ISBN: 978-99983-69-23-8 (Impreso)
ISBN: 978-99983-69-35-1 (E-Book, pdf)

INFORME FINAL DE INVESTIGACIÓN

LABORATORIO DE EXPERIMENTACIÓN DE CONTROL DE PROCESOS DE FLUJO Y NIVEL DE FLUIDOS FPC, QUE INTEGRA UN SIMULADOR VIRTUAL CON REALIDAD MIXTA Y CONTROL A DISTANCIA EN TIEMPO REAL APLICANDO TELEINGENIERÍA

APLICACIÓN EN ITCA-FEPADE SEDE CENTRAL

DOCENTES INVESTIGADORES PRINCIPALES
TEC. JUAN JOSÉ GUEVARA VÁSQUEZ
LIC. LUIS ERNESTO ELÍAS MORALES

DOCENTES COINVESTIGADORES
TEC. CARLOS GEOVANY MELÉNDEZ MOLINA
INGA. RAISA FABIOLA RAMÍREZ REYES

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA Y ESCUELA DE INGENIERÍA DE COMPUTACIÓN, ITCA-FEPADE SEDE CENTRAL

ENERO 2024

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

ITCA EDITORES

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

ISBN: 978-99983-69-23-8 (Impreso)
ISBN: 978-99983-69-35-1 (E-Book, pdf)

INFORME FINAL DE INVESTIGACIÓN

LABORATORIO DE EXPERIMENTACIÓN DE CONTROL DE PROCESOS DE FLUJO Y NIVEL DE FLUIDOS FPC, QUE INTEGRA UN SIMULADOR VIRTUAL CON REALIDAD MIXTA Y CONTROL A DISTANCIA EN TIEMPO REAL APLICANDO TELEINGENIERÍA

APLICACIÓN EN ITCA-FEPADE SEDE CENTRAL

DOCENTES INVESTIGADORES PRINCIPALES
TEC. JUAN JOSÉ GUEVARA VÁSQUEZ
LIC. LUIS ERNESTO ELÍAS MORALES

DOCENTES COINVESTIGADORES
TEC. CARLOS GEOVANY MELÉNDEZ MOLINA
INGA. RAISA FABIOLA RAMÍREZ REYES

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA Y ESCUELA DE INGENIERÍA DE COMPUTACIÓN, ITCA-FEPADE SEDE CENTRAL

ENERO 2024

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA
ITCA EDITORES
ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA
CdA Comisión de Acreditación de la Calidad Académica INSTITUCION ACREDITADA

003.3
L123 Laboratorio de experimentación de control de procesos de flujo y nivel de fluidos FPC, que integra un simulador virtual con
slv realidad mixta y control a distancia en tiempo real aplicando teleingeniería, aplicación en ITCA-FEPADE Sede Central / Juan José Guevara Vásquez, Luis Ernesto Elías Morales, Carlos Geovany Meléndez Molina Raisa Fabiola Ramírez Reyes.— 1ª ed. -- Santa Tecla, La Libertad, El Salv. : ITCA Editores, 2024.

1 recurso electrónico: (57 p. : il. ; 28 cm.)
Datos electrónicos (1 archivo : pdf, 4 MB). - https://www.itca.edu.sv/producción-académica/
ISBN: 978-99983-69-35-1 (E-Book, pdf)
ISBN: 978-99983-69-23-8 (Impreso)

1. Simulación por computadores digitales - Equipo.
2. Control de procesos industriales - Automatización.
3. Sistemas de control digital. I. Guevara Vásquez, Juan José, 1978- II, coaut. Título.

Autores
Tec. Juan José Guevara Vásquez
Lic. Luis Ernesto Elías Morales

Coautores
Tec. Carlos Geovany Meléndez Molina
Inga. Raisa Fabiola Ramírez Reyes

Tiraje: 13 ejemplares
Año 2024

Este documento técnico es una publicación de la Escuela Especializada en Ingeniería ITCA–FEPADE; tiene el propósito de difundir la Ciencia, la Tecnología y la Innovación CTI, entre la comunidad académica, el sector empresarial y la sociedad, como un aporte al desarrollo del país. Para referirse al contenido debe citar el nombre del autor y el título del documento. El contenido de este Informe es responsabilidad de los autores.

