3 COMUNICACIONES INDUSTRIALES

Las redes de comunicación se emplean en el ámbito industrial con el fin de conectar e intercambiar datos entre los elementos de campo (sensores y actuadores), PLCs, sistemas SCADA, equipos y ordenadores, y otros dispositivos que formen parte del proceso automatizado. La forma en que la información se envía y se recibe, el medio y a la velocidad que se transmite y el tipo de red se definen en los protocolos de comunicación.

El modelo OSI es un modelo estandarizado por ISO y tiene como finalidad la comunicación de los protocolos por 7 capas, ordenadas de manera jerárquica. El modelo TCP/IP también se basa en la comunicación por capas, con la diferencia de que el modelo TCP/IP solo presenta 4 capas. Estos dos modelos son la base esencial de los protocolos de comunicación entre dispositivos [21].

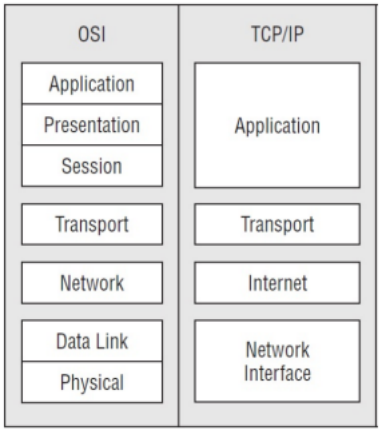


Figura 2. 1 Comparación entre los modelos OSI y TCP [21]

Modbus, desarrollado por Schneider; o Profibus promulgado por Siemens; son dos ejemplos de protocolos en los que el medio de transmisión se basa en tecnología de par trenzado (estándares RS-232 y RS-485). Con el paso de los años, han evolucionado y se han adaptado al paradigma de la industria 4.0 en el que convivimos actualmente, dando lugar a Modbus

TCP y Profinet, en dónde el medio de transmisión es un cable Ethernet estándar, dando lugar a protocolos con una tecnología más moderna [22].

Actualmente, con el IoT (Internet of Things) la movilidad de las redes resulta un aspecto crucial, es por eso por lo que cada vez se tiende a escoger más redes inalámbricas que cableadas, otro ejemplo es el desarrollo del estándar de comunicaciones WirelessHART. Este estándar de comunicación opera en la banda de radio de 2,4GHz, que es la misma que utilizan las redes wifi (IEEE 802.11); y basándose en el protocolo IEEE 802.15.4. Esta tecnología pretende dar lugar a una red de dispositivos, dispuestos en malla, que son capaces de autoorganizarse [23].

Desarrollados los sistemas SCADA, y con el avance hacia redes de comunicación inalámbricas y por tanto móviles, es el momento de que a través de la aplicación de Realidad Aumentada se logre un progreso hacia una industria que permita una mayor flexibilidad, y sea capaz de adaptarse a las circunstancias de cada momento del proceso; sin que el espacio y entorno físicos supongan un impedimento.

3. Modbus TCP

El protocolo de comunicación empleado para la comunicación con el variador será Modbus TCP/IP. Consiste en un protocolo sencillo creado en 1979 por Schneider Electric (Modicon) y que actualmente supone uno de los lenguajes de comunicación con PLCs más extendidos y ampliamente utilizados en la industria.

Modbus TCP se basa en el mismo protocolo que Modbus serie (RTU) al que se aplica una interfaz TCP que se ejecuta en Ethernet [24] Las ventajas que presenta el modo de implementación TCP/IP frente a la implementación de Modbus serie es que varios servidores comunes pueden atender al mismo tiempo a un grupo de clientes conectados a la misma red IP, es decir, no hay un único maestro encargado de controlar un grupo definido de esclavos [25].

Responde a un esquema clásico de comunicación TCP, compuesto por cinco capas (al igual que en Internet), y dónde el maestro inicia la comunicación solicitando una acción al esclavo a través de un código de función. El esclavo responderá con los datos solicitados y con el código de la acción realizada.

Los mensajes están compuestos por dos tipos de tramas:

- Unidad de Datos de Protocolo (PDU)
- Unidad de Datos de Aplicación (ADU)

Figura 3. 1 Tramas Modbus

Los tipos de datos tratados corresponden a cuatro tipos:

1. Discrete inputs: entradas digitales; ocupan 1bit y sólo se puede realizar su lectura.
2. Coils: salidas digitales; ocupan 1 bit y se puede realizar tanto su lectura, como su escritura.
3. Input Registers: entradas analógicas; 16 bits, sólo lectura.
4. Output Registers: salidas analógicas; 16 bits, lectura y escritura.

La comunicación con el protocolo Modbus TCP/IP necesita tres parámetros esenciales para establecer una conexión: Dirección IP; número de puerto que, en Modbus TCP, siempre se utilizará el 502, dirección ID; y, por último, añadir el código de función.

La clase ID hace referencia al objeto en donde se encuentra el atributo; por ejemplo: el atributo tipo de dispositivo está dentro del objeto identidad [26]. Aunque en Modbus TCP en un principio no resulte fundamental el ID, sí que puede resultar necesario en caso de utilizar pasarelas Modbus, ya que, con una misma IP permiten conectar varios dispositivos, y, sin una dirección ID resultará imposible diferenciarlos.

El código función identifica el tipo de operación que deseamos realizar.

Códigos de función				
Acceso a datos	1 bit	Entradas digitales	Read Discrete Inputs	02
		Salidas digitales (Coils)	Read Coils	01
			Write Single Coil	05
			Write Multiple Coils	0F
	16 bits	Registros de entrada	Read Input Register	04
		Registros de Salida	Read Holding Registers	03
			Write Single Register	06
			Write Multiple Registers	10
			Read/Write Multiple Registers	17
			Write Mask Register	16
			Read FIFO Queue	18
	Ficheros		Read File Record	14
			Write File Record	15
Diagnóstico			Read Exception Status	07
			Diagnostic	08
			Get Com Event Counter	0B
			Get Com Event Log	0C
			Report Slave ID	11
			Read Device Identification	2B
Otros			Encapsulated Interface Transport	2B

Figura 3. 2 Códigos de función Modbus

3.1 MAPAS MODBUS

La memoria de los dispositivos viene dividida en registros, que apuntan a una parte concreta de esa memoria dónde quedará almacenada un dato. Estos registros contienen información que puede leerse o escribirse según el tipo de dato. Normalmente, la memoria suele dividirse en los siguientes números de registro :

0-10000 Salidas digitales (DOs); como ya mencionamos ocuparán un bit de información

10001-20000 Entradas digitales (DIs); ocuparán igualmente 1 bit

20001-30000 No se usa

30001-40000 Entradas digitales (AIs); en este caso ocuparán o 16 o 32 bits de información

40001-50000 Salidas analógicas (AOs); su tamaño también será de 16 o 32 bits

Normalmente los fabricantes suelen encargarse de recoger que dato contiene cada registro en tablas y lo dejan a disposición de los usuarios para facilitar el acceso a los datos. Estas tablas se denominan mapas Modbus.

Name	Logic	CANopen	INTERBUS	DeviceNet	Link	Category	Access
Control word	16#2135 = 8501 (command)	16#6040h00	16#6040h00	16#8Bh01h66 = 139h01h102	BCEN_CM	Control parameters	R/W
Extended control word	16#2138 = 8504	16#2037h5	16#5FB6h1E	16#8Bh01h69 = 139h01h105	BCEN_CM	Control parameters	R/W
Reset counters command	16#0C30 = 3120	16#2001h5	16#5FB6h01	16#70h01h79 = 112h01h121	LCEN_RP	Control parameters	R/W
Speed setpoint	16#219A = 8602	16#6042h00	16#6042h00	16#24h01h08 = 42h18 (processing)	-	Setpoint parameters	R/W
Frequency setpoint	16#2136 = 8502	16#2037h3	16#5FB6h1C	16#8Bh01h67 = 139h01h103	-	Setpoint parameters	R/W
Torque setpoint	16#2139 = 8505	16#6071h00	16#5FB6h1F	16#8Bh01h6A = 139h01h106	-	Setpoint parameters	R/W
PID regulator setpoint	16#2137 = 8503	16#2037h4	16#5FB6h1D	16#8Bh01h68 = 139h01h104	-	Setpoint parameters	R/W
PID regulator feedback	16#14A1 = 5281	16#2016h52	16#5FB6h5B	16#7Bh01h52 = 123h01h82	-	Setpoint parameters	R/W
Multiplying coefficient	16#2E37 = 11831	16#2058h20	16#5FB6h3E	16#9Ch01h20 = 156h01h32	-	Setpoint parameters	R/W
Status word	16#215B = 8603 (command)	16#6041h00	16#6041h00	16#71h01h02 = 113h01h02	BCEN_ET	Status parameters	R
Drive state	16#0CA8 = 3240	16#2002h29	16#5FB9h1B	16#71h01h29 = 113h01h41	LCEN_HM	Status parameters	R
Extended status word	16#0C86 = 3206	16#2002h7	16#5FB9h08	16#71h01h07 = 113h01h07	BCEN_ET	Status parameters	R
Extended status word 1	16#0CB2 = 3250	16#2002h33	16#5FB9h1C	16#71h01h33 = 113h01h51	BCEN_LR	Status parameters	R
Extended status word 2	16#0CB3 = 3251	16#2002h34	16#5FB9h1D	16#71h01h34 = 113h01h52	BCEN_LR	Status parameters	R
Extended status word 3	16#0CB4 = 3252	16#2002h35	16#5FB9h1E	16#71h01h35 = 113h01h53	BCEN_LR	Status parameters	R
Extended status word 4	16#0CB5 = 3253	16#2002h36	16#5FB9h1F	16#71h01h36 = 113h01h54	BCEN_LR	Status parameters	R
Extended status word 5	16#0CB6 = 3254	16#2002h37	16#5FB9h20	16#71h01h37 = 113h01h55	BCEN_LR	Status parameters	R
Extended status word 6	16#0CB7 = 3255	16#2002h38	16#5FB9h21	16#71h01h38 = 113h01h56	BCEN_LR	Status parameters	R
Extended status word 7	16#0CB8 = 3256	16#2002h39	16#5FB9h22	16#71h01h39 = 113h01h57	BCEN_LR	Status parameters	R
Extended status word 8	16#0CB9 = 3257	16#2002h3A	16#5FB9h23	16#71h01h3A = 113h01h58	BCEN_LR	Status parameters	R
Active reference channel	16#20F9 = 8441	16#2036h2A	16#5FB9hCE	16#8Bh01h2A = 139h01h42	BCEN_CR	Status parameters	R
Active command channel	16#20FA = 8442	16#2036h2B	16#5FB9hCF	16#8Bh01h2B = 139h01h43	BCEN_CC	Status parameters	R
Config. active	16#1F54 = 8020	16#2032h15	16#5FB9hCD	16#89h01h15 = 137h01h21	LCEN_CN	Status parameters	R

Figura 3. 3 Mapa Modbus Variador ATV71

Quantity	Modbus Register	# of Modbus Registers	Modbus Data Type	Modbus Units	Modbus Permissions	Value Type	Enumerations	Web Page Tag	Display and Web Units
Active Energy Delivered (In	2700	2	FLOAT32	kWh	R/O	numeric	---	kWh del	KWH
Active Energy Received (Ou	2702	2	FLOAT32	kWh	R/O	numeric	---	kWh rec	KWH
Active Energy Delivered + R	2704	2	FLOAT32	kWh	R/O	numeric	---	kWh del+rec	KWH
Active Energy Delivered- Re	2706	2	FLOAT32	kWh	R/O	numeric	---	kWh del-rec	KWH
Reactive Energy Delivered	2708	2	FLOAT32	kVARh	R/O	numeric	---	kVARh del	KVARH
Reactive Energy Received	2710	2	FLOAT32	kVARh	R/O	numeric	---	kVARh rec	KVARH
Reactive Energy Delivered	2712	2	FLOAT32	kVARh	R/O	numeric	---	kVARh del+rec	KVARH
Reactive Energy Delivered	2714	2	FLOAT32	kVARh	R/O	numeric	---	kVARh del-rec	KVARH
Apparent Energy Delivered	2716	2	FLOAT32	kVAh	R/O	numeric	---	kVAh del	KVAH
Apparent Energy Received	2718	2	FLOAT32	kVAh	R/O	numeric	---	kVAh rec	KVAH
Apparent Energy Delivered	2720	2	FLOAT32	kVAh	R/O	numeric	---	kVAh del+rec	KVAH
Apparent Energy Delivered	2722	2	FLOAT32	kVAh	R/O	numeric	---	kVAh del-rec	KVAH
Active Energy in Quadrant I	2724	2	FLOAT32	kWh	R/O	numeric	---	kWh Q1	KWH
Active Energy in Quadrant I	2726	2	FLOAT32	kWh	R/O	numeric	---	kWh Q2	KWH
Active Energy in Quadrant I	2728	2	FLOAT32	kWh	R/O	numeric	---	kWh Q3	KWH
Active Energy in Quadrant I	2730	2	FLOAT32	kWh	R/O	numeric	---	kWh Q4	KWH
Reactive Energy in Quadrar	2732	2	FLOAT32	kVARh	R/O	numeric	---	kVARh Q1	KVARH
Reactive Energy in Quadrar	2734	2	FLOAT32	kVARh	R/O	numeric	---	kVARh Q2	KVARH
Reactive Energy in Quadrar	2736	2	FLOAT32	kVARh	R/O	numeric	---	kVARh Q3	KVARH
Reactive Energy in Quadrar	2738	2	FLOAT32	kVARh	R/O	numeric	---	kVARh Q4	KVARH
Apparent Energy in Quadra	2740	2	FLOAT32	kVAh	R/O	numeric	---	kVAh Q1	KVAH

Figura 3. 4 Mapa Modbus del medidor de energía PM3255

3.2 CASOS PARTICULARES EN TRAMAS MODBUS

Como ya se ha explicado anteriormente, los datos correspondientes a las variables suelen ocupar 16 o 32 bits, existiendo la posibilidad de que haya datos almacenados de hasta 64 bits. Los registros pueden ocupar un máximo de 16 bits. Por tanto, en caso de disponer de una entrada o una salida que ocupe 32 bits, se mandará en dos registros de 16 bits que se unirán posteriormente. La unión puede realizarse mediante dos métodos :

- Big Endian: en este caso el bit más significativo, es decir, el primer bit del primer registro, es el que se almacenará primero, uniéndose a continuación con el primer bit del segundo registro
- Little Endian: en este caso, se almacenará primero el bit menos significativo, es decir, el último bit del segundo registro, y a continuación se almacenará el primer bit del primer registro [18].

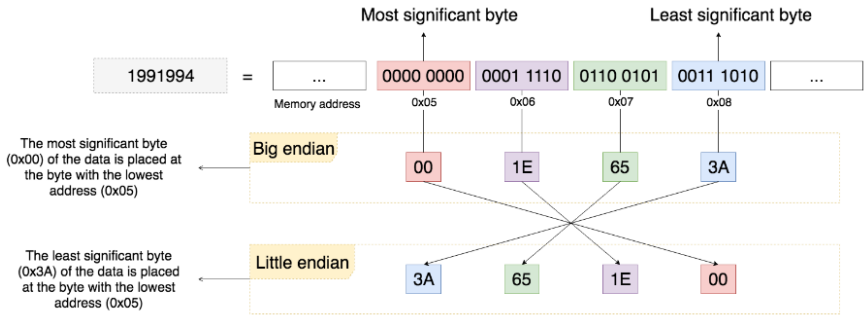


Figura 3. 5 Big Endian vs Little Endian [27]

En caso de que los datos ocupen 64 bits el método de operar será el mismo, pero en vez de ocupar dos registros, ocupará 4 registros.

3.3 DIRECCIONES ABSOLUTAS Y OFFSET

Como ya se ha mencionado anteriormente, según el tipo de variable que queramos almacenar (salida o entrada, digital o analógica), debemos apuntar hacia distintas divisiones de la memoria. Por defecto, el código de función y los mapas ya suele contar con estas divisiones, así que, si queremos acceder a un Output Register (salida analógica), el cliente Modbus empezará a solicitar el registro indicado, pero a partir del 40 000 o 40 001. Según si se suma ese uno o no, se establecerá un offset de -1 a la hora de solicitar los registros definidos en el mapa.

Por ejemplo, si deseamos leer el registro 1350 del mapa, en caso de que el offset sea de -1, debemos apuntar realmente al 1349 para poder acceder a él y el valor absoluto del registro que estamos leyendo en realidad será el 1349 + 40001, es decir el 41 350. Mientras que, si no existe offset debemos apuntar al 1350 y el valor absoluto del registro será 1350 + 40 0000, el 41 350, lo que quiere decir que el valor absoluto no ha cambiado.

3.4 ARQUITECTURA MODBUS DEL ARMARIO DEL LABORATORIO

Los armarios de control de Schneider, dispuestos en el laboratorio cuentan con equipos y elementos de protección, medida, actuación, comunicaciones y supervisión. Proporcionan la protección requerida según las normativas, a la vez que facilitan la supervisión, gestión y mando remoto de los dispositivos que están conectados, gracias a una interfaz de comunicación.

Asimismo, combinan Modbus serie con Modbus TCP. Los equipos de medida y sensores, que asumirán el rol de esclavos se comunicarán utilizando Modbus serie, mientras que los dispositivos de actuación, protección y supervisión se conectarán utilizando el protocolo Modbus TCP. Además, el elemento principal se basa en un servidor Modbus TCP.

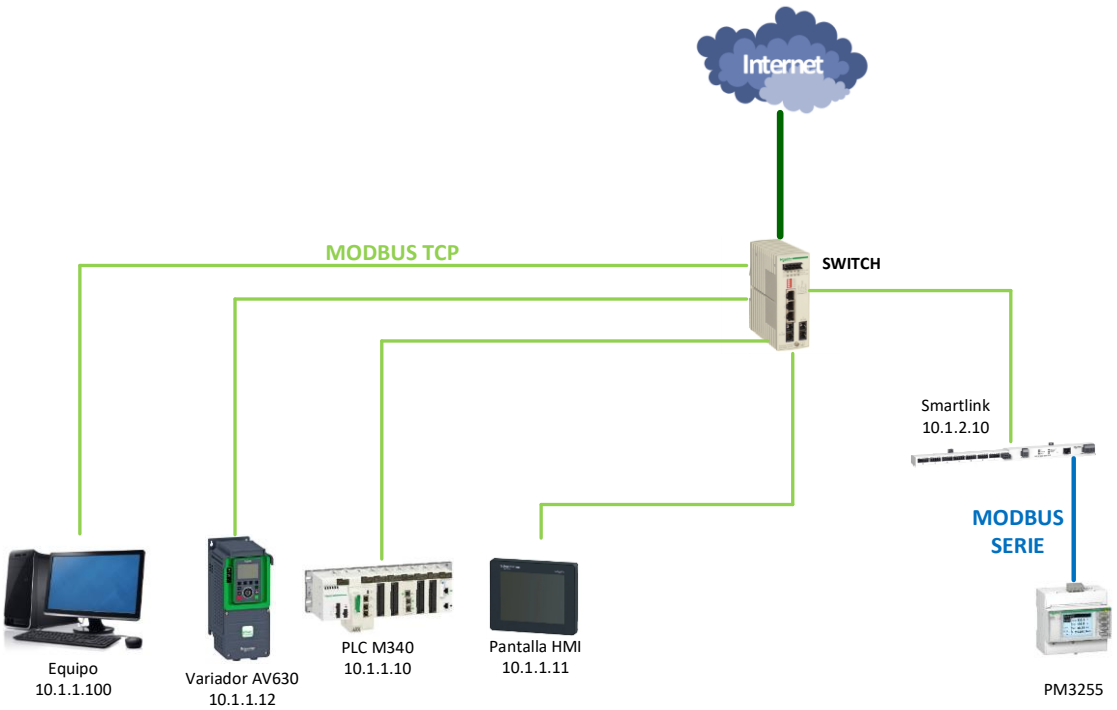


Figura 3. 6 Arquitectura del armario Schneider

4.Recursos Hardware

El objetivo del proyecto es la supervisión del variador, el autómata y el medidor de energía de uno de los armarios de control del Aula IoT (laboratorio B4) de la Escuela de Ingenierías de la Universidad de León.

Figura 4. 1 Dispositivos y Armario de Control Aula IoT

4.1 VARIADOR ALTIVAR 630

Se trata de un variador que puede alimentar motores síncronos y asíncronos trifásicos. Cuenta con tres puertos RJ45, dos puertos serie y un puerto para poder conectarlo con un cable Ethernet. Está especialmente diseñado para sistemas de gestión de fluidos y ahorro de energía. Cumple las normas internacionales IEC/EN 61800-5-1 e IEC/EN 61800-3 (inmunidad y emisiones CEM conducidas y radiadas) [28].