Escuela Especializada en Ingeniería ITCA-FEPADE
Km 11.5 carretera a Santa Tecla, La Libertad, El Salvador, Centro América
Sitio Web: www.itca.edu.sv
TEL: (503)2132-7423

CONTENIDO

1. INTRODUCCIÓN

En este informe se describe el proceso que se llevó a cabo y los resultados obtenidos durante el desarrollo del proyecto de investigación *Laboratorio de Experimentación de Control de Procesos de Flujo y Nivel de Fluidos FPC, que integra un Simulador Virtual con Realidad Mixta y Control a Distancia en Tiempo real aplicando Teleingeniería.* El objetivo del proyecto fue mejorar la experiencia de aprendizaje y experimentación en el campo del control de Procesos Industriales de Flujo y Nivel FPC y control automático.

La investigación fue desarrollada por un equipo multidisciplinario de investigadores de la Escuela de Ingeniería Eléctrica y Electrónica y de la Escuela de Ingeniería en Computación de ITCA-FEPADE Sede Central. Ambos equipos desarrollaron el trabajo en forma paralela en la ejecución de las actividades definidas en una matriz metodológica para alcanzar los resultados. La metodología de investigación se abordó desde la perspectiva de programación de ecuaciones matemáticas, control electrónico, comunicación industrial, la programación de aplicaciones web, así como diseño y modelado 3D.

Los docentes investigadores de la Escuela de Ingeniería Eléctrica y Electrónica fueron responsables del diseño y montaje de un controlador electrónico que permite el control a distancia de un entrenador FPC de flujo y nivel. Se diseñó el firmware del controlador electrónico basándose en un RTOS (Sistema Operativo en Tiempo Real) que se ejecutará en los procesadores que integran el circuito. Se diseñó una interfaz gráfica en una nube VNET que permite monitorear y controlar el proceso de flujo y nivel de forma remota por medio de un navegador Web. Se integró el control electrónico del entrenador FPC con la plataforma de Teleingeniería y simulador virtual con realidad aumentada.

Los docentes investigadores de la Escuela de Ingeniería en Computación tuvieron a su cargo la programación de los módulos de entrenamiento y simulación con componentes 3D y realidad aumentada. Se programó el Dashboard de análisis de datos con Business Intelligence y se realizaron pruebas integradas de la aplicación de Teleingeniería con el dispositivo físico.

Se desarrolló un modelo matemático del proceso industrial de flujo y nivel y, sobre la base de ese modelo, se creó un programa computacional para que, a partir de datos de entradas típicas de un control Proporcional Integral Derivativo PID, sea capaz de generar comportamientos similares a los que se producen en el proceso real. Se comprobaron estos resultados con pruebas tanto en Matlab como de forma experimental, obteniendo resultados satisfactorios. Posteriormente se integró el programa a los objetos tridimensionales que en su conjunto conforman el simulador. Paralelamente, se creó un control electrónico de Internet Industrial de las Cosas IIoT y se programó de tal manera que permite el control del proceso industrial a distancia a través de un Dashboard que se ejecuta en la nube. Como resultado se obtuvo un simulador virtual con realidad mixta que se comporta como un gemelo digital de un entrenador de procesos industriales, y un entrenador físico del proceso que se controla de forma remota por medio de un Dashboard alojado en la plataforma IIoT VNET.

Se incorporó soporte para tecnologías de realidad mixta, lo que implica la integración de elementos virtuales en el entorno real del usuario. Esto se logró combinando información del mundo físico con objetos y datos generados digitalmente. Asimismo, se incluye la funcionalidad de realidad aumentada, que enriquece la experiencia del usuario al superponer información adicional y elementos virtuales en el mundo real a través de la cámara del dispositivo que se utilice.

Se elaboró un “Manual de Prácticas” de diferentes niveles para utilizar el simulador y el entrenador en prácticas remotas programadas para estudiantes de Técnico en Ingeniería Eléctrica de Sede Central, Santa Ana y San Miguel, a través de la Plataforma de Teleingeniería implementada. El laboratorio también cuenta con un simulador de la planta que se puede instalar y ejecutar en un dispositivo móvil. Además, se elaboró un “Manual de Usuario” del simulador virtual con realidad aumentada.

2. Planteamiento del Problema

2.1. Definición del Problema

En las instituciones educativas, sucede con bastante regularidad que los equipos especializados para prácticas de laboratorio son escasos, ya que son de alto costo y mantenimiento. Esto tiene como inconveniente que la cantidad de horas efectivas que los estudiantes pueden pasar frente a estos equipos es muy reducida, dificultando su proceso de aprendizaje. Al planteamiento anterior es necesario agregar que debido a la pandemia por COVID-19 se implementaron medidas de distanciamiento físico entre estudiantes y docentes limitando aún más el acceso físico a los equipos de laboratorio.

El empleo de asistentes virtuales, así como la educación virtual en general, es muy demandado y al mismo tiempo exigido como una herramienta de apoyo para el aprendizaje. La educación virtual ha tenido que afrontar muchos retos, desde la actualización de competencias de IT por parte de docentes y alumnos, así como la disminución de la brecha digital. Otro factor que ha afrontado la educación a nivel mundial es la competencia que existe con la cantidad de material de ocio que el alumno encuentra en la red, desviándolo del objetivo principal por el que hace uso de equipo informático o móvil al momento de sus clases.

Actualmente, muchas instituciones educativas no cuentan con herramientas tecnológicas que ayuden al aprendizaje de los alumnos de una manera inmersiva, en la que el uso de simuladores se vuelve un lujo por el elevado costo que estos presentan, teniendo que limitarse únicamente con aulas virtuales, videos, documentos en PDF u otro formato multimedia.

Este proyecto buscó proveer una solución que permite optimizar el tiempo de acceso a los equipos para las prácticas de control de procesos industriales, registrar las actividades que los estudiantes lleven a cabo, generar métricas que permiten al docente identificar conductas y aprendizajes para tomar acciones pedagógicas con las que se puedan alcanzar las competencias de una manera más efectiva.

2.2. Antecedentes / Estado de la Técnica

En el área de los laboratorios de control de procesos y simulación, ITCA-FEPADE ha venido investigando y documentando, desde el año 2020, el diseño de una plataforma de Teleingeniería para el control de procesos industriales, concretamente para el proceso de control de temperatura de un fluido (H_2O). Mientras que, en el año 2022 se estudió la estructura y funcionamiento de una planta de control de flujo y nivel para crear los modelos matemáticos que rigen su funcionamiento, se analizaron y se pusieron a prueba los dispositivos de control de una planta (entrenador) de control de procesos y se crearon modelos en 3D de los dispositivos de la planta. Estos procesos de investigación se han desarrollado en etapas similares a las que se establecen en la descripción de este proyecto, por lo que el equipo de investigadores de la Escuela de Ingeniería Eléctrica y Electrónica, así como de la Escuela de Ingeniería en

Computación de la Sede Central, cuentan con la experiencia base que les permite avanzar e incorporar a la plataforma existente el control de un nuevo proceso de flujo y nivel.

Existen diversos tipos de dispositivos que proporcionan las capacidades necesarias en función del tipo de solución a implementar. Por tanto, respondiendo a la demanda creciente de dispositivos para aplicaciones de realidad aumentada, el mercado cuenta con una oferta cada vez más variada. Hoy en día, la empresa Microsoft con su dispositivo de realidad virtual llamado HoloLens es un referente con lo que a realidad mixta concierne. Este va por su segunda versión y es utilizado para las áreas de fabricación, atención sanitaria y educación. Su precio en el mercado varía de acuerdo con la función que se oriente. Se cuenta con una variedad de aplicaciones de paga que se pueden instalar en el dispositivo proporcionado por Microsoft Dynamics 365, las cuales son una línea de productos de aplicaciones empresariales inteligentes de planificación de recursos empresariales y gestión de relaciones con los clientes.