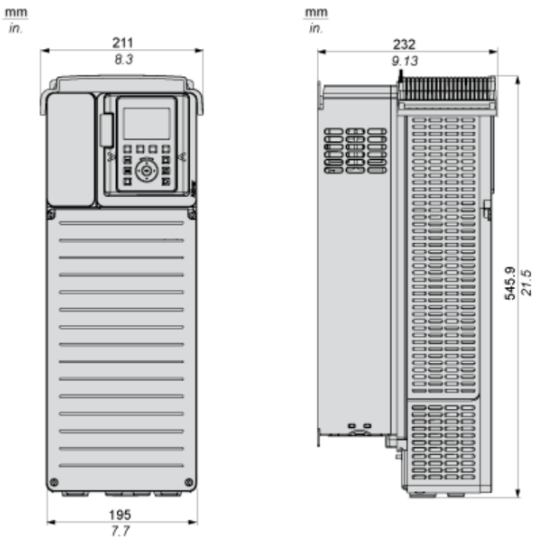


Figura 4. 2 Altivar 630 vistas frontal y lateral izquierda [28]

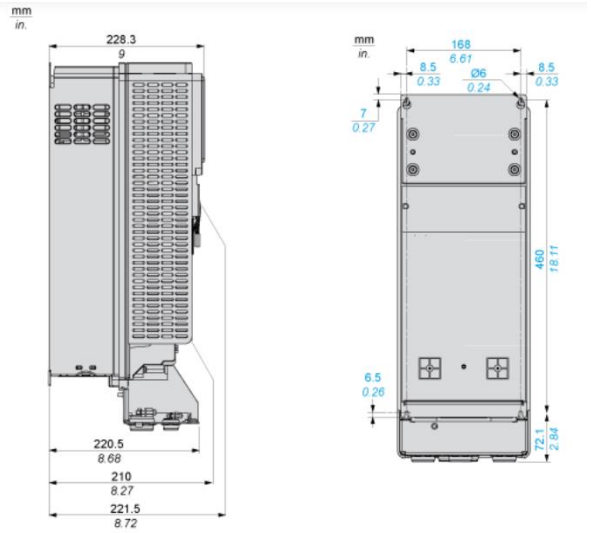


Figura 4. 3 Altivar 630 Vistas lateral izquierda y posterior [28]

Tabla 4. 1 Especificaciones Variador Altivar 630 [28]

Principal	
Gama de producto	Altivar Process ATV600
Tipo de producto o componente	Variador de velocidad
Aplicación específica de producto	Proceso y utilidades
Nombre corto del dispositivo	ATV630
Variante	Versión estándar
Destino del producto	Motores asíncronos Motores síncronos
Filtro CEM	Integrado con capacidad de sujeción: 50 m máxima corriente de conmutación acorde a EN/IEC 61800-3 categoría C2 Integrado con capacidad de sujeción: 150 m máxima corriente de conmutación acorde a EN/IEC 61800-3 categoría C3
Grado de protección IP	IP21 acorde a IEC 61800-5-1 IP21 acorde a IEC 60529
[Us] tensión de alimentación asignada	380...480 V
Grado de protección IP	UL tipo 1 acorde a UL 508C
Tipo de refrigeración	Convenc forzada
Frecuencia de alimentación	50...60 Hz - 5...5 % 380...480 V - 15...10 %
Potencia del motor en kW	22 kW - tipo de cable: carga normal) 18,5 kW - tipo de cable: carga pesada)
Potencia del motor en HP	30 hp carga normal 25 hp carga pesada
Corriente de línea	39,6 A en 380 V - tipo de cable: carga normal) 34,4 A en 480 V - tipo de cable: carga normal) 34,1 A en 380 V - tipo de cable: carga pesada) 29,9 A en 480 V - tipo de cable: carga pesada)
Corriente de cortocircuito de la red	50 kA
Potencia aparente	28,6 kVA en 480 V - tipo de cable: carga normal) 24,9 kVA en 480 V - tipo de cable: carga pesada)
Corriente de salida en continuo	46,3 A en 4 kHz para carga normal 39,2 A en 4 kHz para carga pesada
Máxima corriente transitoria	50,9 A durabilidad eléctrica 60 s - tipo de cable: carga normal) 58,8 A durabilidad eléctrica 60 s - tipo de cable: carga pesada)

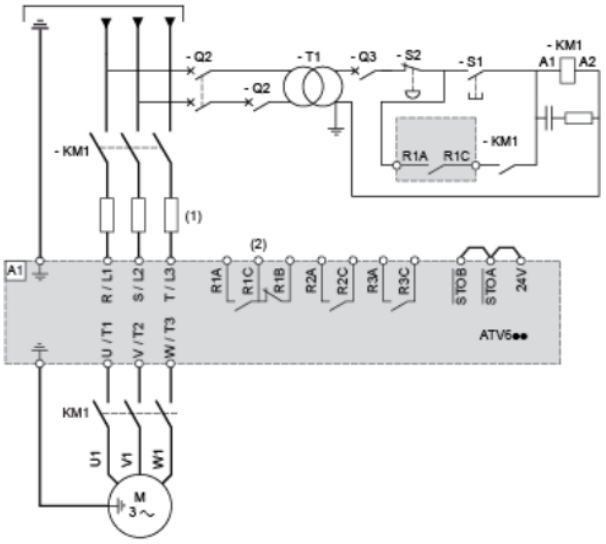


Figura 4. 4 Diagramas de conexión conforme a las normas EN 954-1 categoría 1 e IEC/EN 61508 capacidad SIL1, categoría de parada 0 según la norma IEC/EN 60204-1 [28]

4.2 MÓDULOS SCHNEIDER

La parte de control del armario se realiza mediante una serie de módulos de la gama Modicon X80 de Schneider Electric

4.2.1 MÓDULO DE ALIMENTACIÓN BMXCPS2000

Consiste en un módulo de alimentación que cuenta con una tensión primaria de 100 a 240V de corriente alterna. La alimentación secundaria será de 16,8W a 24V de corriente continua. La máxima potencia de puede disipar son 8,5W. Está equipado con un conector de dos pines para relés de alarmas, y 5 pines para toma de línea y conexión a tierra [29].

Tabla 4. 2 Especificaciones módulo de alimentación BMXCPS2000 [29]

Principal	
Gama de producto	Modicon X80 ((*))
Tipo de producto o componente	Módulo da fonte de alimentação
Backplane compatibility	Not compatible with BMEXBP...02 ((*))
Tensión primario	100...240 V
Tipo de circuito de alimentación	AC
Potência secundária	10,8 W 24 V CC fuente alimentación detector 16,8 W 24 V CC processador e fonte de alimentação do módulo E/S 8,3 W 3,3 V CC fonte de alimentação lógica do módulo E/S

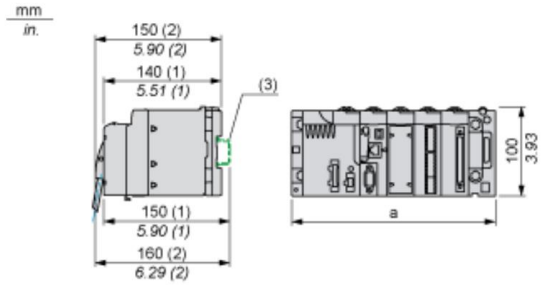


Figura 4. 5 Dimensiones módulo de alimentación BMXCPS2000 [29]

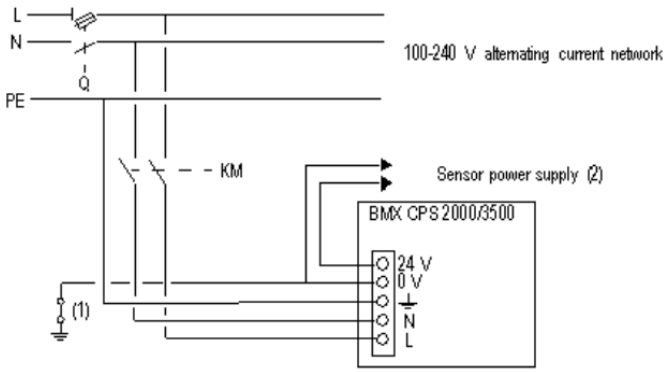


Figura 4. 6 Conexión módulo de alimentación BMXCPS2000 [29]

4.2.2 MÓDULO DE PROCESAMIENTO BMXP342020

Este módulo cuenta con un conector de tipo RJ45 que permite la conexión a Ethernet o Modbus TCP, y otro conector del mismo tipo (RJ45) para la conexión a Modbus serie. Funciona duplicando la información proporcionando un respaldo de los datos cuando el PLC está apagado [30].

Tabla 4. 3 Especificaciones módulo BMXP342020 [30]

Principal	
Gama de producto	Plataforma autom. Modicon M340
Tipo de producto o componente	Módulo do processador
Concepto	Transparent Ready CANopen
Número de racks	4
Número de ranuras	11
Capacidade do processador de E/S digital	1024 E/S configuração de multirrack 704 E/S configuração de rack simples
Capacidade do processador de E/S analógica	256 E/S configuração de multirrack 66 E/S configuração de rack simples
Número de canal específico da aplicação	36
Monitorización	Contadores de diagnóstico Modbus Contadores de eventos Modbus

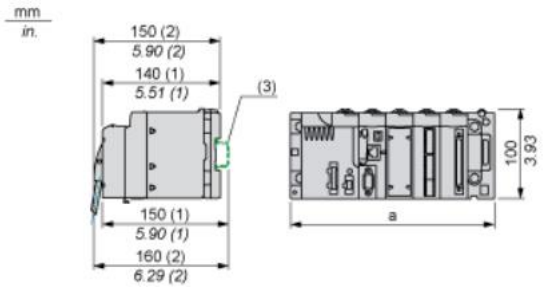


Figura 4. 7 Dimensiones módulo BMXP342020 [30]

4.2.3 MÓDULO DE ENTRADAS Y SALIDAS BMXAMM0600

Es el módulo al que cablearemos las entradas analógicas necesarias para el proceso de control del motor. Está compuesto por 4 canales de entrada analógica y 2 canales de salida analógica [31].

Tabla 4. 4 Especificaciones módulo BMXAMM0600 [31]

Principal	
Gama de producto	Modicon X80 ((*))
Tipo de producto o componente	Módulo de E/S analógica mista
Consecutivo, seguido, continuo, adosado	20 vías 1 conector
Isolation between channels	Não isolada
Nivel de entrada	Nivel alto
Número de entrada analógica	4
Tipo de entrada analógica	Corriente 0...20 mA Corriente 4...20 mA Tensión +/- 10 V Tensión 0...10 V Tensión 0...5 V Tensión 1...5 V

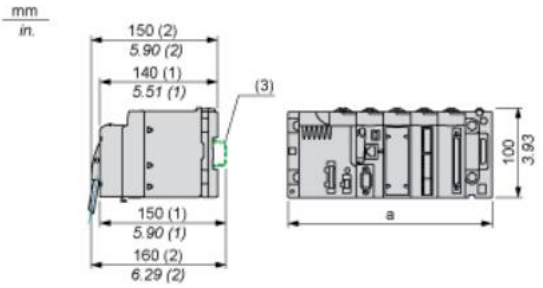


Figura 4. 8 Dimensiones módulo AMM0600 [31]

4.2.4 MÓDULO DE ENTRADAS Y SALIDAS BMXDDM3202K

Este módulo de entradas y salidas digitales suministra 24V de corriente continua. Ofrece 16 canales de entrada y 16 canales de salida [32].

Tabla 4. 5 Especificaciones módulo BMXDDM3202K [32]

Principal	
Gama de producto	Modicon X80 ((*))
Tipo de producto o componente	Módulo E/S discreta
Número de entrada digital	16
Tipo de entrada	Coletor de corriente (lógica positiva)
Voltaje entrada	24 V CC positiva
Corriente de entrada discreta	2,5 mA
Número de salidas discretas	16
Tipo de salida digital	Estado sólido
Tensión de salida	24 V 19...30 V CC
Corriente de salida digital	0.1 A

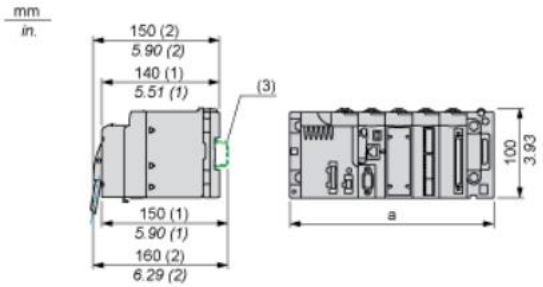


Figura 4. 9 Dimensiones módulo BMXDDM3202K [32]

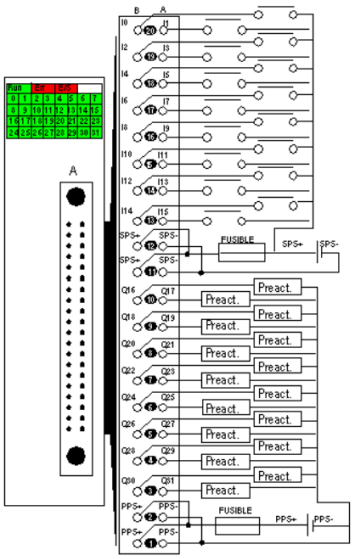


Figura 4. 10 Conexión módulo BMXDDM3202K [32]

4.3 MEDIDOR DE ENERGÍA METSEPM3255

El equipo mide potencia activa y reactiva, potencia aparente, factor de potencia, intensidad, tensión, frecuencia, distorsión armónica total y armónicos hasta el orden 15. Proporciona una precisión de clase 0,5S (norma IEC 62053-22) y 32 muestras por ciclo [33].

Tabla 4. 6 Especificaciones medidor de energía METSEPM3255 [33]

Principal	
Gama	PowerLogic
Nombre del producto	PowerLogic PM3000
Nombre abreviado del equipo	PM3255
Tipo de producto o componente	Central de medida
Complementario	
Análisis de calidad de energía	Hasta armónico 15
Función	Multi-tarifa Supervisión de potencia Facturación sub
Tipo de medición	Potencia activa y reactiva Potencia aparente Corriente Tensión Energía Factor de potencia Frecuencia Distorsión armónica de corriente total THD(I) Distorsión armónica de tensión total THD(U)
Supply voltage	100...277 V AC 45-65 Hz 173...480 V AC 45-65 Hz 100...300 V corriente continua
Frecuencia de red	60 Hz 50 Hz
[In] Corriente nominal	5 A 1 A
Type of network	3P 3P + N 1P + N
Consumo de potencia en W	5 VA
Tipo de pantalla	LCD retroiluminada
Resolución de la pantalla	128 x 96 pixels
Velocidad de muestreo	32 muestras/ciclo
Corriente de medición	0,02...1,2 A 0,05...6 A
Tipo de entrada analógica	corriente 0...5 A corriente 0...1 A
Tensión de medida	50...330 V AC 45-65 Hz directo 50...330 V AC 45-65 Hz fase a neutro 80...570 V AC 45-65 Hz directo 80...570 V AC 45-65 Hz fase a fase

Figura 4. 11 Medidor de energía METSEPM3255 [33]

4.4 ADAPTADOR DE RED

La red a la que está conectado el armario de control se conecta al ordenador por medio de un cable Ethernet. Para poder compartir esta red y permitir que otros dispositivos, como por ejemplo la Tablet, puedan acceder a esta red, necesitamos conectar un adaptador de red al equipo, para poder realizar este tipo de comunicación.

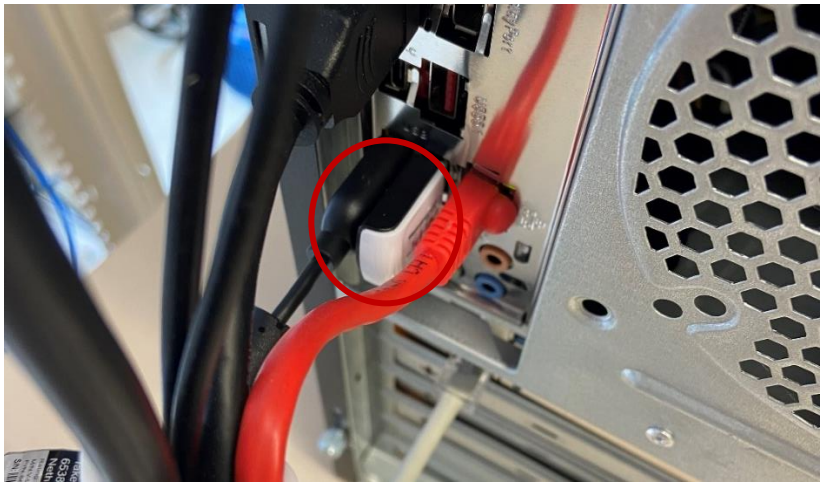


Figura 4. 12 Adaptador de red

5.Recursos Software

5.1 UNITY 3D

Unity 3D es una plataforma de desarrollo que habitualmente se ha utilizado en el campo de los videojuegos, pero que, gracias al reciente crecimiento y aplicación de la realidad aumentada cada vez más en ámbitos científicos y tecnológicos, se está abriendo un hueco en estos sectores.

Incluye hasta 8 productos en un mismo entorno; algunos ejemplos son : Unity, Unity Pro, Asset Server, IOS, IOS Pro, Android y Android Pro. La mayor ventaja que presenta es que permite importar recursos y objetos en forma de paquetes, por lo que la función especial de AR podrá ser importada ; en este caso utilizaremos otra plataforma Vuforia que nos proporciones dichas funciones. Además, los «scripts» son compatibles con numerosas plataformas, lo que permite desarrollar en un mismo tiempo programas para Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone y Android. Por último, su tiempo de ejecución es tres veces más rápido que el lenguaje Java tradicional [34]. Todos estos motivos han llevado a la determinación de que Unity 3D es la plataforma más apropiada para llevar a cabo el desarrollo del proyecto de manera óptima.

El código básico de Unity se basa en dos partes funcionales principales, en las que podremos incluir nuestros programas que irán definidos como “clases”. En “start” incluiremos los programas que solo queremos que se realicen una vez, al principio, nada más arrancar la máquina. En “update” podemos incluir todas aquellas funciones que queramos que el programa realice cíclicamente, es decir un bucle. Estas dos funcionalidades vendrán incluidas en una clase heredada de “Monobehaviour” para que el script pueda ser reconocido y procesado por Unity.

Unity dispone la arquitectura del proyecto en árbol, que está compuesto por ramas principales que actúan como “padres” de las ramas secundarias que les suceden que se denominarán “hijos”, a su vez estas ramas secundarias harán de “padres” en caso de que le sucedan otros “hijos”.

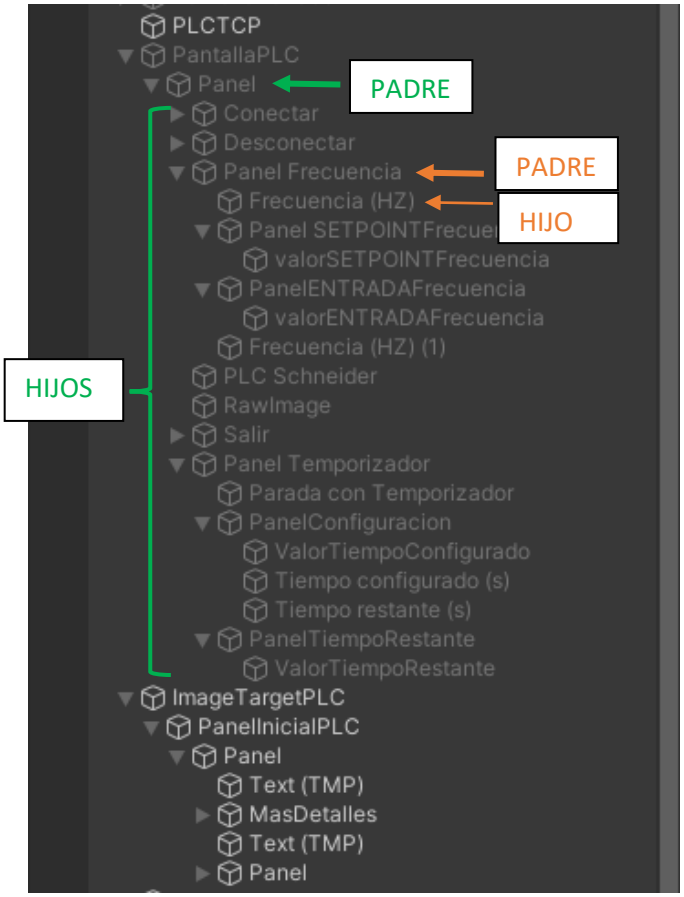


Figura 5. 1 Árbol del proyecto del PLC

5.2 VUFORIA

El kit de software para el desarrollo de AR de Vuforia (Vuforia AR SDK) ha sido diseñado por Qualcomm para la implementación de realidad aumentada en aplicaciones móviles. Identifica y captura de manera instantánea imágenes planas y objetos en tres dimensiones sencillos (como pueden ser cubos o esferas). A continuación, permite al usuario colocar objetos virtuales en esta cámara de AR y ajustar su posición sobre la lente [35].