Fig. 1. Diferentes formas de aplicación con la HoloLens 2 de Microsoft en la industria y educación.

Actualmente, en el mercado existe una amplia variedad de dispositivos para realidad aumentada, cuyo funcionamiento es similares al de HoloLens, pero con diferente calidad y rendimiento, lo que los vuelve, en algunos casos, más económicos o de mayor costo. Ejemplo de ellos son: Oculus Quest 2, Magic Leap, Daqri Smart Helmet.

Oculus Quest 2 Magic Leap Daqri Smart Helmet

Fig. 2. Ejemplos de dispositivos de realidad aumentada.

Para nuestro proyecto se usó un Smartphone. El modelo por emplear debe ser lo más genérico posible, ya que dentro de la población estudiantil existe una amplia gama de marcas y modelos con las que el simulador debe ser compatible. Por lo tanto, el smartphone a emplear debe tener estas características:

- Pantalla de 5.8" a 6"
- Poseer giroscopio
- Sistema Operativo Android 8.0 como mínimo

Junto con el hardware descrito anteriormente, se requerirá de una aplicación que permita cargar la plataforma con todas las simulaciones de las prácticas del módulo a impartir. Una vez iniciada la aplicación, se elige la práctica y posteriormente el dispositivo móvil es ubicado de tal manera que pueda leer una marca impresa (código QR u otra personalizada) para iniciar con el desarrollo de la simulación.

Fig. 3. Usuario utilizando una simulación de realidad aumentada en el mantenimiento de maquinaria.

2.3. JUSTIFICACIÓN

Debido al problema de pandemia que se vive en el mundo, las aulas y talleres se han visto trasladados hacia los hogares de la población estudiantil, teniendo que vérselas con recursos propios de los alumnos, los cuales no siempre cuentan con ellos y en especial cuando se trata de equipo especializado. Esto repercute en el aumento de la deserción estudiantil por desmotivaciones respecto a la educación que perciben que se les imparte.

En tal situación, se requiere de un mecanismo que ayude a la población estudiantil a realizar sus prácticas de una manera inmersiva de forma virtual con el uso de tecnología y aprovechar al máximo la realidad aumentada. Adicionalmente con la implementación de las tecnologías descritas es posible retroalimentar a los docentes para que puedan efectuar acciones que minimicen los errores y problemas de los estudiantes incrementando significativamente la calidad de la enseñanza en la institución.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Diseñar un Laboratorio para experimentación y simulación virtual de Control de Procesos de Flujo y Nivel de Fluidos FPC, integrando realidad mixta y control a distancia en tiempo real en una plataforma de Teleingeniería.

3.2. OBJETIVOS ESPECÍFICOS

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

a) Diseñar y realizar el montaje de un controlador electrónico que permita el control a distancia de un entrenador FPC de flujo y nivel.

b) Diseñar el firmware del controlador electrónico basándose en un RTOS (Sistema Operativo en Tiempo Real) que se ejecutará en los procesadores que integran el circuito.

c) Diseñar una interfaz gráfica en una nube VNET que permita monitorear y controlar el proceso de flujo y nivel de forma remota por medio de un navegador Web.

d) Integrar el control electrónico del entrenador FPC con la plataforma de Teleingeniería y simulador virtual con realidad aumentada.

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

a) Programar módulos de entrenamiento y simulación con componentes 3D y realidad aumentada.

b) Programar el Dashboard de análisis de datos con Business Intelligence.

c) Realizar pruebas integradas de la aplicación de Teleingeniería con el dispositivo físico.

4. HIPÓTESIS

La virtualización y acceso remoto a los laboratorios optimiza los recursos de las instituciones educativas mejorando la calidad de la enseñanza y ampliando la oferta educativa y a su vez contribuye en la disminución del riesgo de contagios por enfermedades como el COVID-19.

5. MARCO TEÓRICO

5.1. SISTEMA AUTOMÁTICO INDUSTRIAL

Un sistema automático industrial es un conjunto de dispositivos eléctricos, mecánicos, electrónicos, o la combinación de ellos, los cuales pueden estar interconectados entre sí para controlar un proceso [1].

El objetivo de este sistema es que la variable que se desea controlar se mantenga dentro de un margen de error, y en otros casos más críticos, que sea cero. El error se define como la diferencia entre el valor medido (variable de proceso) y el valor deseado o de referencia (Set Point). Cuando el error es diferente de cero se utiliza esta información para realizar ajustes en tiempo real para reducir o eliminarlo lo más pronto posible, teniendo en cuenta que esto depende de la estrategia de control que se está utilizando y de la naturaleza del proceso que se desea controlar.

La automatización es la tecnología que se encarga de aplicar la electrónica, mecánica e informática en un sistema o proceso que se desea controlar, estos sistemas pueden ser:

- Máquina-herramientas para procesar partes metálicas.
- Robots industriales.
- Sistemas de inspección automáticos para el control de calidad.
- Maquinaria para procesos industriales.

5.2. El Control de Procesos

El control de procesos consiste en dos funciones claramente diferenciadas: la adquisición de datos y el control. Para producir un producto que sea de alta calidad de forma consistente es necesario un estricto control del proceso de producción.

El control de proceso es el control automático de una variable de salida al detectar la amplitud del parámetro de salida del proceso y comparándolo con el nivel deseado o establecido y retroalimentando una señal de error para controlar una variable de entrada [2].

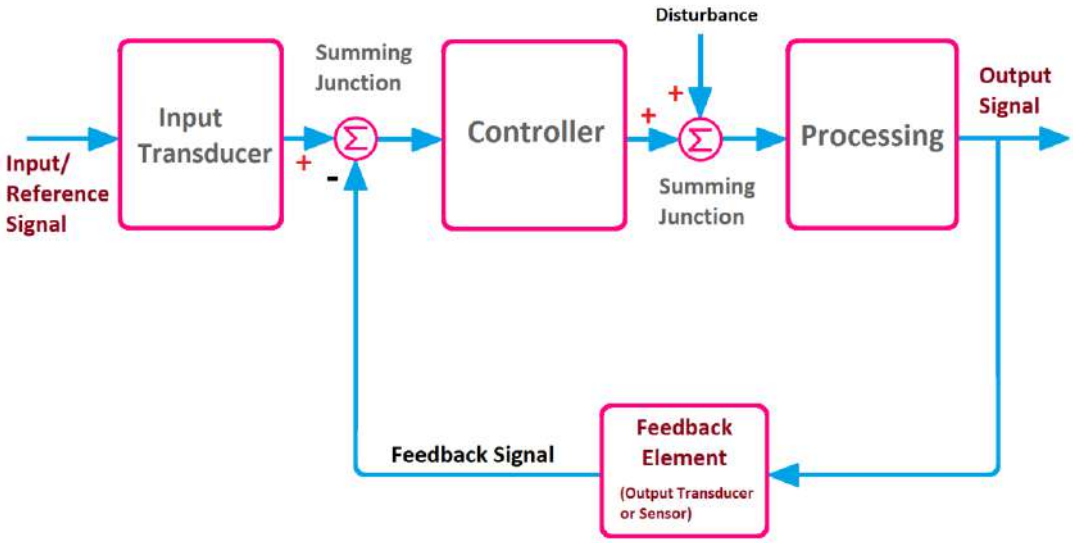

Fig. 4. Diagrama típico de un proceso controlado.

En todo proceso hay una serie de insumos, es decir, desde productos químicos hasta mercancías sólidas. Estos son manipulados en el proceso y un nuevo producto químico o compuesto emerge en la salida. Las entradas y salidas controladas del proceso se denominan variables como se muestra en la fig. 4. En una instalación de control de procesos, el controlador no está necesariamente limitado a una variable por lo que puede medir y controlar muchas variables. La mayoría de las variables controladas representan a seis u ocho dispositivos como por ejemplo los cilindros de un motor de combustión.