La aplicación típica de AR se basa en un primer proceso on-line y a continuación un proceso off-line. La parte off-line cubre la selección y tratamiento de datos y objetos, generando un modelo de comportamiento y creando un escenario 3D. La parte on-line incluye la identificación de objetos, devolviendo la información correspondiente al momento en el escenario real. El objeto virtual es añadido en la escena, llevando a cabo la interacción humano-máquina [36]. En este caso, Vuforia se encarga de la parte on-line, implementando una cámara AR, mientras que Unity3D se encargará de complementar la parte off-line.

Vuforia se puede implementar fácilmente como una librería en Unity. Recientemente el propio Unity ha implementa su propia aplicación de AR, ARFoundation. Aun así, se ha considerado Vuforia para la implementación del proyecto, ya que pertenece a la empresa PTC, la cual podemos comprobar en la documentación del programa que está más orientada al desarrollo de aplicaciones y nueva tecnología, mientras Unity continúa siendo una herramienta más dirigida al desarrollo de videojuegos. Además, Unity y Vuforia llevan trabajando juntos bastantes años, lo cual otorga una cierta seguridad y fiabilidad frente a una tecnología más reciente.

5.3 VISUAL STUDIO BASIC

Visual Studio es un entorno de desarrollo de programación. Es uno de los IDE más completos para el desarrollo de .NET y C++, compatible con Windows y MACos. Permite editar, depurar y compilar código. Será el IDE (Entorno de Desarrollo Integrado) utilizado para editar los "scripts" en lenguaje C# creados en Unity. Utilicé la última versión gratuita disponible, Visual Studio Community 2022 17.5.5

5.4 SIMULADOR RMMS

El simulador Radzio Mobus Master, es una herramienta que permite establecer conexiones Modbus con diferentes dispositivos conectados en la misma red. Permite comprobar la comunicación que estamos realizando con el variador y, que los datos a los que estamos accediendo son correctos.

5.5 UNITY PRO XL

Es el software de programación y control de los autómatas y módulos de la gama de Modicon X80 de Schneider Electric. Sigue el estándar IEC 61131-3 [37]. Se emplea, además de para programar el proceso de control del motor, para poder visualizar los espacios de memoria asignados a cada variable, y, por tanto, conocer la dirección de registro a la que necesitaremos acceder para leer los datos de las variables del proceso.

5.6 TECNOLOGÍA QR

Los códigos QR pueden ser considerados la evolución de los códigos de barras, proporcionando, además, mayor capacidad de almacenamiento de la información. Consisten en una matriz de puntos que codifica y almacena una serie de datos [38]. Estos códigos permiten su fácil impresión y lectura desde la cámara de la mayoría de los dispositivos móviles [39].

Para el reconocimiento, a través de la interfaz de Vuforia, de los distintos elementos que se quieren supervisar del armario de control de Schneider, utilizaremos códigos QR, ya que, a diferencia de las imágenes, estos presentan un mayor contraste y son más fáciles de distinguir a través de la cámara de Realidad Aumentada. Esto favorece que el funcionamiento de la aplicación no se vea comprometido por la calidad de las lentes del dispositivo que se utilice, ya que, si se empleasen imágenes, según las condiciones de luz y la calidad de la cámara, se podría dificultar el reconocimiento de los distintos elementos.

Además, el utilizar códigos QR en lugar de un reconocimiento puramente por imágenes, no compromete la escalabilidad del proyecto, evitando posibles solapamientos o interferencias en caso de que existan dos elementos de igual modelo y fabricante, ya que los valores de las variables a controlar serán distintos, dado que a cada componente se le asigna un código QR único. Esto permite poder utilizar la misma aplicación para la supervisión de diferentes armarios de control.

6.Desarrollo de la aplicación

6.1 INSTALACIÓN DEL SOFTWARE

Para el desarrollo del proyecto necesitamos instalar el entorno de Unity 3D. Instalaremos la versión 2021.3.2lf1, que se trata de la última versión LTS (Long Term Suport) disponible actualmente.

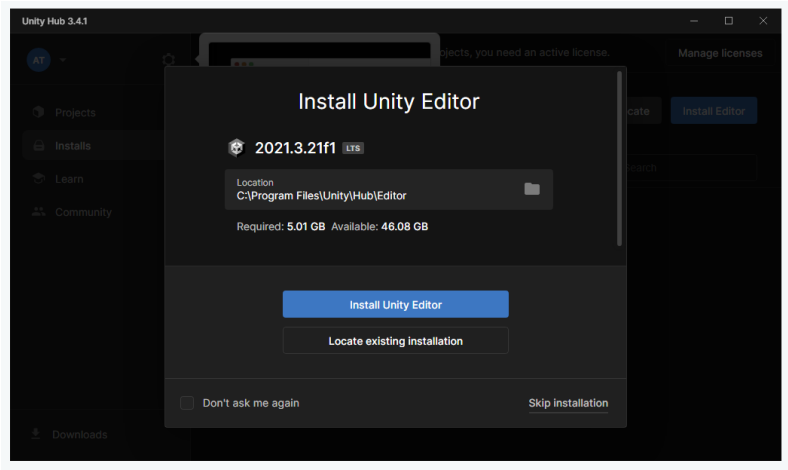


Figura 6. 1 Versión de Unity3D Instalada

Una vez se disponga del Unity Hub, procedemos a descargar el Vuforia Engine, que será el encargado de ejecutar la parte de AR, a través de la cámara de la que dispone. Instalaremos la última actualización que se trata de Vuforia Engine 10.14

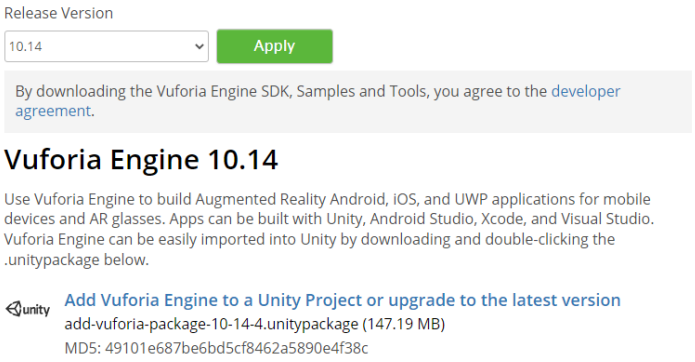


Figura 6. 2 Versión Vuforia Instalada

Para poder añadirlo a nuestro proyecto, lo haremos a través de Assets (que se encuentra en el menú superior), “Import Package”; si creamos un nuevo paquete nos permitirá añadir el archivo de Vuforia que hemos descargado previamente.

Una vez disponemos de Vuforia, si pulsamos con el botón derecho en el árbol, nos aparecerá una opción llamada “Vuforia Engine” la cual nos permite añadir la AR cámara. Una vez añadida la cámara, debemos configurarla; para ello nos pedirá una licencia, que podemos obtener de manera sencilla y gratuita (en caso de seleccionar la básica) a través del portal de desarrollo del software. También de manera opcional, y desde el mismo portal, seleccionándolo en el menú “Target Manager”, podremos obtener una base de datos para poder importar objetos que serán reconocidos por dicha cámara. Por el momento, omitimos este paso.

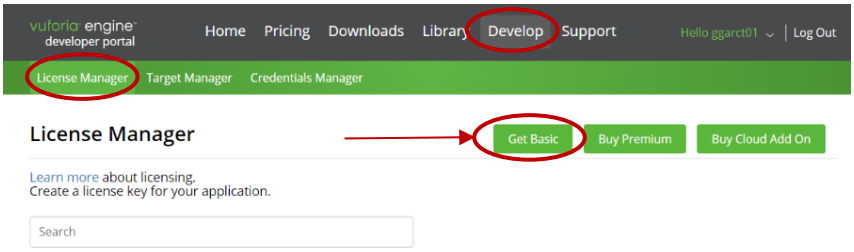


Figura 6. 3 Obtener una licencia en Vuforia

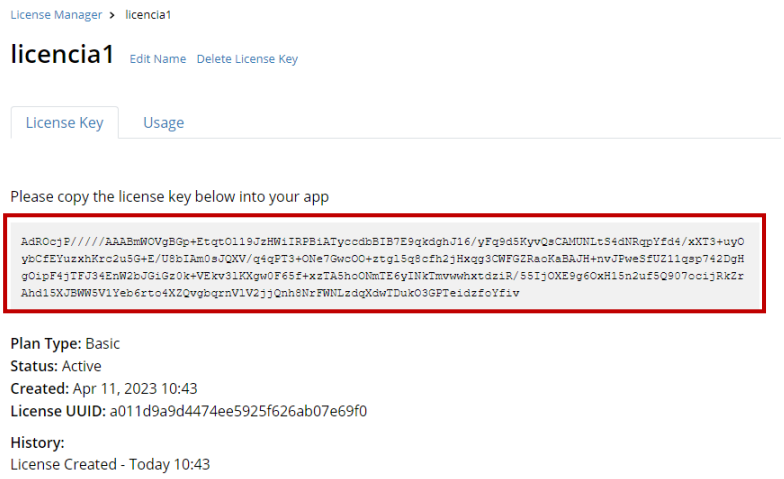


Figura 6. 4 Licencia obtenida en Vuforia

Copiamos la licencia obtenida en el portal de desarrollo en la configuración de la cámara

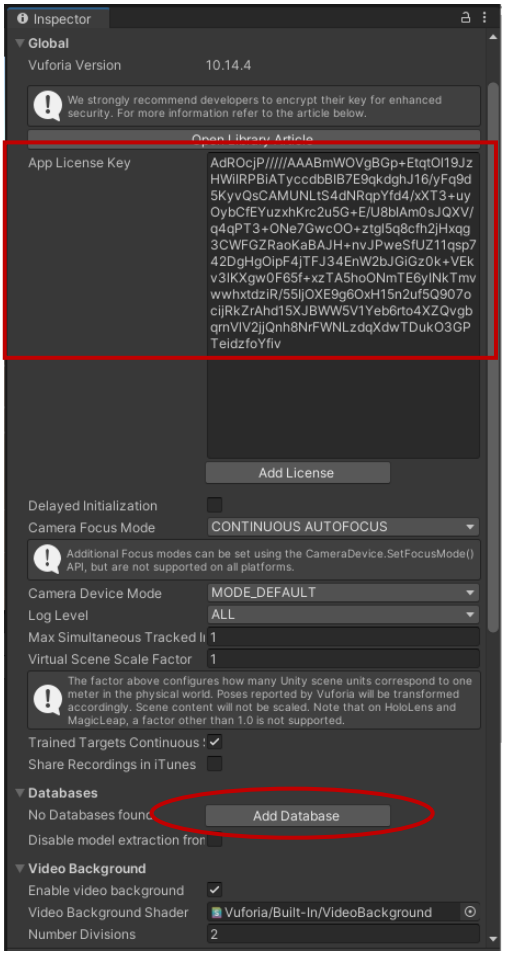


Figura 6. 5 Añadir licencia y base de datos de Vuforia en Unity

6.2 IMPLEMENTACIÓN EASYMODBUS

Para que Unity pueda procesar el script en #C con la librería Easymodbus, necesitamos implementar el DLL correspondiente.

Primero debemos comprobar que, en la configuración del proyecto, el API sea compatible con .NET

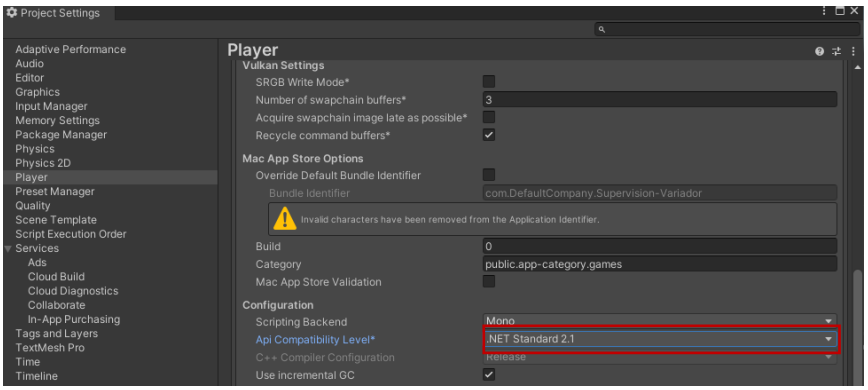


Figura 6. 6 Compatibilidad con .NET en Unity

A continuación, crearemos una carpeta de Plugins dentro de la raíz (Assets) del proyecto. Dentro de esta carpeta añadiremos el DLL que descargaremos de la propia página web de EasyModbus. En realidad, se podría omitir el paso de crear una carpeta de Plugins y añadirse directamente en "Assets", pero siempre será más recomendable hacerlo de cara a trabajar con un proyecto de manera más ordenada.

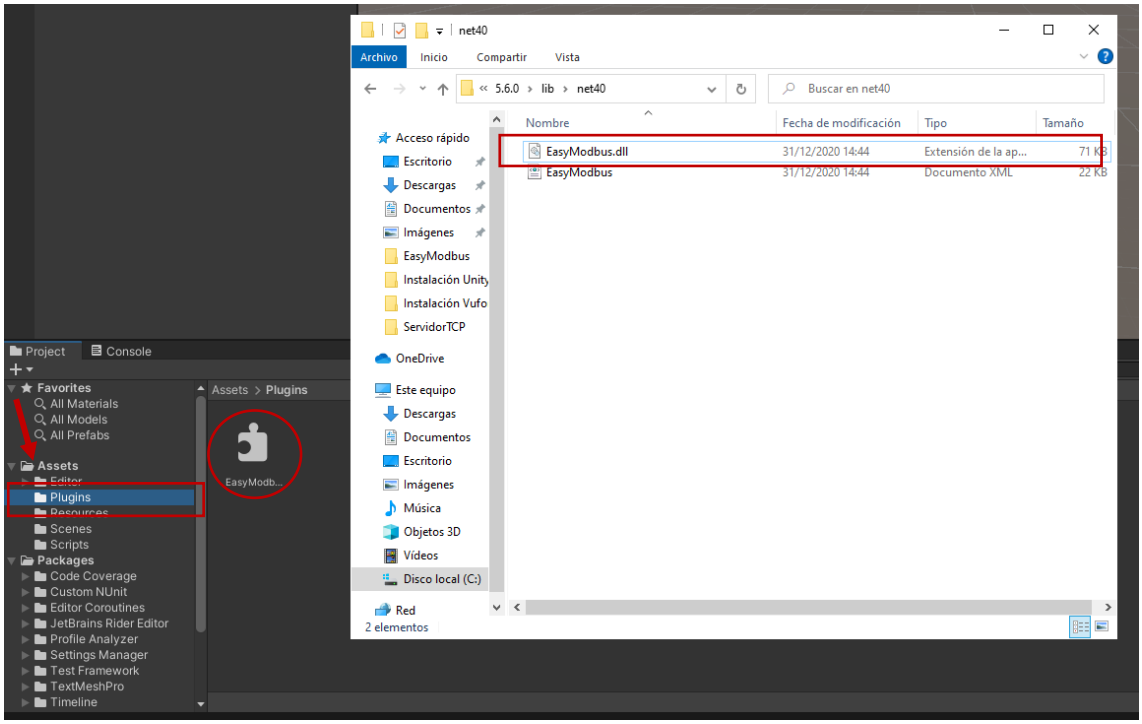


Figura 6. 7 Añadir el DLL de EasyModbus

Una vez añadido el DLL, ya estará disponible para poder trabajar con él en Unity3D.

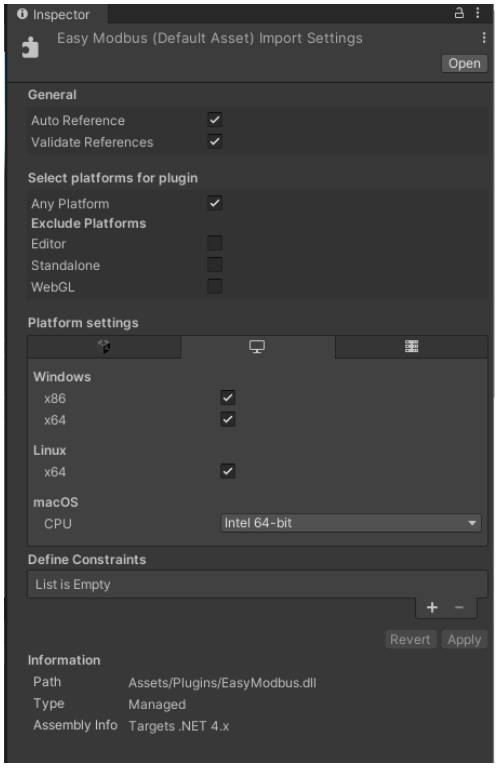


Figura 6. 8 DLL EasyModbus

6.3 PRUEBA DE COMUNICACIÓN INICIAL CON EL VARIADOR

Como ya se ha mencionado anteriormente, para realizar el código nos ayudaremos de la librería "EasyModbus". Esto debemos declararlo al principio del código con:

```
using EasyModbus;
```

Antes de comenzar definiremos las variables fuera de las funciones start y loop. Esto es importante para que se creen de manera global y no local, y para no estar redefiniéndolas cada vez que se ejecute un bucle del programa.

Luego en la función "Start" crearemos un nuevo cliente con la dirección IP y número de puerto almacenados en las variables que hemos definido previamente.

```
ModbusClient modbusClient = new ModbusClient(IP, 502);
```

A continuación, definiremos la dirección ID de nuestro cliente. Esto es importante, ya que por defecto la librería Easymodbus utilizará el número "1" como ID.

```
modbusClient.UnitIdentifier = direccionID;
```

Después comenzaremos a trabajar en "update". Primero estableceremos la conexión. Una vez establecida la conexión comenzaremos a leer registros con:

`int[] readHoldingRegisters =modbusClient.ReadHoldingRegisters(dirección, registros);`

En "dirección" especificaremos la dirección del registro al que queremos acceder, y en "registros" definiremos el número de registros que queremos leer. Luego procederemos a imprimir por pantalla los registros leídos, utilizando un bucle que recorra toda la cadena de datos que hemos creado anteriormente (`int[] readHoldingRegisters`) y con la función `Debug.log`. Por último, cerraremos la conexión. Esta parte del programa se ejecutará de manera cíclica hasta que se cierre y apague la aplicación.

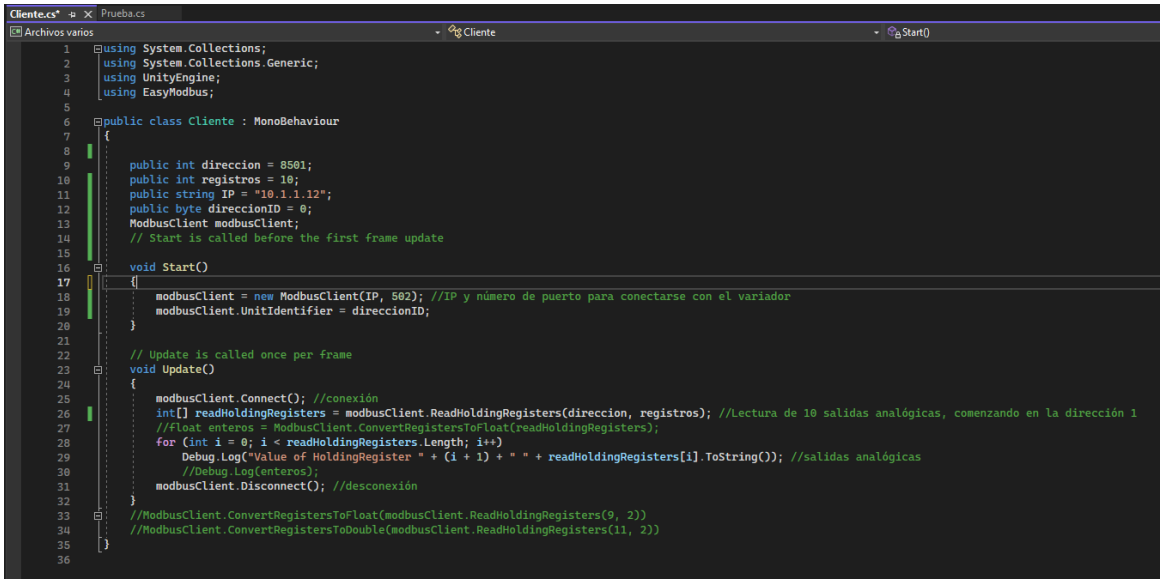


Figura 6. 9 Código de prueba de comunicación modbus

6.3.1 COMPROBAR COMUNICACIÓN

Para comprobar que la comunicación se está realizando de manera correcta, utilizaremos el simulador RMMS para simular un cliente Modbus TCP que lea los mismos registros que pretendemos leer con el código. En esta prueba, pondremos que nos lea 10 registros comenzando por el 8500.