5.3. SENSORES Y TRANSDUCTORES

El término sensor se refiere a un elemento que produce una señal relacionada con la cantidad que se está midiendo. Por ejemplo, si se está midiendo la temperatura con un sensor resistivo, la cantidad que se mide es temperatura y el sensor transforma la entrada en cambios en la resistencia eléctrica. Por lo general se utiliza el término transductor en lugar de sensor.

Un transductor se define como el elemento que al someterlo a un cambio físico experimenta un cambio relacionado, por lo que los sensores son en realidad transductores. Sin embargo, un sistema de medición puede utilizar transductores, además de sensores, en otras partes del sistema para convertir señales de una forma dada en otra distinta. Por otra parte, el término digital se utiliza cuando los sistemas ofrecen salidas que son digitales por naturaleza.

Los siguientes términos describen las características de funcionamiento de los transductores y de los sistemas de medición:

- **Intervalo y extensión**: Define los límites entre los cuales puede variar la entrada.
- **Error**: Es la diferencia entre el resultado de una medición y el valor verdadero de la cantidad que se mide:

$$error = valor\ medido - valor\ real \quad (1)$$

- **Exactitud**: Es el grado hasta el cual un valor producido por un sistema de medición puede estar equivocado. Es, por lo tanto, la suma de todos los errores posibles más el error en la exactitud de la calibración del transductor.
- **Sensibilidad**: Es la relación que indica cuanta salida se obtiene por unidad de entrada, es decir, salida/entrada. Por ejemplo, un termómetro de resistencia puede tener una sensibilidad de 0.5Ω/°C.
- **Error por histéresis**: De produce cuando los transductores producen salidas distintas a partir de la misma cantidad medida ya sea por incremento o decremento continuo.
- **Error por no linealidad**: Se supone que, en su intervalo de funcionamiento, la relación entre la entrada y la salida del transductor es lineal. Sin embargo, esto no es así, por lo que este error se define como desviación máxima respecto a la línea recta.
- **Repetibilidad/reproducibilidad**: Describe la capacidad del transductor para producir la misma salida después de aplicar varias veces el mismo valor de entrada. El error relacionado consiste en no obtener la misma salida después de aplicar el valor de entrada, se representa como un porcentaje del intervalo total de salida:

$$repetibilidad = \frac{val.\max - val.min}{intervalo\ total} * 100 \quad (2)$$

- **Estabilidad**: Es la capacidad que posee un transductor para producir la misma salida cuando se usa para medir una entrada constante en un período.
- **Banda/tiempo muerto**: Es el intervalo de valores de entrada para los cuales no hay salida.

- **Resolución**: es el cambio mínimo del valor de entrada capaz de producir un cambio observable en la salida.
- **Impedancia de salida**: Cuando un sensor produce una salida eléctrica se conecta con otro circuito electrónico, es necesario conocer su impedancia de salida ya que esta se va a conectar en serie o en paralelo. Si no se toma en cuenta la impedancia de salida al conectar el sensor, el funcionamiento del sistema puede verse afectado.

Sensores capacitivos de proximidad

Son sensores que no requieren del contacto físico con el material a detectar. Los sensores de proximidad capacitivos permiten la detección tanto de materiales conductivos como no conductivos, pero se usan principalmente para detectar materiales como plásticos, líquidos, materiales granulados, etc.

Fig. 5. Sensor de proximidad capacitivo.

En las aplicaciones de sensores de proximidad capacitivos, la sensibilidad de estos depende de las características del material objeto. Estas características de los metales, plásticos, líquidos, etc. a detectar influyen en la distancia de detección y el punto de conmutación del sensor [3].

Válvula Proporcional

Las válvulas proporcionales controlan la presión o el caudal de un sistema proporcionalmente a la señal eléctrica de entrada.

El embrague impulsor de la válvula es inversamente proporcional al flujo de aceite y la presión que se maneja en ella, es decir, que un aumento de la corriente resulta en la disminución del flujo de aceite del embrague y por tanto de la presión.