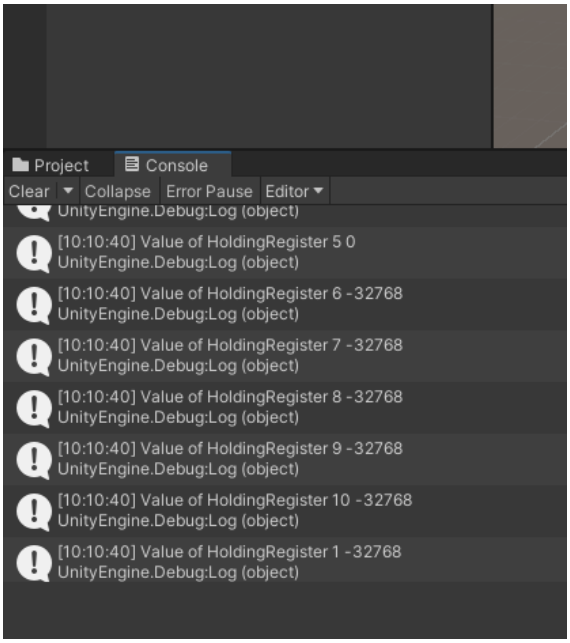


Figura 6. 10 Registros leídos por Unity

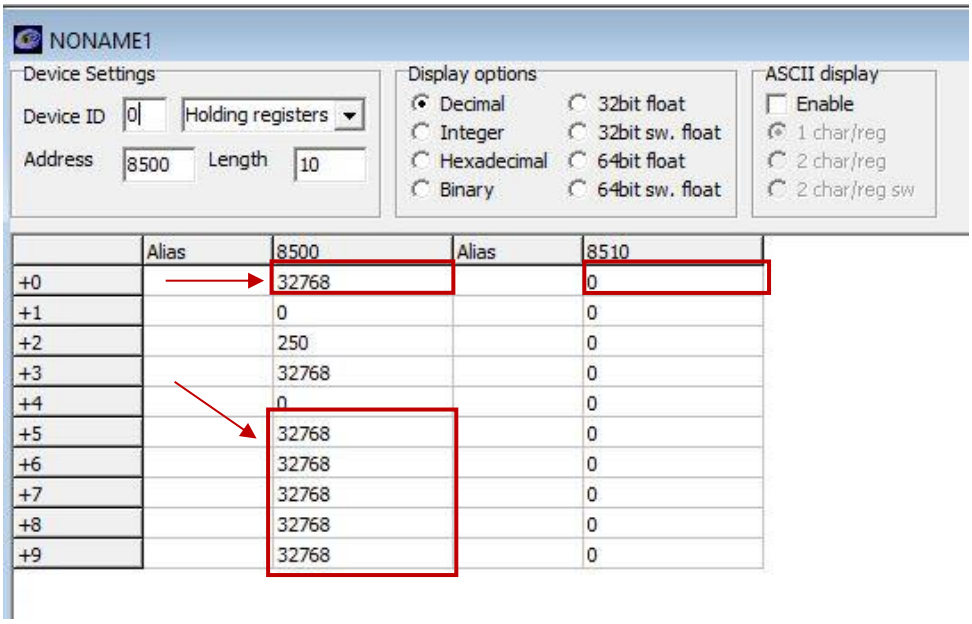


Figura 6. 11 Registros capturados por RMMS

Queda comprobado que los datos que se imprimen por la consola de Unity coinciden con los datos leídos por el simulador RMMS, por tanto, la comunicación con el variador se está estableciendo de manera correcta.

6.4 DISEÑO DE LA APLICACIÓN

Para crear la aplicación, primero crearemos un nuevo "canvas" en el árbol de nuestro proyecto, cuya escala ajustaremos al tamaño de la pantalla, para que se ajuste de manera

independiente a la pantalla del dispositivo que queramos conectar. A continuación, añadiremos un panel, en el cual podemos seleccionar el color de fondo de nuestra aplicación.

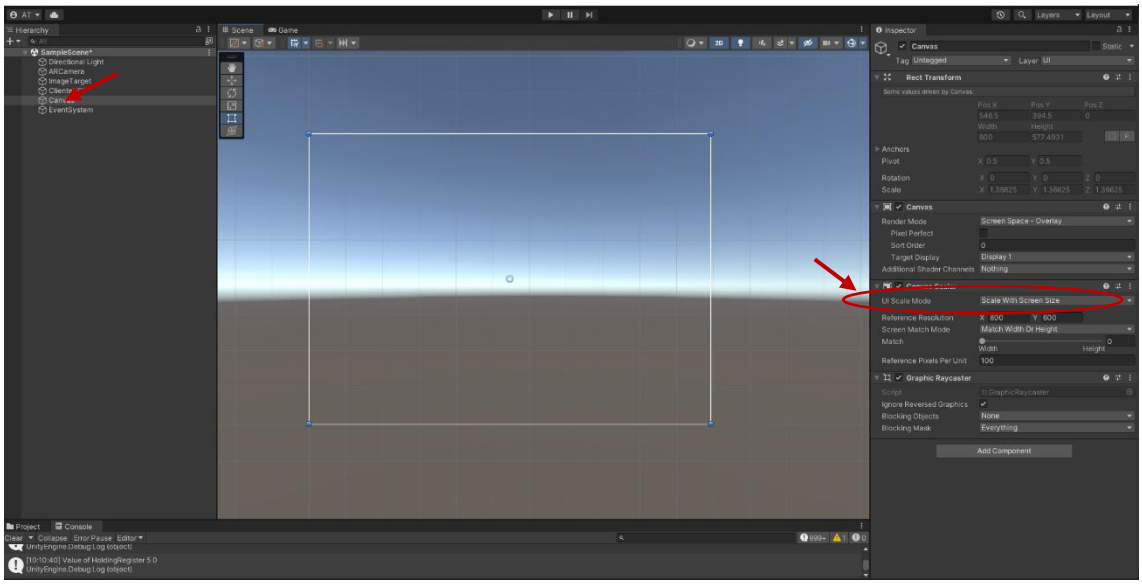


Figura 6. 12 Añadir canvas y escalarlo con la pantalla

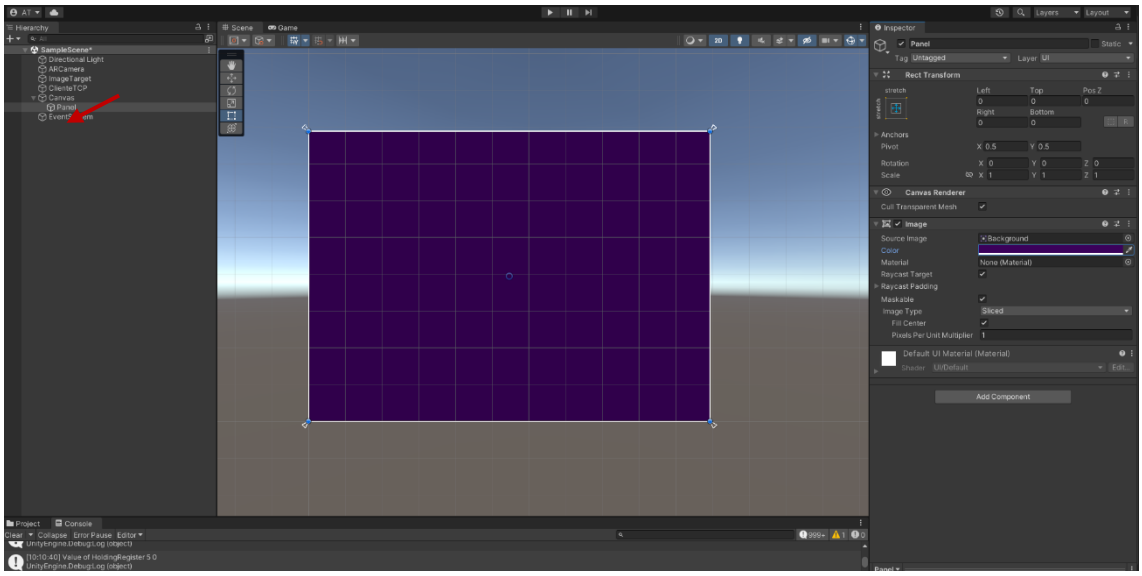


Figura 6. 13 Añadir panel

Luego añadiremos dos botones de conexión y desconexión. Unity nos facilita esta herramienta, que podemos añadir desde el panel de UI en el desplegable que aparece si hacemos “click” en el botón derecho del ratón sobre el árbol del proyecto. También podemos añadir un cuadro de texto que indique frecuencia y otro panel más pequeño sobre el que se resalte el texto dónde indicaremos el valor que queremos imprimir por pantalla en la aplicación. Es importante tener bien localizado dónde se encuentra este segundo cuadro de

texto en la estructura del árbol, ya que más adelante lo necesitaremos relacionar con la variable que creemos en el script para poder imprimir la lectura realizada por pantalla.

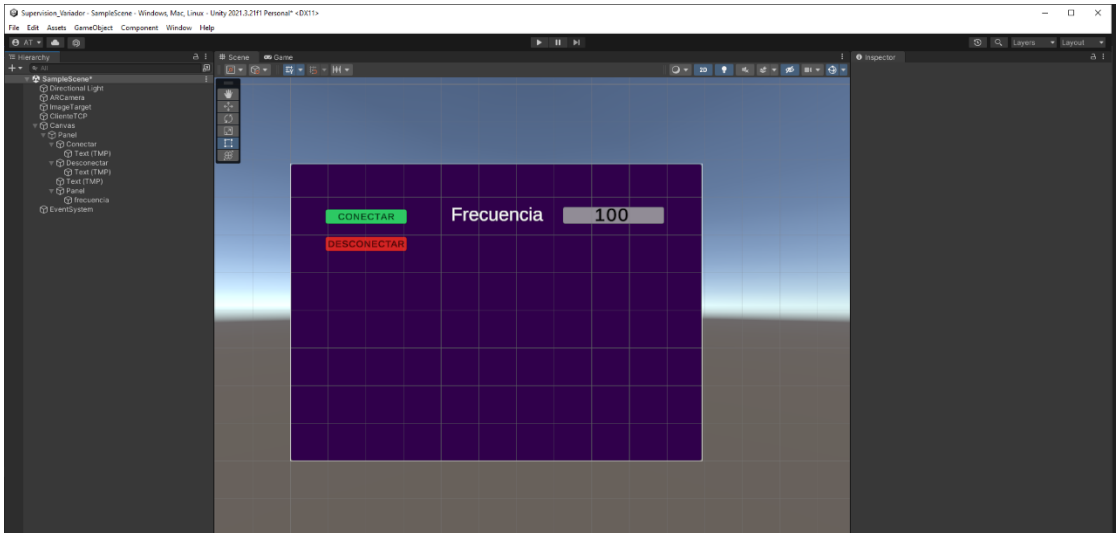


Figura 6. 14 Añadir botones y cuadros de texto

Podemos importar una imagen en la raíz de nuestro proyecto, es decir en “Assets”, en la carpeta de “Resources”, con la opción de “import new assets” que nos muestra al hacer clic en el botón derecho del ratón y seleccionando el directorio de destino dónde tengamos descargada la imagen que queremos añadir.

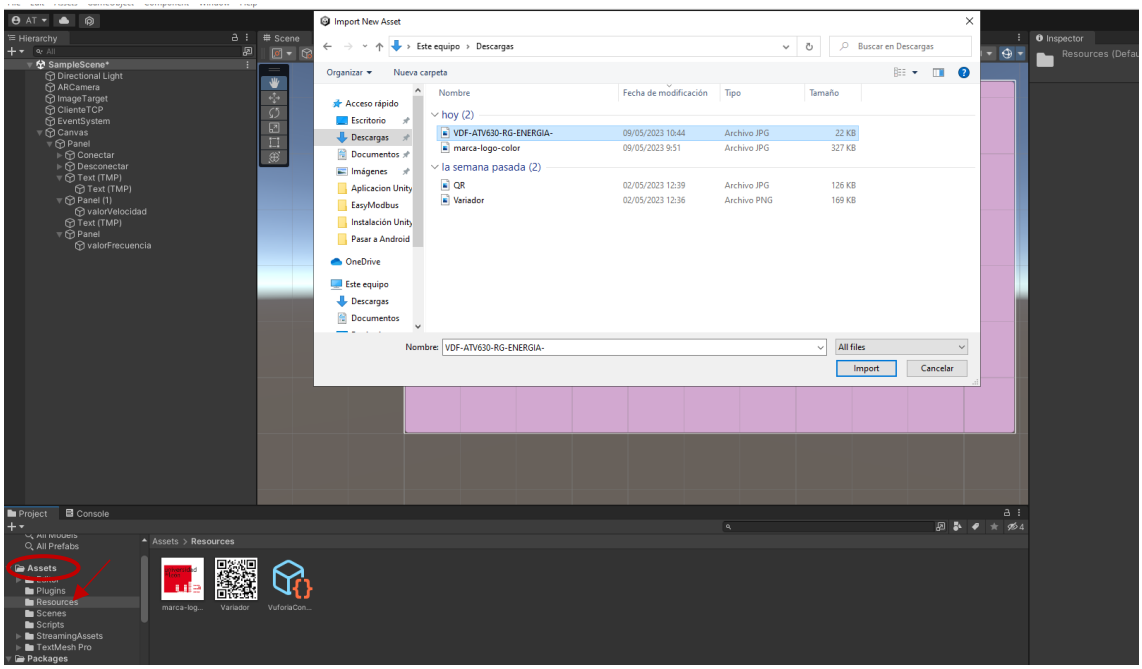


Figura 6. 15 Importar imágenes a la zona de trabajo

Añadiendo más cuadros de texto y paneles con la información que queramos representar, el resultado final para la pantalla que mostraremos cuando se enfoque al QR del variador será la siguiente.

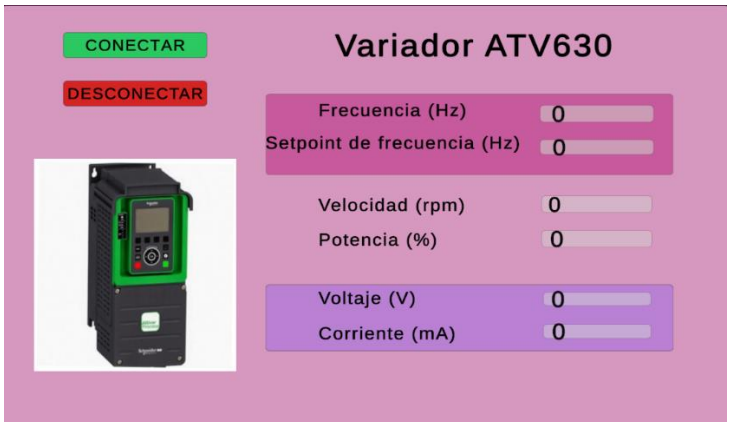


Figura 6. 16 Diseño de la pantalla final para el variador

Operando de manera similar crearemos las pantallas para el PLC y el medidor de energía.

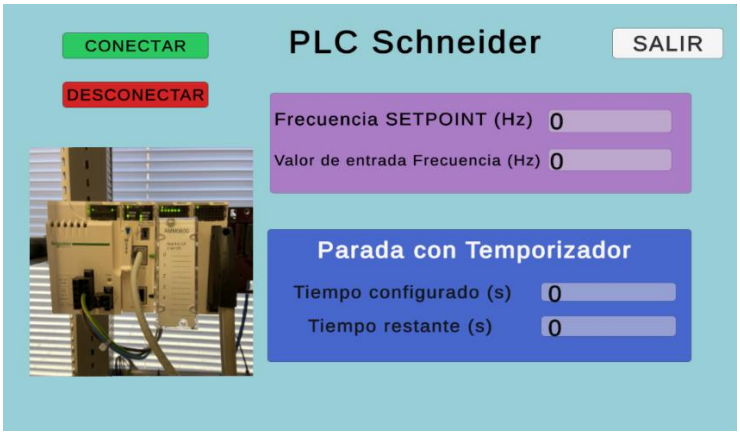


Figura 6. 17 Diseño de la pantalla final para el autómata

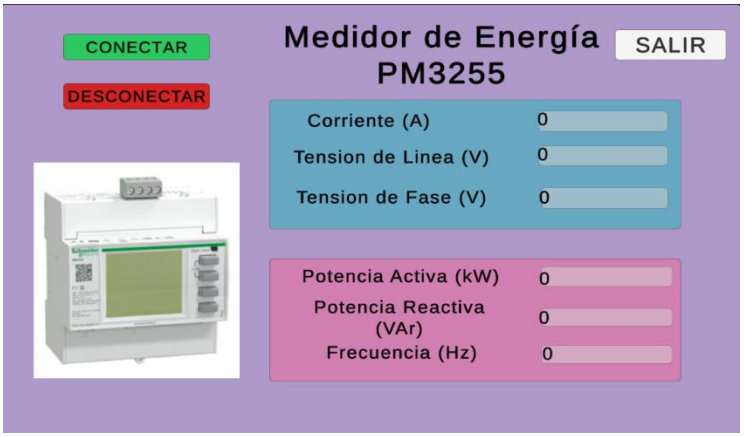
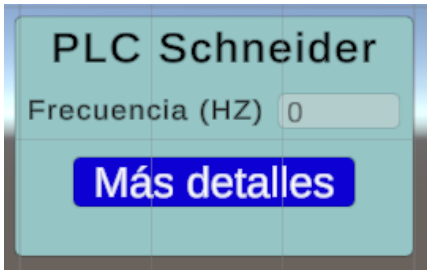


Figura 6. 18 Diseño de la pantalla final para el medidor de energía

Procediendo de la misma manera, podemos crear unos menús iniciales de menor tamaño, que se muestren cuando aparezca el QR del elemento deseado, en el que aparezca una breve descripción y alguna variable que resulte fundamental. En este menú podemos ofrecer al usuario la opción de "ver más detalles", en la que desplegará las pantallas finales (mostradas anteriormente) con todas las variables del proceso que se necesitan conocer. Lo haremos creando un panel más pequeño en otros canvas (siempre escalados al tamaño de la pantalla), y añadiendo botones y cuadros de texto de igual forma que ya se ha explicado anteriormente.

Figura 6. 19 Menú inicial para el QR del variador



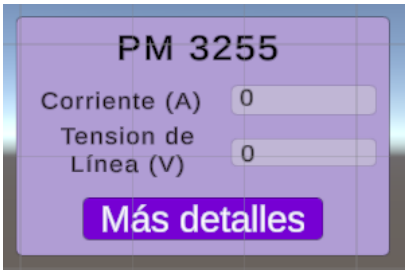


Figura 6. 20 Menú inicial para el QR del PLC

Figura 6. 21 Menú inicial para el QR del medidor de energía

6.5 CÓDIGO FINAL

El código final realizado en C# para poder comunicar la aplicación con el variador es el siguiente

6.5.1 LIBRERÍAS Y VARIABLES

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

public class Variador : MonoBehaviour
{

    public int DireccionSETFrecuencia = 8502;
    public int DireccionVALORFrecuencia = 3202;
    public int DireccionVelocidad = 8603;
    public int DireccionPotencia = 3211;
    public int DireccionVoltaje = 3208;
    public int DireccionCorriente = 3204;
    public string IPVariador = "10.1.1.12";
    public byte VariadorID = 0;
    public TMP_Text VALORFrecuencia;
    public TMP_Text SETFrecuencia;
    public TMP_Text ValorVelocidad;
    public TMP_Text ValorPotencia;
    public TMP_Text ValorVoltaje;
    public TMP_Text ValorCorriente;
    ModbusClient modbusClient;
    int[] RegistroSETFrecuencia;
    int[] RegistroVALORFrecuencia;
    int[] RegistroVelocidad;
    int[] RegistroPotencia;
    int[] RegistroVoltaje;
    int[] RegistroCorriente;
    int boton_conectar;
```

Figura 6. 22 Librerías y variables en el código final del variador

Además de las librerías explicadas anteriormente, en la prueba de conexión al variador, declaramos también las librerías “UnityEngine.UI” y “TMPro”, que nos permitirán utilizar las funciones necesarias para configurar los cuadros de texto que queramos imprimir por pantalla. Antes de comenzar, declararemos todas las variables que vamos a emplear, para que se creen de manera global y no sólo sean variables locales de las funciones “start” y “update”. Las primeras seis variables corresponden a los números de los registros que contienen la información que queremos leer. A continuación, le precede una variable con el número de IP y el ID del variador, imprescindibles para poder realizar su conexión con Modbus. Luego otras series variables de tipo “TMP_text” que son en las que almacenaremos los valores de las lecturas realizadas que queramos imprimir por la pantalla. También añadiremos una variable con un cliente Modbus, para poder realizar la conexión con el variador. Por último, seis cadenas de enteros (“INT”) donde almacenaremos las lecturas de los registros, y una variable que almacene si el botón de conexión está activado o no.

6.5.2 FUNCIÓN START

```
34      // Start is called before the first frame update
35      void Start()
36      {
37          modbusClient = new ModbusClient(IPVariador, 502); //IP y número de puerto para conectarse con el variador
38          modbusClient.UnitIdentifier = VariadorID;
39
40      }
```

Figura 6. 23 Función Start en el código final del variador

En la función Start, ya que se realiza únicamente al principio, estableceremos la configuración con la IP y el ID necesarios para realizar la conexión con el variador posteriormente.

6.5.3 FUNCIÓN UPDATE

```
// Update is called once per frame
void Update()
{
    modbusClient.Connect(); //conexión
    RegistroSETFrecuencia = modbusClient.ReadHoldingRegisters(DireccionSETFrecuencia, 1); //leemos el setpoint de la frecuencia
    RegistroVALORFrecuencia = modbusClient.ReadHoldingRegisters(DireccionVALORFrecuencia, 1); //leemos la frecuencia
    RegistroVelocidad = modbusClient.ReadHoldingRegisters(DireccionVelocidad, 2); //leemos la velocidad
    RegistroPotencia = modbusClient.ReadHoldingRegisters(DireccionPotencia, 1); //leemos la potencia
    RegistroVoltaje = modbusClient.ReadHoldingRegisters(DireccionVoltaje, 1); //leemos el voltaje
    RegistroCorriente = modbusClient.ReadHoldingRegisters(DireccionCorriente, 1); //leemos la corriente
    VelocidadInicio.text = " " + (RegistroVelocidad[1]).ToString(); //imprimimos la velocidad en la pantalla inicial

    if (boton_conectar == 1)
    {
        SETFrecuencia.text = " " + (RegistroSETFrecuencia[0] / 10).ToString(); //imprimimos el set point de la frecuencia en el panel
        VALORFrecuencia.text = " " + (RegistroVALORFrecuencia[0] / 10).ToString(); //imprimimos la frecuencia en el panel
        ValorVelocidad.text = " " + (RegistroVelocidad[1]).ToString(); //imprimimos la velocidad en el panel
        ValorPotencia.text = " " + (RegistroPotencia[0]).ToString(); //imprimimos la potencia en el panel
        ValorVoltaje.text = " " + (RegistroVoltaje[0]).ToString(); //imprimimos el voltaje en el panel
        ValorCorriente.text = " " + (RegistroCorriente[0]*10).ToString(); //imprimimos la corriente en el panel

    }
    if (boton_conectar == 0)
    {
        VALORFrecuencia.text = "0";
        SETFrecuencia.text = "0";
        ValorVelocidad.text = "0";
        ValorPotencia.text = "0";
        ValorVoltaje.text = "0";
        ValorCorriente.text = "0";
    }
    modbusClient.Disconnect(); //Desconexión
}
```

Figura 6. 24 Función Update en el código final del variador

Todas las órdenes que se encuentre dentro de la función “update”, como ya se ha explicado antes, se ejecutarán en bucle mientras el programa esté en ejecución. Primero, establecerá una conexión Modbus con el variador. A continuación, procederá a leer los registros de las variables que deseamos leer almacenándolos en las cadenas de enteros definidas anteriormente (apartado 6.5.1 “Librerías y Variables”). Después, siempre que el botón de conexión se encuentre activado (=1), imprimirá los registros que se han leído; para ello será necesario utilizar la función `ToString()` para convertir las cadenas de enteros a cadenas de texto (“String”). Por último, cerraremos la conexión Modbus, ya que, aunque se haya diseñado un botón para que el usuario se desconecte, conviene asegurar que no exista la posibilidad de que quede una conexión abierta.

6.5.4 FUNCIONES CONEXIÓN Y DESCONEXIÓN DEL VARIADOR

```
71    public void ConexionVariador()
72    {
73        boton_conectar = 1;
74    }
75    public void DesconexionVariador() //ponemos toodos los valores a cero
76    {
77        boton_conectar = 0;
78
79    }
```

Figura 6. 25 Funciones conexión y desconexión del variador

Son dos funciones que pondrán el estado del botón de conexión/desconexión, almacenado en una de las variables globales que se declararon al principio, a « 0 » o a « 1 », según lo desee el usuario.

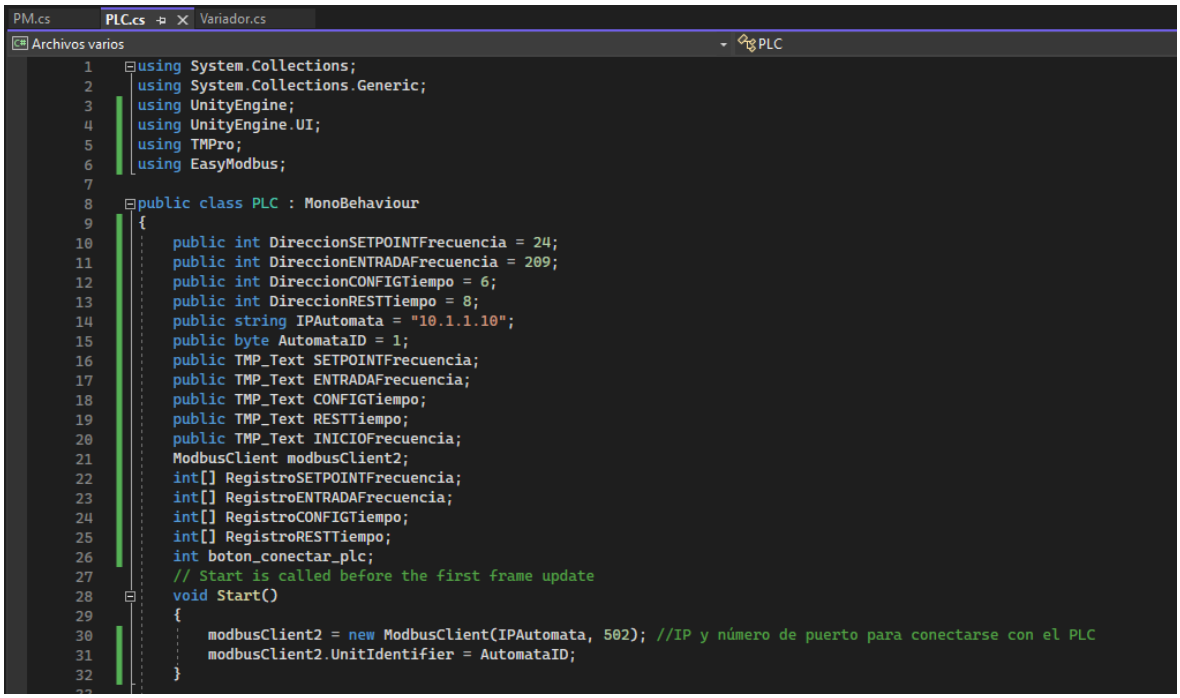
6.5.5 CÓDIGO DEL AUTÓMATA

La forma de generar un código que sea capaz de comunicarse con el autómata es idéntica a la realizada con el variador. Lo único que se debe tener en cuenta es que ahora la IP que utilizaremos será la del PLC (1.1.1.10) y que el ID será igual a 1. También es importante resaltar que los registros que queramos leer se encontrarán en sitios distintos de la memoria, ya que estamos accediendo a otro dispositivo completamente distinto. En el caso del autómata no existen mapas Modbus, ya que la memoria de éste, la gestionará el usuario a su gusto. Para poder conocer los lugares de la memoria a los que necesitamos apuntar en el código, abrimos el software UNITY PRO XL y cargaremos el programa del autómata al equipo. En el apartado de variables del árbol encontraremos una tabla dónde se especificarán todas las variables del programa y en qué lugares de la memoria están almacenadas.

Arranque_reverse	EBOOL	%Q0.2.17.0			
config_time	INT	%MW6		tiempo a configurar...	
eleccion_sentido	EBOOL	%M6	FALSE	elección del sentid...	Ninguno
emision	ARRAY[1....	%MW220			
Estado_motor	EBOOL	%M5		1- indica motor en ...	Ninguno
Fallo_conexion	EBOOL	%M10		1- indica existencia...	Ninguno
Fin_temporizador	EBOOL	%M7		1-indica que el tem...	Ninguno
frec_entrada_tcp_abs	REAL	%MW20		frecuencia de entra...	
frec_salida_analogica_negat	REAL	%MW16		frecuencia de salid...	
frec_salida_tcp_abs	REAL	%MW22		frecuencia de salid...	
frec_salida_tcp_negat	REAL	%MW18		frecuencia de salid...	
frecuencia_entrada_analogica	REAL	%MW2		frecuencia de E an...	
frecuencia_entrada_analogica_abs	REAL	%MW14		frecuencia de entra...	
frecuencia_entrada_tcp	INT	%MW209			
frecuencia_entrada_tcp_escalado	REAL	%MW10		frecuencia de entra...	
frecuencia_salida_analogica	REAL	%MW4		frecuencia de S an...	
frecuencia_salida_tcp	INT	%MW24		frecuencia de salid...	
frecuencia_salida_tcp_negat	INT				
frecuencia_salida_tcp_negativa	INT				
Parada	EBOOL	%M1		1- realizar parada p...	Ninguno
Parada_m	EBOOL	%I0.2.4.0			
Parada_manual_4	EBOOL	%M11		1- activar parada m...	Ninguno
Parada_temp	EBOOL	%M2		1- activar parada c...	Ninguno
Recepcion	ARRAY[1....	%MW208			
Reinicio_de_conexion	EBOOL	%M9		1- señal para reinci...	Ninguno
reset_modbus	EBOOL	%M16			Ninguno
segundos	INT	%MW7		segundo que lleva ...	
tiempo_restante	INT	%MW8		tiempo restante en ...	
tipo_arranque1	EBOOL	%M3		1- indicador de que...	Ninguno

Figura 6. 26 Variables del autómata

Decidí leer las variables configuración del temporizador, frecuencia de entrada y salida, y el tiempo restante cuando el temporizador está activado (opción de parada con temporizador). Los registros a los que apuntar, por tanto, son : 6, 209, 24 y 8 respectivamente. Una vez conocidos los registros, la IP y el ID podemos elaborar el código. El código de función continuará siendo el 3 « Read Holding Registers », ya que continuamos haciendo la lectura de valores enteros (tipo INT). El código final es el siguiente.




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

public class PLC : MonoBehaviour
{
    public int DireccionSETPOINTFrecuencia = 24;
    public int DireccionENTRADAFrecuencia = 209;
    public int DireccionCONFIGTiempo = 6;
    public int DireccionRESTTiempo = 8;
    public string IPAutomata = "10.1.1.10";
    public byte AutomataID = 1;
    public TMP_Text SETPOINTFrecuencia;
    public TMP_Text ENTRADAFrecuencia;
    public TMP_Text CONFIGTiempo;
    public TMP_Text RESTTiempo;
    public TMP_Text INICIOFrecuencia;
    ModbusClient modbusClient2;
    int[] RegistroSETPOINTFrecuencia;
    int[] RegistroENTRADAFrecuencia;
    int[] RegistroCONFIGTiempo;
    int[] RegistroRESTTiempo;
    int boton_conectar_plc;
    // Start is called before the first frame update
    void Start()
    {
        modbusClient2 = new ModbusClient(IPAutomata, 502); //IP y número de puerto para conectarse con el PLC
        modbusClient2.UnitIdentifier = AutomataID;
    }
```

Figura 6. 27 Código de comunicación con el PLC. Librerías, variables y función Start

```
// Update is called once per frame
void Update()
{
    modbusClient2.Connect(); //conexión
    RegistroSETPOINTFrecuencia = modbusClient2.ReadHoldingRegisters(DireccionSETPOINTFrecuencia, 1); //leemos el setpoint de la frecuencia
    RegistroENTRADAFrecuencia = modbusClient2.ReadHoldingRegisters(DireccionENTRADAFrecuencia, 1); //leemos el valor de entrada de la frecuencia
    RegistroCONFIGTiempo = modbusClient2.ReadHoldingRegisters(DireccionCONFIGTiempo, 1); //leemos la configuración del temporizador
    RegistroRESTTiempo = modbusClient2.ReadHoldingRegisters(DireccionRESTTiempo, 1); //leemos el tiempo restante del temporizador
    INICIOFrecuencia.text = " " + (RegistroENTRADAFrecuencia[0] / 10).ToString(); //imprimimos el valor de entrada de la frecuencia en el menú de inicio
    if (boton_conectar_plc == 1)
    {
        SETPOINTFrecuencia.text = " " + (RegistroSETPOINTFrecuencia[0]/10).ToString(); //imprimimos el setpoint de la frecuencia en el panel
        ENTRADAFrecuencia.text = " " + (RegistroENTRADAFrecuencia[0] / 10).ToString(); //imprimimos el valor de entrada de la frecuencia en el panel
        CONFIGTiempo.text = " " + (RegistroCONFIGTiempo[0] /1000).ToString(); //imprimimos el valor de la configuración del temporizador
        RESTTiempo.text = " " + (RegistroRESTTiempo[0]).ToString(); //imprimimos el valor del tiempo restante en el temporizador
    }
    else
    {
        SETPOINTFrecuencia.text = "0";
        ENTRADAFrecuencia.text = "0";
        CONFIGTiempo.text = "0";
        RESTTiempo.text = "0";
    }
    modbusClient2.Disconnect(); //Desconexión
}
public void ConexionPLC()
{
    boton_conectar_plc = 1;
}
public void DesconexionPLC() //ponemos toodos los valores a cero
{
    boton_conectar_plc = 0;

}
}
```

Figura 6. 28 Código de comunicación con el PLC. Funciones Update, conexión y desconexión

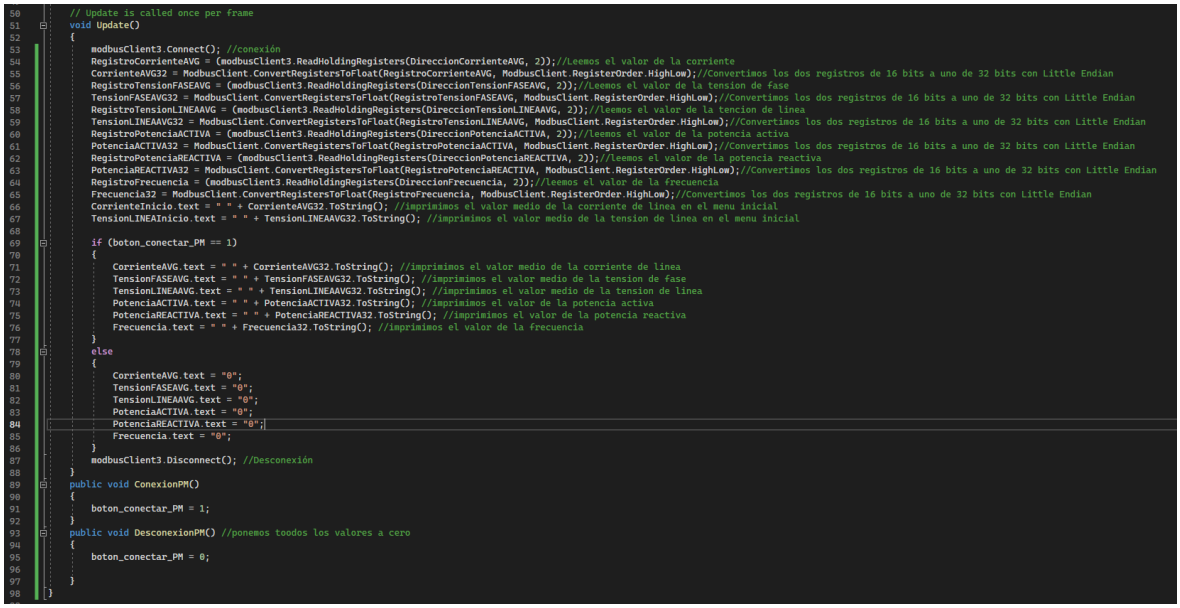
6.5.6 CÓDIGO DEL MEDIDOR DE ENERGÍA

Para el medidor de energía, la única diferencia con respecto al código del variador es que ahora los datos, en vez de ocupar 16 bits, son datos reales de 32 bits, por lo que necesitaremos solicitarlos de dos en dos registros y a continuación unirlos. Utilizando nuevamente el simulador RMSS comprobamos que la forma correcta de unir los datos es Big Endian (el bit más significativo del primer registro primero).

```
PM.cs | PLC.cs | Variador.cs
Archivos varios                                                    PM

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using EasyModbus;

public class PM : MonoBehaviour
{
    ModbusClient modbusClient3;
    public int DireccionCorrienteAVG = 3009;
    public int DireccionTensionFASEAVG = 3025;
    public int DireccionTensionLINEAAVG = 3035;
    public int DireccionPotenciaACTIVA = 3059;
    public int DireccionPotenciaREACTIVA = 3067;
    public int DireccionFrecuencia = 3109;
    int[] RegistroCorrienteAVG;
    int[] RegistroTensionFASEAVG;
    int[] RegistroTensionLINEAAVG;
    int[] RegistroPotenciaACTIVA;
    int[] RegistroPotenciaREACTIVA;
    int[] RegistroFrecuencia;
    float CorrienteAVG32;
    float TensionFASEAVG32;
    float TensionLINEAAVG32;
    float PotenciaACTIVA32;
    float PotenciaREACTIVA32;
    float Frecuencia32;
    public TMP_Text CorrienteAVG;
    public TMP_Text TensionFASEAVG;
    public TMP_Text TensionLINEAAVG;
    public TMP_Text PotenciaACTIVA;
    public TMP_Text PotenciaREACTIVA;
    public TMP_Text Frecuencia;
    public TMP_Text CorrienteInicio;
    public TMP_Text TensionLINEAInicio;
    public string IP_PM = "10.1.2.10";
    public byte PM_ID = 1;
    int boton_conectar_PM;


    // Start is called before the first frame update
    void Start()
    {
        modbusClient3 = new ModbusClient(IP_PM, 502); //IP y número de puerto para conectarse con el medidor de energía
        modbusClient3.UnitIdentifier = PM_ID;
    }

```

Figura 6. 29 Código de comunicación con el PM3255. Librería, variables y función Start.

Además de declarar el mismo tipo de variables que en el código del variador, necesitaremos declarar variables de tipo float dónde almacenaremos el valor de los dos registros ya unidos, es decir, el valor real de la variable de 32 bits. Cabe mencionar también que, en este caso, el mapa Modbus presenta un offset de -1, por lo que, si queremos leer una variable de 32 bits que está almacenada en el registro 3150, debemos apuntar al registro 3149 y solicitar dos registros.



```
// Update is called once per frame
void Update()
{
    modbusClient3.Connect(); //conexión
    RegistroCorrienteAVG = (modbusClient3.ReadHoldingRegisters(DireccionCorrienteAVG, 2));//Leemos el valor de la corriente
    CorrienteAVG32 = ModbusClient.ConvertRegistersToFloat(RegistroCorrienteAVG, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    RegistroTensionFASEAVG = (modbusClient3.ReadHoldingRegisters(DireccionTensionFASEAVG, 2));//Leemos el valor de la tension de fase
    TensionFASEAVG32 = ModbusClient.ConvertRegistersToFloat(RegistroTensionFASEAVG, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    RegistroTensionLINEAAVG = (modbusClient3.ReadHoldingRegisters(DireccionTensionLINEAAVG, 2));//leemos el valor de la tencion de linea
    TensionLINEAAVG32 = ModbusClient.ConvertRegistersToFloat(RegistroTensionLINEAAVG, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    RegistroPotenciaACTIVA = (modbusClient3.ReadHoldingRegisters(DireccionPotenciaACTIVA, 2));//leemos el valor de la potencia activa
    PotenciaACTIVA32 = ModbusClient.ConvertRegistersToFloat(RegistroPotenciaACTIVA, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    RegistroPotenciaREACTIVA = (modbusClient3.ReadHoldingRegisters(DireccionPotenciaREACTIVA, 2));//leemos el valor de la potencia reactiva
    PotenciaREACTIVA32 = ModbusClient.ConvertRegistersToFloat(RegistroPotenciaREACTIVA, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    RegistroFrecuencia = (modbusClient3.ReadHoldingRegisters(DireccionFrecuencia, 2));//Leemos el valor de la frecuencia
    Frecuencia32 = ModbusClient.ConvertRegistersToFloat(RegistroFrecuencia, ModbusClient.RegisterOrder.HighLow);//Convertimos los dos registros de 16 bits a uno de 32 bits con Little Endian
    CorrienteInicio.text = " " + CorrienteAVG32.ToString(); //imprimimos el valor medio de la corriente de linea en el menu inicial
    TensionLINEAInicio.text = " " + TensionLINEAAVG32.ToString(); //imprimimos el valor medio de la tension de linea en el menu inicial

    if (boton_conectar_PM == 1)
    {
        CorrienteAVG.text = " " + CorrienteAVG32.ToString(); //imprimimos el valor medio de la corriente de linea
        TensionFASEAVG.text = " " + TensionFASEAVG32.ToString(); //imprimimos el valor medio de la tension de fase
        TensionLINEAAVG.text = " " + TensionLINEAAVG32.ToString(); //imprimimos el valor medio de la tension de linea
        PotenciaACTIVA.text = " " + PotenciaACTIVA32.ToString(); //imprimimos el valor de la potencia activa
        PotenciaREACTIVA.text = " " + PotenciaREACTIVA32.ToString(); //imprimimos el valor de la potencia reactiva
        Frecuencia.text = " " + Frecuencia32.ToString(); //imprimimos el valor de la frecuencia
    }
    else
    {
        CorrienteAVG.text = "0";
        TensionFASEAVG.text = "0";
        TensionLINEAAVG.text = "0";
        PotenciaACTIVA.text = "0";
        PotenciaREACTIVA.text = "0";
        Frecuencia.text = "0";
    }
    modbusClient3.Disconnect(); //Desconexión
}
public void ConexionPM()
{
    boton_conectar_PM = 1;
}
public void DesconexionPM() //ponemos toodos los valores a cero
{
    boton_conectar_PM = 0;

}
}
```

Figura 6. 30 Código de comunicación con el PM3255. Funciones Update, conexión y desconexión

Para poder juntar registros, la librería Easymodbus facilita la función `ConvertRegistersToFloat` a la que pasaremos el vector con los dos registros (enteros de 16 bits) almacenados. Devolverá el valor real de la unión de los dos registros en las variables de tipo float que se declararon anteriormente. Debemos, además, especificar el orden del bit que queramos unir; en este caso el bit más significativo se debe colocar primero, por lo que el orden correcto será `HighLow`. El resto del código utilizará las mismas funciones y métodos que el primer código del variador, lo único que ahora las cadenas de texto se obtendrán convirtiendo a string las variables de tipo flotante.

6.5.7 RESUMEN DE LAS VARIABLES QUE SE ESTÁN LEYENDO Y LOS REGISTROS A LOS QUE SE ESTÁ ACCEDIENDO

Tabla 6. 1 Variables y registros pedidos en el variador

Variable	Registro	Tipo	Unidades
Setpoint Frecuencia	8502	INT 16 bits	Hertzios (Hz)
Frecuencia de entrada	3202	INT 16 bits	Hertzios (Hz)
Velocidad	8603	INT 16 bits	r.p.m
Potencia	3211	INT 16 bits	%
Voltaje	3208	INT 16 bits	Voltios (V)
Corriente	3204	INT 16 bits	Amperios (A)

Tabla 6. 2 Variables pedidas en el PLC

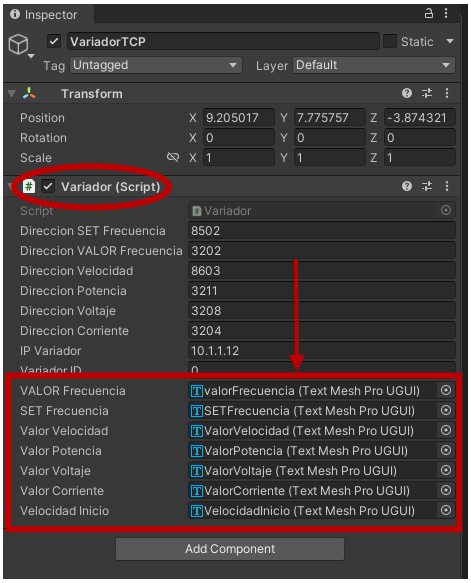
Variable	Primer byte de la memoria	Tipo	Unidades
Setpoint Frecuencia	MW24	INT 16 bits	Hertzios (Hz)
Frecuencia de entrada	MW209	INT 16 bits	Hertzios (Hz)
Config. Temporizador	MW6	INT 16 bits	Segundos (s)
Tiempo restante	MW8	INT 16 bits	Segundos (s)

Tabla 6. 3 Variables y registros pedidos en el PM3255

Variable	Registros	Tipo	Unidades
Frecuencia	3109 y 3110	REAL 32 bits	Hertzios (Hz)
Corriente	3009 y 3010	REAL 32 bits	Amperios (A)
Tensión de fase	3025 y 3026	REAL 32 bits	Voltios (V)
Tensión de línea	3035 y 3036	REAL 32 bits	Voltios (V)
Potencia activa	3059 y 3060	REAL 32 bits	Kilovatios (kW)
Potencia reactiva	3067 y 3068	REAL 32 bits	Voltiamperio reactivo (VAr)

6.6 CONFIGURACIÓN DE LOS BOTONES Y CUADROS DE TEXTO DEL PANEL

Para que Unity 3D pueda utilizar los scripts que hemos generado anteriormente, necesitamos crear un objeto vacío (clic en el botón derecho, new empty objetc) en la escena, al que vincularemos nuestro Script en C#, arrastrando el script desde Assets hasta el objeto vacío. Una vez creados los objetos y vinculados los códigos, desde el Inspector de los objetos vacíos (panel derecho) nos dejará vincular las variables de tipo « text », que hemos declarado en el fichero, con los cuadros de texto que hemos generado en el canvas; igualmente sólo necesitaremos arrastrar los cuadros de texto desde el árbol hasta los campos de las variables. De esta manera los datos y valores almacenados en estas variables se visualizarán en dichos cuadros de texto.



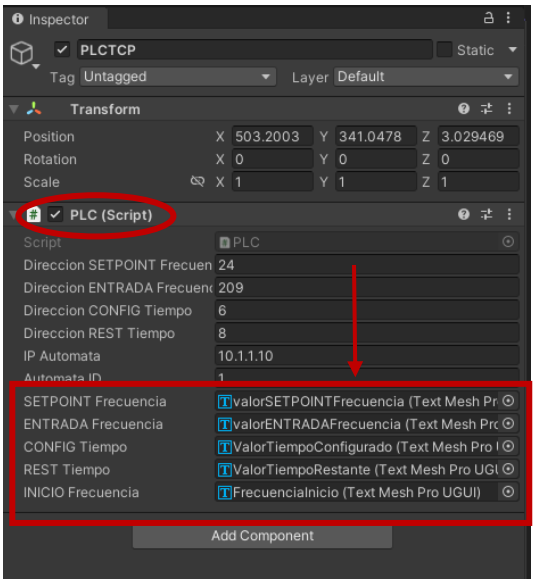


Figura 6. 31 Inspector del objeto vacío generado para el variador

Figura 6. 32 Inspector del objeto vacío generado para el autómata

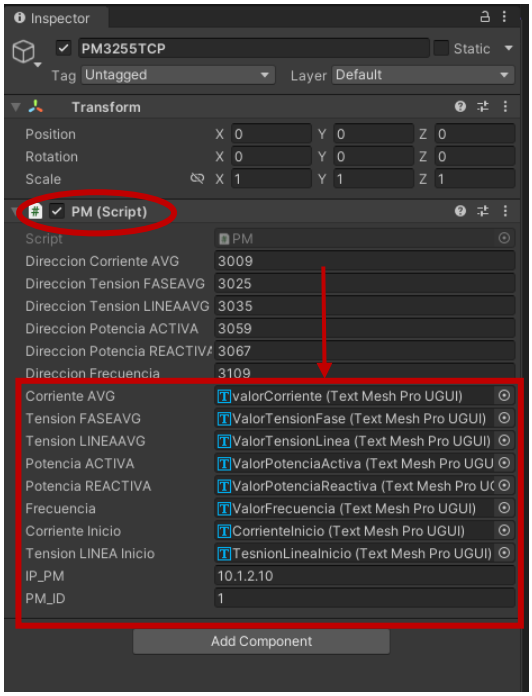
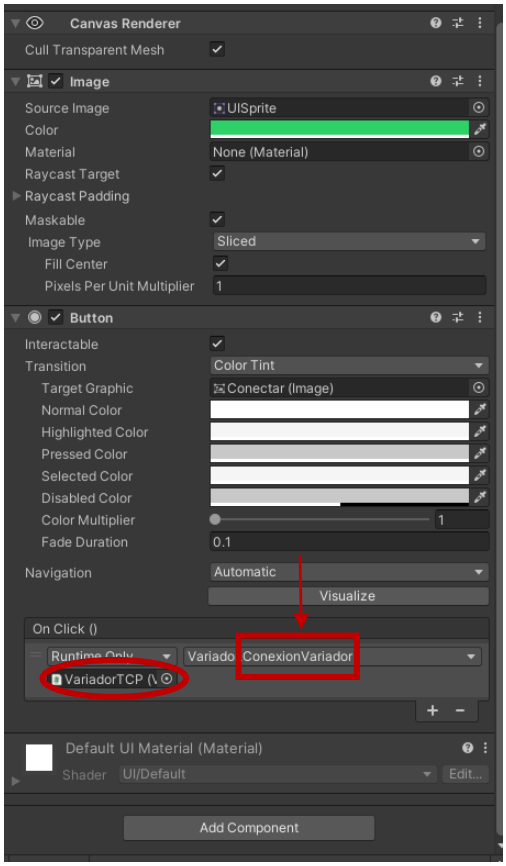


Figura 6. 33 Inspector del objeto vacío generado para el medidor de energía

Para la configuración de los botones iremos al inspector de cada uno de estos. Si arrastramos en el apartado de “On click” el objeto vacío que hemos generado, nos permitirá vincular las funciones que hemos creado en los scripts con los botones de conexión y desconexión.



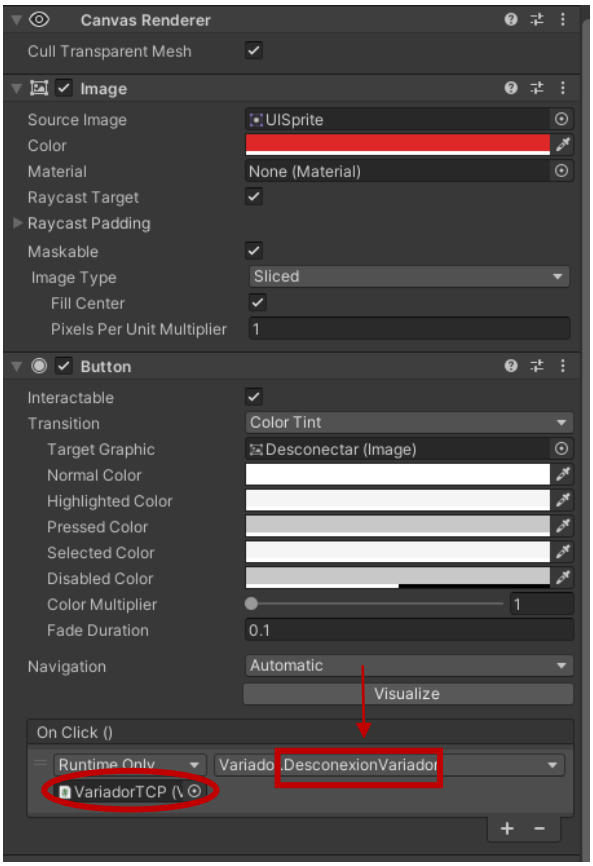
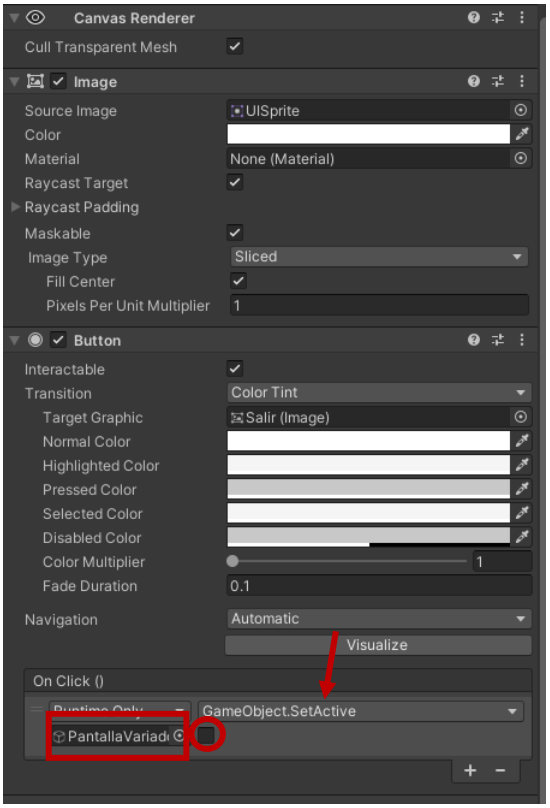


Figura 6. 34 Botón de conexión del variador

Figura 6. 35 Botón de desconexión del variador

Para el botón de salir de la pantalla jugaremos con la visibilidad de la pantalla, la cual arrastraremos, en este caso, al apartado de “On click”, y se utilizará la opción de “Game Object. Set Active”. Si dejamos marcado el recuadro, la visibilidad se activará cuando demos click al botón, por el contrario, si no se deja seleccionado la pantalla dejará de ser visible cuando se pulse el botón.

Para el botón de más detalles del menú inicial se habilitará la visibilidad de la pantalla de cada componente y se desactivará la visibilidad del propio menú.



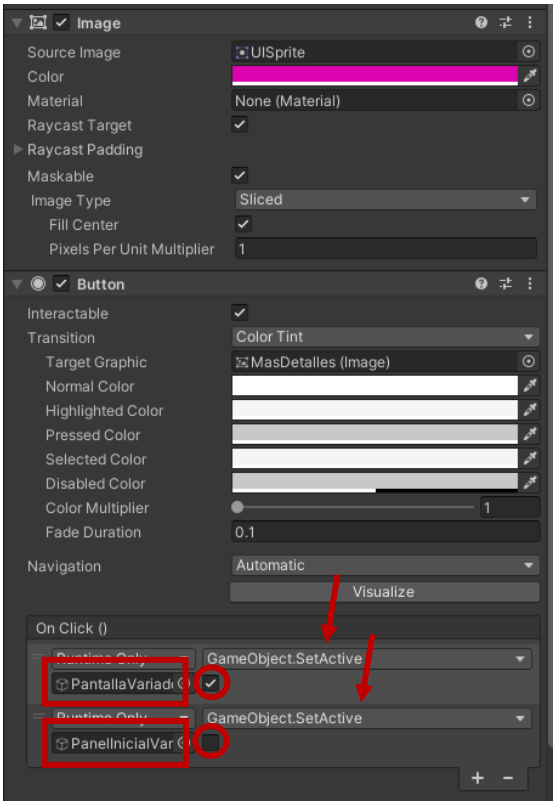


Figura 6. 36 Botón "salir" de la pantalla

Figura 6. 37 Botón más detalles del menú inicial

6.7 AÑADIR Y CONFIGURAR UN IMAGE TARGET

Para que la cámara de realidad aumentada de Vuforia pueda reconocer los tres códigos QR de cada elemento del armario de control, necesitaremos configurar tres « Image Target », los cuales podemos añadir a la escena haciendo click en el botón derecho del ratón en el árbol del proyecto. Asociaremos los códigos QR que habremos cargado previamente en la carpeta de Recursos del Asset, arrastrándolos hacia cada uno de los inspectores (panel derecho) de cada Image Target. Además, configuraremos eventos para que hagan visible el menú de cada componente cuando se muestre cada uno de los QR, y que desaparezca cuando el Image Target pertinente no sea reconocido por la cámara.

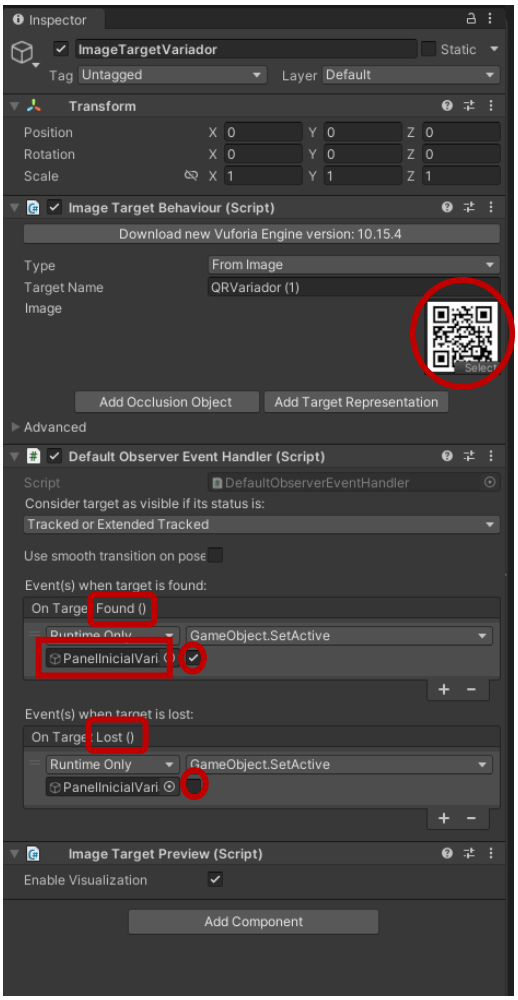


Figura 6. 38 Inspector del image target del variador

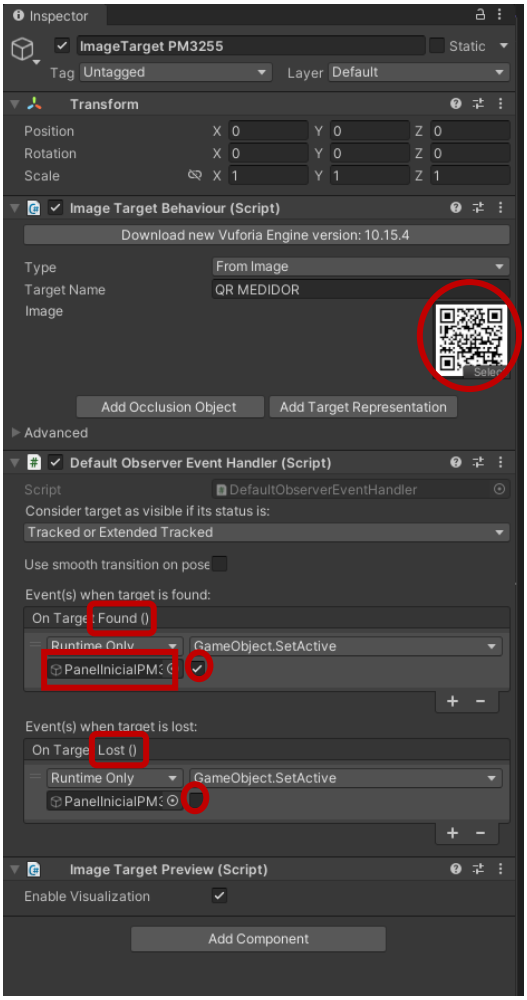


Figura 6. 39 Inspector del image target del medidor de energía

6.8 CONSTRUIR UN APK Y CARGAR LA APLICACIÓN EN LA TABLET

Para poder cargar el programa en un dispositivo android, necesitaremos cambiar de plataforma a android en los ajustes del proyecto que lo encontraremos en el panel superior izquierdo, en el apartado de file y seleccionar las escenas que deseemos cargar (en nuestro caso solo hay una). Debemos además configura los « player settings » para que la versión de Android no sea una API de nivel inferior a 26 (Android 8.0 Oreo) ya que si no resultará incompatible con la interfaz de Vuforia y no nos permitirá generar el APK. En este menú también nos permitirá elegir el ícono con el que queremos que se muestre nuestra aplicación, así como el nombre.

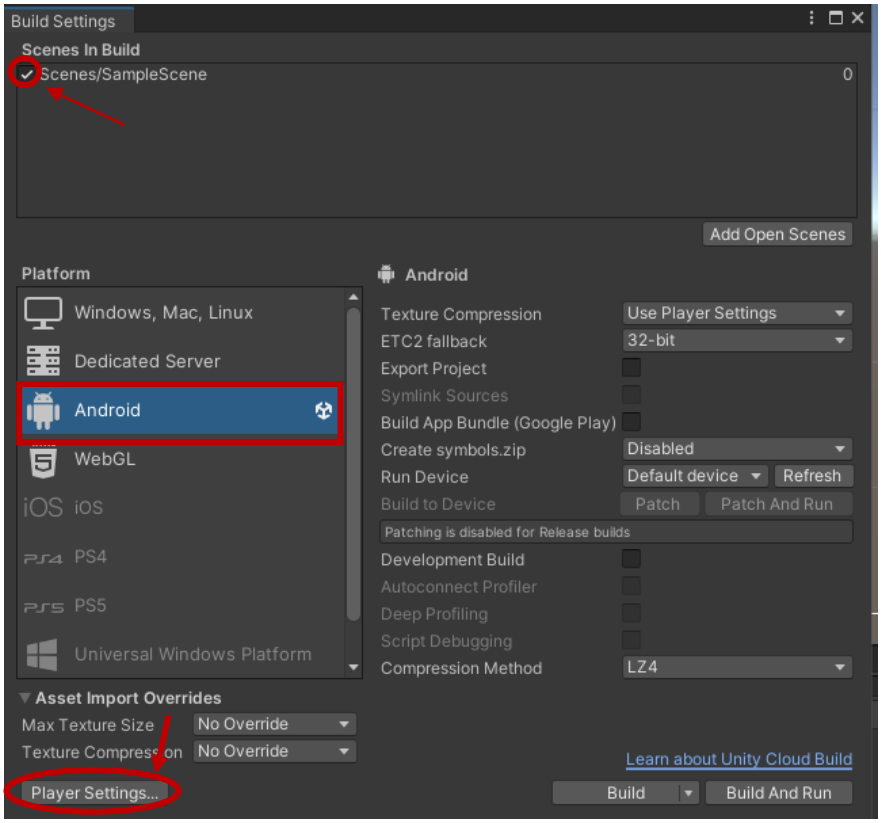


Figura 6. 40 Build Settings del proyecto

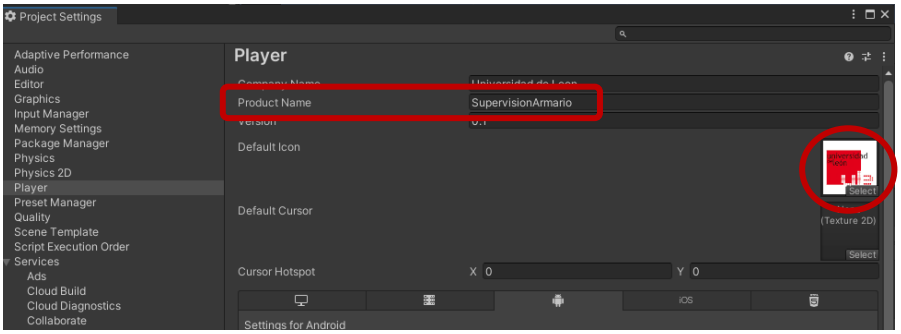


Figura 6. 41 Player Settings. Icono y nombre

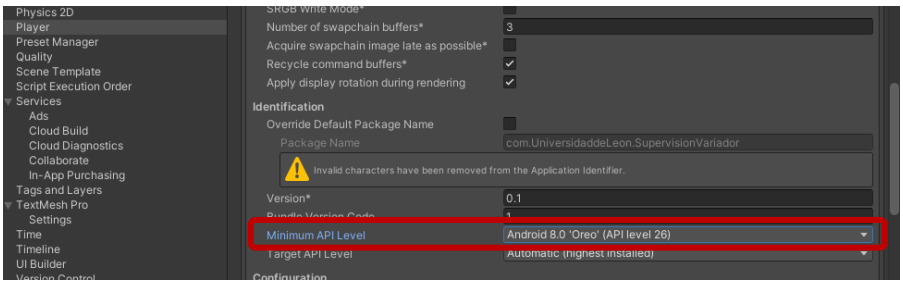


Figura 6. 42 Player Settings. Nivel de API

Una vez generado el APK solo necesitamos transferirlo a la carpeta de descargas de la tablet, conectándola a través de un cable de USB al equipo, e instalar el APK.

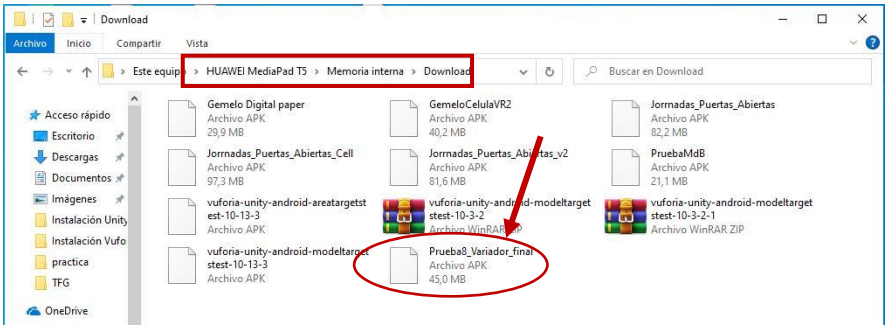


Figura 6. 43 Transferir APK a la Tablet

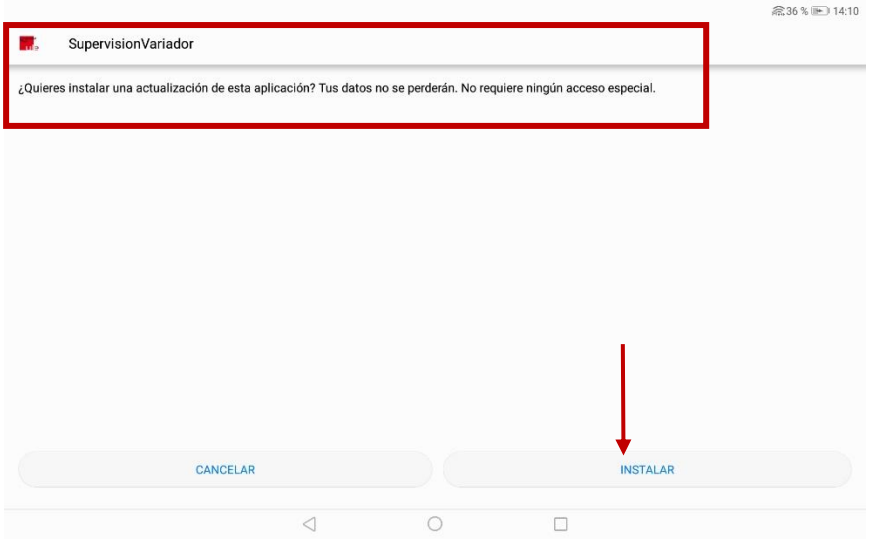


Figura 6. 44 Descargar en la Tablet

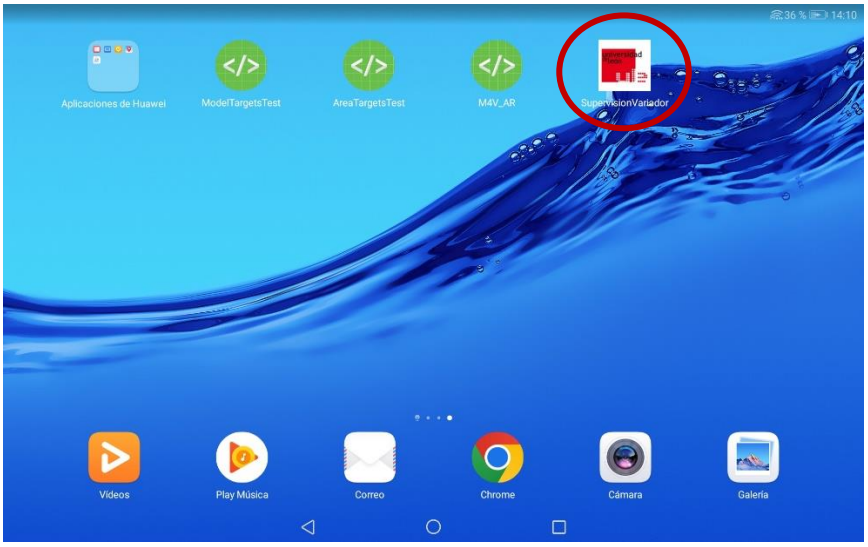


Figura 6. 45 Aplicación instalada en la tablet

6.9 RED TCP DEL LABORATORIO

Para poder conectar tanto el variador como el PLC necesitamos levantar una red para que, tanto la tablet como el armario de control, se encuentren conectados a la misma red. Para ello vamos a la configuración de red de Windows, y en « zona con cobertura inalámbrica móvil » compartimos la conexión a internet, para poder permitir a la tablet conectarse a la red.

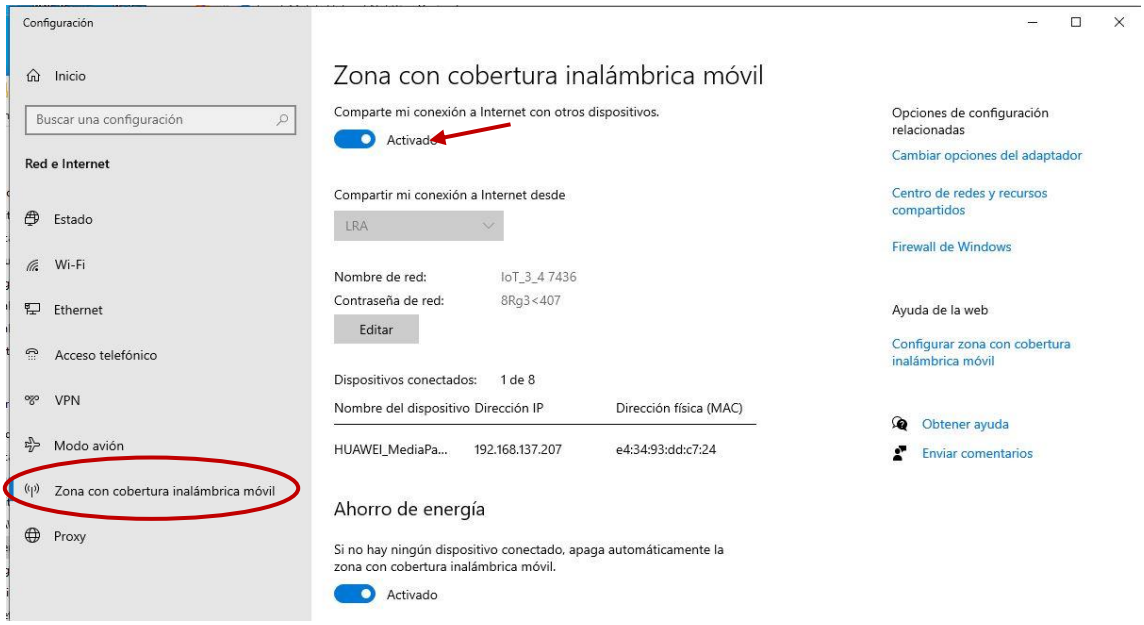


Figura 6. 46 Compartir red del laboratorio

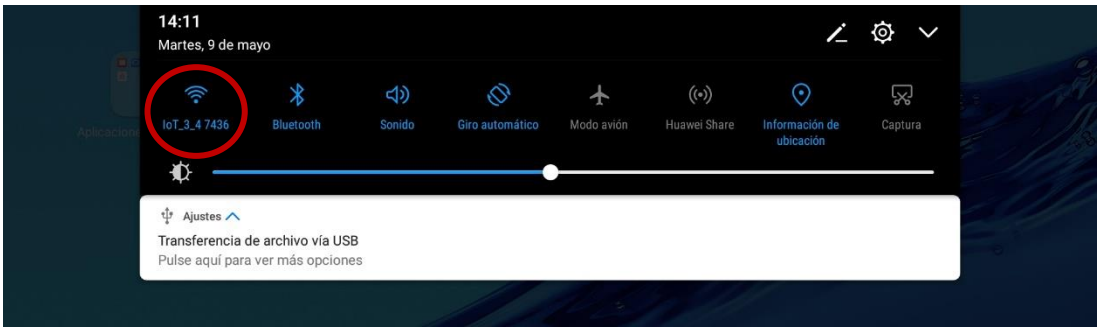


Figura 6. 47 Conectar la tablet a la red del laboratorio

7.Resultados

7.1 VÍDEO RESULTADO FINAL

El resultado final de la ejecución de la aplicación se puede observar en vídeo accediendo al siguiente enlace: https://www.youtube.com/watch?v=9Lp-fmTB42Q

7.2 SUPERVISIÓN DEL VARIADOR

Lo primero que aparece, según la cámara de RA reconoce el código QR del variador, es el menú inicial, dónde solo se imprime por pantalla el valor de la velocidad del motor en revoluciones por minuto y, también, se muestra un botón de "más detalles".

Figura 7. 1 Menú inicial del variador visto desde la pantalla de la Tablet

Si se pulsa sobre el botón de "más detalles", el usuario puede acceder a la pantalla de supervisión del variador, dónde se mostrarán más datos: setpoint y valor de entrada de la frecuencia en hertzios, velocidad del motor en revoluciones por minuto, potencia del motor en porcentaje, voltaje en voltios, y corriente en miliamperios consumidos. Además, permite dos botones de conexión, con los que se mostrarán los valores de las variables supervisadas, y el de desconexión, que permite dejar de hacer lecturas, poniendo el valor de las variables a cero. Como ya se ha mencionado anteriormente en el capítulo de desarrollo de la aplicación (6.Desarrollo de la aplicación), no resulta ni seguro ni eficiente permitir al usuario que deje una conexión abierta, por tanto, estos botones de conexión y desconexión realmente son ficticios, ya que el propio programa cierra la conexión cada vez que realiza una lectura.

Pulsando sobre el botón de salir el usuario puede abandonar la pantalla de supervisión permitiéndole detectar otros QR y otros componentes.

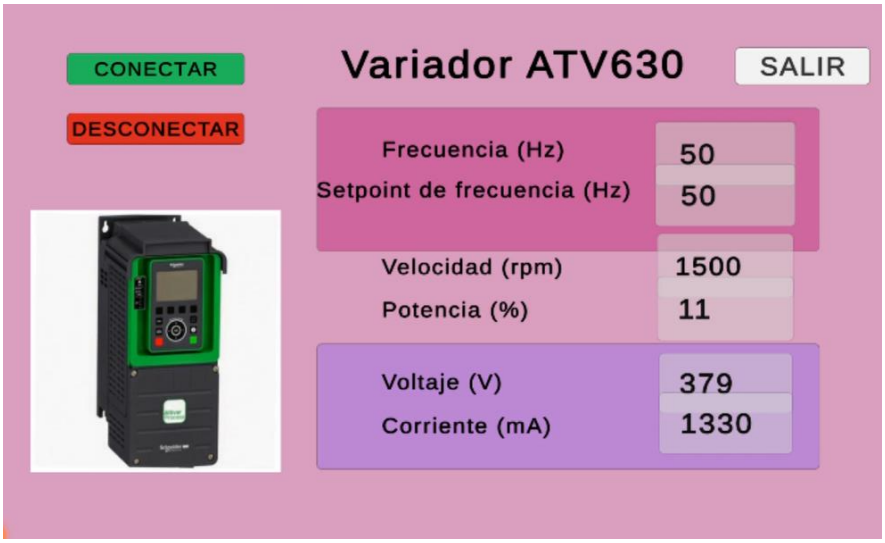


Figura 7. 2 Pantalla de supervisión del variador visto desde la Tablet

Podemos observar que las lecturas coinciden con las mostradas en el HMI del armario de control

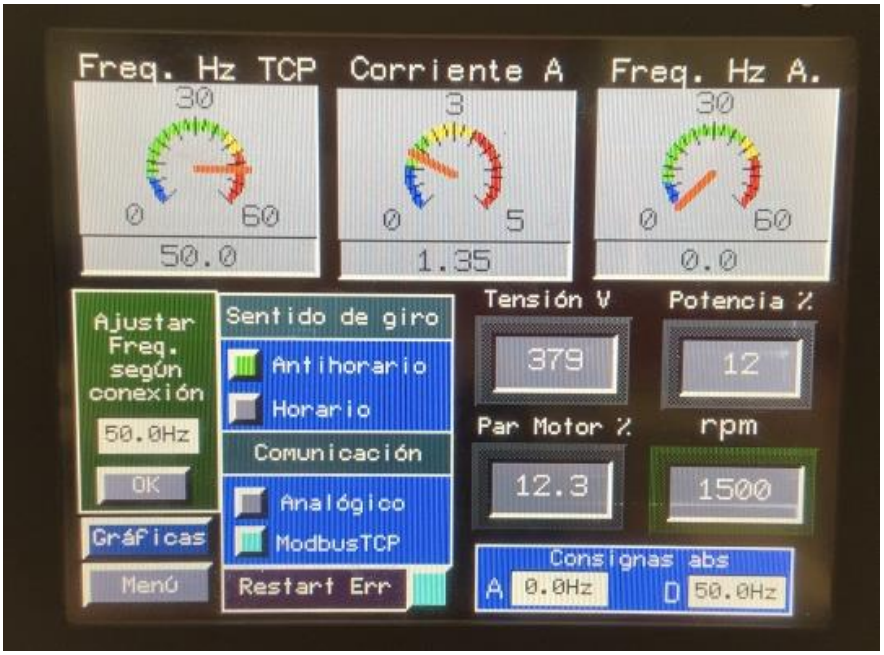


Figura 7. 3 HMI del variador visto desde la pantalla del armario de control

7.3 SUPERVISIÓN DEL PLC

La manera de proceder para la supervisión del PLC es idéntica a la seguida en el variador. Inicialmente al reconocer el QR que identifica al PLC, se mostrará el menú inicial, que en

este caso mostrará el valor de la frecuencia de entrada y permitirá, de igual manera, acceder a la pantalla de supervisión del autómata pulsando sobre el botón de más detalles.

Figura 7. 4 Menú inicial del PLC visto desde la tablet

Del mismo modo que en el variador, la pantalla de supervisión del PLC presenta botones de conexión y desconexión, y un botón de salida. En este caso se mostrarán el setpoint y valor de entrada de la frecuencia en hertzios y, también, se mostrarán los tiempos en segundos de configuración del temporizador para la opción de parada con temporización, y el tiempo restante en caso de que este modo se active.

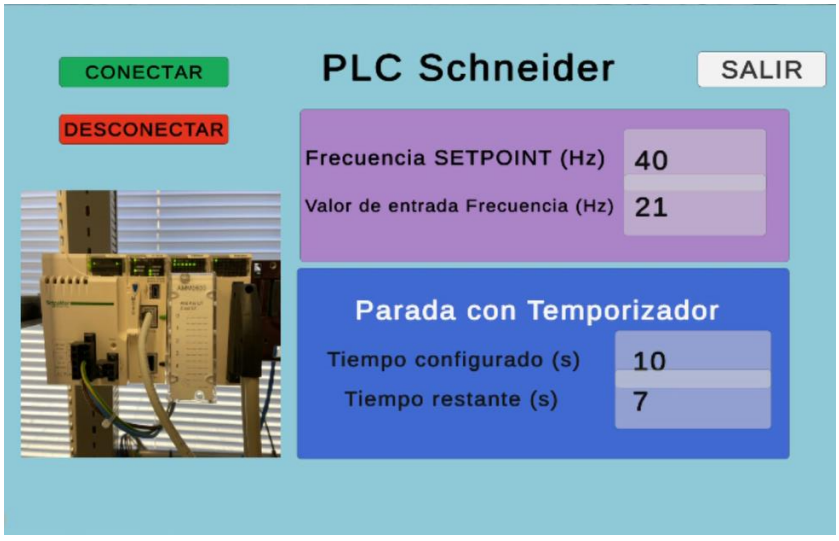


Figura 7. 5 Pantalla de supervisión del PLC visto desde la Tablet

Figura 7. 6 HMI del PLC visto desde la pantalla del armario de control

7.4 SUPERVISIÓN DEL MEDIDOR DE ENERGÍA

Para la supervisión del medidor de energía el menú inicial mostrado será idéntico al del variador y el del PLC, pero, en este caso, se imprimirán por pantalla los valores de la corriente media en amperios, y la tensión de fase media en voltios.

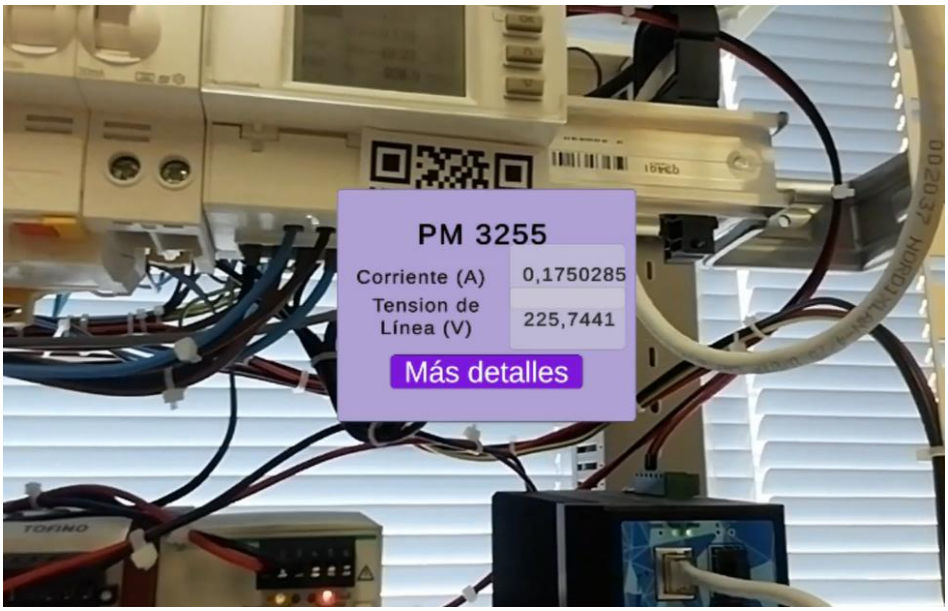


Figura 7. 7 Menú inicial de la PM3255 visto desde la pantalla de la Tablet

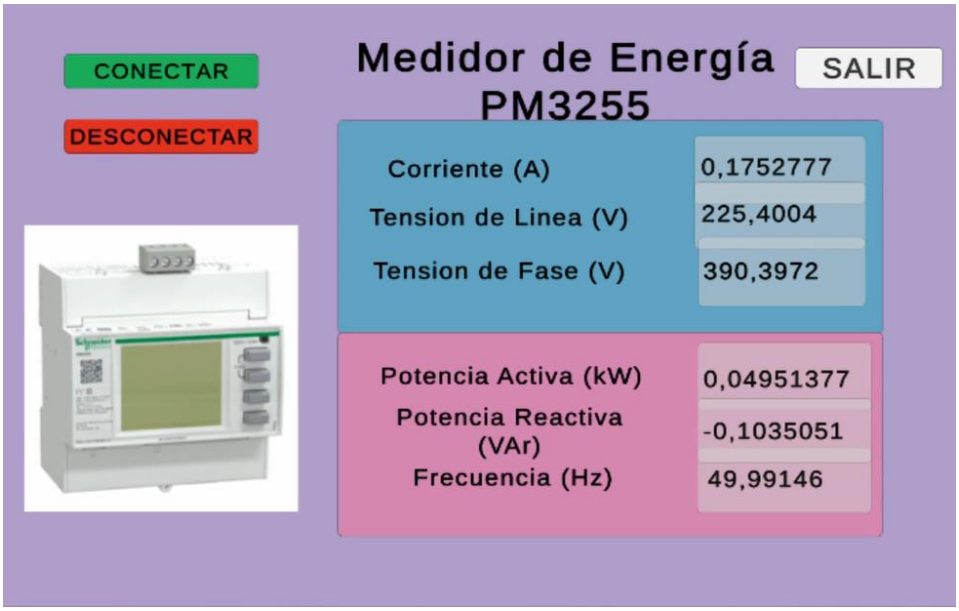


Figura 7. 8 Pantalla de supervisión del medidor de energía visto desde la Tablet

Podemos comprobar en la pantalla de la PM3255 que los valores mostrados son similares, ya que estos varían de manera constante. Se puede observar que resulta más cómodo al usuario poder supervisar los valores medidos por la PM3255 desde la pantalla de la tablet que desde la pantalla del medidor de energía que es de menor tamaño y se encuentra ubicada a cierta altura.

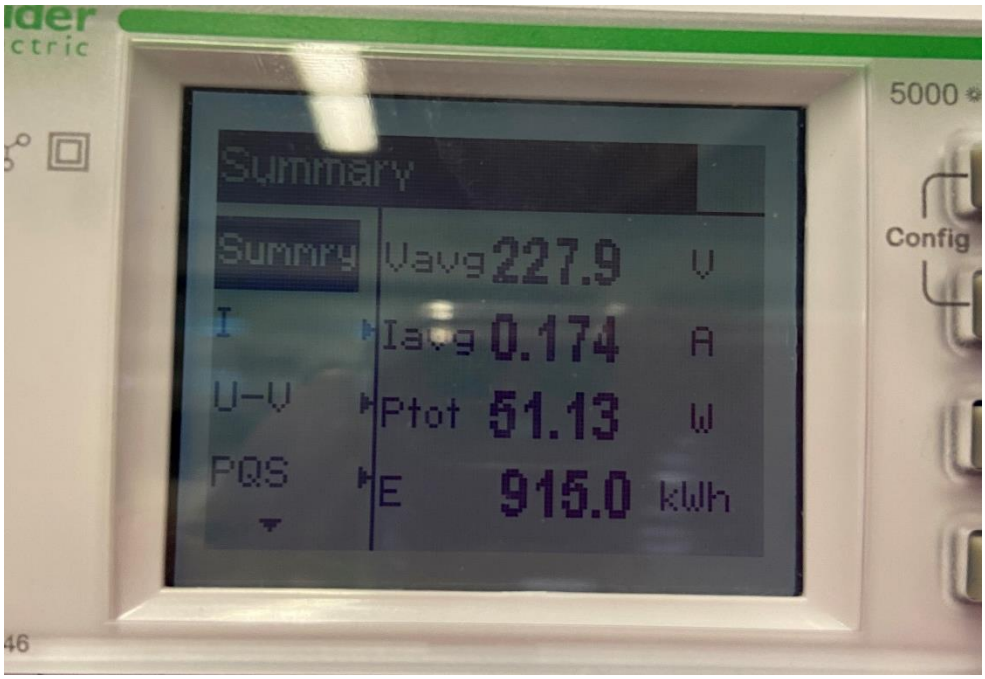


Figura 7. 9 Pantalla del medidor de energía

8.Conclusiones

Con la ejecución de este proyecto se ha demostrado el gran potencial que tiene la realidad aumentada en el ámbito industrial, facilitando y agilizando los procesos de supervisión, gestión y control. La fusión del entorno virtual con el real puede dar lugar al desarrollo de nuevos objetivos y avances que, en un principio, resultaban inalcanzables. Me gustaría resaltar que la implementación de Realidad Aumentada permite a un operario realizar las tareas de supervisión y control a nivel de campo, sin necesidad de desplazarse a un equipo donde se encuentre disponible el HMI de proceso; esto puede resultar crucial de cara a mejorar la eficacia y reducir los tiempos de reacción ante los problemas que puedan surgir en la automatización de un proceso.

En la realización del proyecto ha resultado fundamental la aplicación de los conocimientos, tanto teóricos como prácticos, adquiridos a lo largo del Grado de Ingeniería Electrónica Industrial y Automática. Ha resultado esencial conocer cómo se desarrollan las tramas de comunicación en el protocolo Modbus, y cómo y dónde se almacenan los valores de las variables en los elementos que componen un proceso de control industrial.

La tarea que ha resultado más compleja ha sido la programación de los scripts en C#, ya que es un lenguaje que no se estudia en la carrera (aunque sí se estudia el lenguaje C) y no están contempladas casi prácticas de programación en los estudios del Grado por lo que no estaba muy familiarizada con el desarrollo de códigos. Sin embargo, considero que actualmente resulta imprescindible que los ingenieros tengan cierto nivel de programación, ya que resulta muy útil y fundamental en cualquier ámbito industrial o tecnológico.

Por último, los dos aspectos que más me han llamado la atención en el desarrollo del trabajo son: la gran versatilidad que puede tener la implementación de la realidad aumentada ; y la posibilidad de utilizar una interfaz que originalmente fue pensada para el desarrollo de videojuegos, para una aplicación totalmente distinta como es la supervisión de un armario de control.

8.1 FUTURAS LÍNEAS DE TRABAJO

El proyecto da lugar a futuras líneas de trabajo entre las que se encuentran:

1. La implementación de códigos QR en cada componente, lo que supondría extender el software desarrollado a todos los elementos del armario de control, y no

únicamente al variador, al PLC y al medidor de energía. También la extensión de la aplicación para distintos armarios de control, y, por tanto, para distintas redes Modbus TCP.

2. La posibilidad de realizar escritura de datos en las variables de entrada del proceso, permitiendo de esta manera poder ajustar los valores de consigna desde la propia aplicación en la tablet. Esto daría como resultado una aplicación software que funcionaría exactamente igual que un HMI, pero sin necesidad de que el operario se tenga que desplazar hasta la pantalla del equipo donde esté instalado. De este modo, las tareas de supervisión y control se podrían realizar a nivel de campo.
3. La generación de registros históricos de las lecturas de los valores de las variables, pudiendo consultar y analizar estas bases de datos en un futuro, así como la generación de gráficos de tendencia de cada variable del proceso.
4. Actualmente la realidad aumentada ya no se extiende a tablets y teléfonos móviles, existen gafas de realidad aumentada que son capaces de generar un entorno virtual a partir del reconocimiento de elementos del entorno real. Puede resultar una línea de trabajo muy interesante la implementación de la aplicación en unas gafas de ese tipo.

9. Referencias y bibliografía

[1] Alexandru Ujvarosi. “Evolution of SCADA Systems”, Bulletin of the Transilvania University of Braşov • Vol. 9 (58) No. 1. 2016. Disponible online en: http://webbut2.unitbv.ro/BU2016/Series%20I/2016/BULETIN%20I%20PDF/Ujvarosi_Al.pdf

[2] Nagy, Judit, et al. "The role and impact of Industry 4.0 and the internet of things on the business strategy of the value chain—the case of Hungary." Sustainability 10.10 (2018).

[3] Shinde, Gitanjali Rahul, et al. "Augmented Reality and IoT." Internet of Things Integrated Augmented Reality. Springer, Singapore, 2021.pp. 55-71.

[4] Derakhshani D. Introducing Autodesk Maya. John Wiley & Sons; 2013.

[5] Blackman S. Beginning 3D game development with Unity: the world' s most widely used multi-platform game engine. New York: Apre ss; 2011.

[6] Salinas P, González ME, Quintero E, Rios H, Ramírez H, Morales S. The Development of a Didactic Prototype for the Learning of Mathematics through Augmented Reality, Procedia Computer Science 2013; 25: pp. 62-70.

[6] F. Loch, F. Quint, and I. Brishtel, ‘‘Comparing video and augmented reality assistance in manual assembly,’’ in Proc. 12th Int. Conf. Intell. Environ. (IE), Sep. 2016, pp. 147–150.

[7] Jeff Payne, “PLC vs PAC”, Control Engineering Articles. Feb. 2013. URL: https://www.idc-online.com/technical_references/pdfs/electronic_engineering/PLC_vs_pac.pdf

[8] María Gladys Rodríguez, “Sistemas de control distribuido y PLCs en aplicaciones industriales”, Universidad de La Salle, Bogotá. 1966. URL: https://ciencia.lasalle.edu.co/cgi/viewcontent.cgi?article=1346&context=ing_electrica

[9] Esteban Pérez López, “Los sistemas SCADA en la automatización industrial”, Tecnología en Marcha. Vol. 28, Nº 4, Octubre-Diciembre. Pág 3-14. Feb. 2015. URL: https://www.scielo.sa.cr/pdf/tem/v28n4/0379-3982-tem-28-04-00003.pdf

[10] Ynzunza Cortés, Carmen Berenice; Izar Landeta, Juan Manuel; Bocarando Chacón, Jacqueline

Guadalupe; Aguilar Pereyra, Felipe; Larios Osorio, Martín, “El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras”, Instituto Tecnológico de Aguascalientes, Conciencia Tecnológica, núm. 54. 2017. Disponible online en: https://www.redalyc.org/journal/944/94454631006/94454631006.pdf

[11] José Luis del Val Román, “Industria 4.0: la transformación digital de la industria”, CODII informe, Universidad de Deusto. 2021. URL: https://planeamientoeducativo.utu.edu.uy/sites/planeamientoeducativo.utu.edu.uy/files/2022-04/Industria%204.0%20la%20transformaci%2B%C2%A6n%20digital%20de%20la%20industria%20CODDII.pdf

[12] G. Li, N. Xi, H. Chen, and A. Saeed, ‘‘Augmented reality enhanced ‘top-down’ nano-manufacturing,’’ in Proc. 4th IEEE Conf. Nanotechnol., Aug. 2004, pp. 352–354.

[13] H. Regenbrecht, G. Baratoff, and W. Wilke, Augmented reality projects in the automotive and aerospace industries, Comput. Graph. Appl. IEEE 2005, vol. 25, no. 6, pp. 48–56.

[14] Ó. Blanco-Novoa, T. M. Fernández-Caramés, P. Fraga-Lamas and M. A. Vilar-Montesinos, "A Practical Evaluation of Commercial Industrial Augmented Reality Systems in an Industry 4.0 Shipyard," in *IEEE Access*, vol. 6, pp. 8201-8218, 2018, doi: 10.1109/ACCESS.2018.2802699.

[15] M. Olbrich, H. Wuest, P. Riess, and U. Bockholt, ‘‘Augmented reality pipe layout planning in the shipbuilding industry,’’ in Proc. 10th IEEE Int. Symp. Mixed Augmented Reality, Oct. 2011, pp. 269–270.

[16] Chaves-Diéguez, David, et al. "Providing IoT services in smart cities through dynamic augmented reality markers." Sensors 15.7 2015, pp. 16083-16104.

[17] Lobo, Nestor. "Intelli-mirror: An augmented reality based IoT system for clothing and accessory display." 2016 International Conference on Internet of Things and Applications (IOTA). IEEE, 2016.

[18] M. Antonijević, S. Sučić and H. Keserica, "Augmented reality for substation automation by utilizing IEC 61850 communication," *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 2016, pp. 316-320, doi: 10.1109/MIPRO.2016.7522159.

[19] Pokrić, Boris, Srdan Krco, and Maja Pokrić. "Augmented reality based smart city services using secure iot infrastructure." 2014 28th international conference on advanced information networking and applications workshops. IEEE, 2014.

[20] S. Sureshkumar, C. P. Agash, S. Ramya, R. Kaviyaraj and S. Elanchezhiyan, "Augmented Reality with Internet of Things," *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, Coimbatore, India, 2021, pp. 1426-1430, doi: 10.1109/ICAIS50930.2021.9395941.

[21] R. B. James Edwards, Networking Self-Teaching Guide OSI, TCP/IP, LANs, MANs, WANs, Implementation, Management, and Maintenance. Indianapolis, Indiana, United States of America: Wiley Publishing, Inc., first ed., 2009.

[22] Jonnathan Jesús Rosero Topón, William Armando Mendoza Yaguachi, "Comunicacion industrial sobre ´ sistemas de proteccion con IEDs en subestaciones de distribucion", Universidad Politécnica Salesiana sede Quito. Jul. 2013. URL: https://dspace.ups.edu.ec/handle/123456789/5164

[23] Sergio Montero, Javier Gozalvez, Miguel Sepulcre, Gonzalo Prieto, "Efecto de la Movilidad en Redes Inalámbricas de Comunicaciones Industriales", Uwicore, Universidad Miguel Hernández de Elche. 2012. URL: https://uwicore.umh.es/files/paper/2012_national/uwicore_URSI2012_Efecto%20de%20la%20Movilidad%20en%20Redes%20Inal%C3%A1mbricas%20de%20Comunicaciones%20Industriales.pdf

[24] Marlon José Matosa Pérez, Ricardo Antonio Martínez Bayuelo, "Implementación de Modbus sobre TCP/IP utilizando labview", 2008 URL: https://hdl.handle.net/20.500.12585/1923

[25] George Thomas, "Introduction to Modbus Serial and Modbus TCP" CCControls, septiembre-octubre 2008, URL: https://icscsi.org/library/Documents/ICS_Protocols/Extension%20-%20Introduction%20to%20Modbus.pdf

[26] Crystal Bedell, "Definition: Big Endian and Litlle Endian", TechTarget, URL: https://www.techtarget.com/searchnetworking/definition/big-endian-and-little-

endian#:~:text=Big%2Dendian%20is%20an%20order,the%20sequence)%20is%20stored%20first.

[27] Uy Nguyen, “Big Endian vs Little Endian”, Github, 2018 URL: https://uynguyen.github.io/2018/04/30/Big-Endian-vs-Little-Endian/

[28] Scheneider Electric, “Hoja del producto: Altivar Process ATV600”, 3 Mayo 2023, Disponible online en: https://www.se.com/es/es/product/ATV630D22N4/altivar-process-atv600-variador-de-velocidad-atv630-22-kw-30-hp-380-480-v-ip21-ul-tipo-1/

[29] Scheneider Electric, “Hoja de característica del producto: BMXCPS2000”, 1 Junio 2023, Disponible online en: file:///C:/Users/alici/Downloads/M%C3%B3dulos%20Modicon%20x80_BMXCPS2000.pdf

[30] Scheneider Electric, “Hoja de característica del producto: BMXP342020”, 1 Junio 2023, Disponible online en: file:///C:/Users/alici/Downloads/Modicon%20M340_BMXP342020.pdf

[31] Scheneider Electric, “Hoja de característica del producto: BMXAMM0600”, 5 Junio 2023, Disponible online en: https://www.se.com/es/es/product/BMXAMM0600/m34004-ea-02-sa-12baisl10v20ma/

[32] Scheneider Electric, “Hoja de característica del producto: BMXDDM3202K”, 5 Junio 2023, Disponible online en: https://www.se.com/es/es/product/BMXDDM3202K/m34016-ed-24vdc-16-sd-rele2a-40p/

[33] Scheneider Electric, “Hoja de característica del producto: METSEPM3255”, 2 Junio 2023, Disponible online en: https://www.se.com/es/es/product/METSEPM3255/pm3255-central-de-medida-2-i-digitales-2-o-digitales-rs485/

[34] J. Xie, "Research on key technologies base Unity3D game engine", 2012 7th International Conference on Computer Science & Education (ICCSE), Melbourne, VIC, Australia, 2012, pp. 695-699, doi: 10.1109/ICCSE.2012.6295169.

[35] Gu Jiawei, Zhang Yanpeng, Cao Dayuan, et al. Desigb and Simulation of virtual experiment system based on Unity 3D and AR Technology[J]. 2014.

[36] Luo Dongyong, Zhang Shujun. An Automatic Navigation Method of Mobile AR Based on Unity 3D [J]. Computer and Digital Engineering 2015(11):2024-2028

[37] Schneider Electric "Manual Unity Pro" 23 Julio 2008. Disponible online en: http://automata.cps.unizar.es/webcursoaut/ManualformacionUnityPro.pdf

[38] Fernando Gómez-Gonzalvo, Mónica Mir Daud "Revisión bibliográfica sobre usos pedagógicos de los códigos QR". TIC Revista de Innovación educativa, Universidad de Valencia. Julio-diciembre 2015. URL: https://www.redalyc.org/pdf/3495/349543461004.pdf

[39] Jessica Gladys Contreras Orellana, "Tecnología QR en el proceso de control de inventario del programa de vaso de leche de la municipalidad de Quilmaná – Cañete". Universidad de Lima. 23 sept 2017. URL: https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/16296/Contreras_OJG.pdf?sequence=1&isAllowed=y