El aceite de la bomba fluye por el centro del carrete de la válvula, pasa el orificio y la bola, y pasa al drenaje. El resorte de la válvula mueve hacia la izquierda el carrete de la válvula. El carrete de la válvula bloquea el conducto entre el embrague impulsor y la bomba, y abre el conducto entre el embrague impulsor y el drenaje. El flujo de la bomba al embrague impulsor se bloquea. El aceite del embrague impulsor fluye y pasa el carrete de la válvula al drenaje.

Fig. 6. Válvula proporcional Burkert 6223 utilizada en el proyecto.

Cuando se desactiva el solenoide del embrague impulsor, el resorte mueve el conjunto del pasador contra la bola. La bola bloquea el flujo de la bomba, a través del orificio, al drenaje. La presión de aceite aumenta en el extremo izquierdo del carrete de la válvula y lo mueve a la derecha contra el resorte. El carrete de la válvula bloquea el conducto entre el embrague impulsor y el drenaje, y abre el conducto entre el embrague impulsor y la bomba. El aceite de la bomba fluye y pasa el carrete de la válvula al embrague impulsor. En este tipo de válvula, un aumento de la corriente resulta en una disminución del flujo al embrague, y por lo tanto de la presión.

Medidores de Flujo

Los medidores de flujo son instrumentos que controlan, miden o registran la tasa de flujo, el volumen o la masa de un gas o líquido. Aportan un control y/o monitoreo preciso de lo que pasa por un caño o una tubería, incluyendo agua, aire, vapor, aceite, gases y otros líquidos. Los medidores de flujo específicos para una aplicación permiten a los gestores de instalaciones, contratistas de control, ingenieros consultores y otras partes interesadas:

- Entender y controlar las operaciones de flujo.
- Identificar y mejorar las eficiencias.
- Abordar los problemas del equipo y el uso irresponsable.

Los medidores de presión diferencial miden el flujo de líquido dentro de una tubería introduciendo una constricción que cree una caída de presión. Los sensores de presión miden la presión antes y después de la constricción. La caída de presión resultante que se produce a lo largo de la constricción es relativa a la tasa de flujo al cuadrado; mientras mayor sea la caída de presión, mayor es la tasa de flujo.

Los medidores de presión diferencial son adecuados para aplicaciones que incluyen filtros, intercambiadores de calor, dispositivos de prevención de reflujo, tuberías y conductos, entre otros. Una razón clave por la que los gestores de las instalaciones prefieren los medidores DP se debe a que no tienen partes móviles, lo que significa que requieren un mantenimiento mínimo [4].

5.4. Realidad Virtual, Realidad Aumentada y Realidad Mixta

Realidad aumentada y realidad mixta son términos con relativamente poco tiempo, ambos parten desde uno que es bastante más antiguo, el de realidad virtual. Este concepto, aunque se comenzó a llamar así en los años 80, tiene más tiempo del que en principio nos pudiésemos imaginar, pues las primeras investigaciones en este campo se hicieron en los años 50 y las primeras máquinas de realidad virtual estuvieron disponibles a principios de los 60. Las primeras máquinas estaban basadas, como en la actualidad, en la estereoscopía o *visión estereoscópica*. Sin entrar por el momento en mucho detalle, diremos que consiste en disponer de dos imágenes desde dos puntos de vista distintos, cada una de los cuales se correspondería al punto de vista de uno de los ojos y que al mostrarle a cada ojo su correspondiente imagen por separado, el cerebro las ordena creando la sensación de tridimensionalidad. El uso de la estereoscopía es mucho más antiguo que la realidad virtual, en la siguiente imagen podemos ver dos fotografías para ser vistas en 3D mediante un visor estereoscópico, las imágenes datan de 1875 y forman parte de una colección del fotógrafo Robert N. Dennis que está compuesta por algo más de 42.000 imágenes estereoscópicas y que puede consultarse de forma digital en la página web de colecciones digitales de la Biblioteca Pública de New York [5].

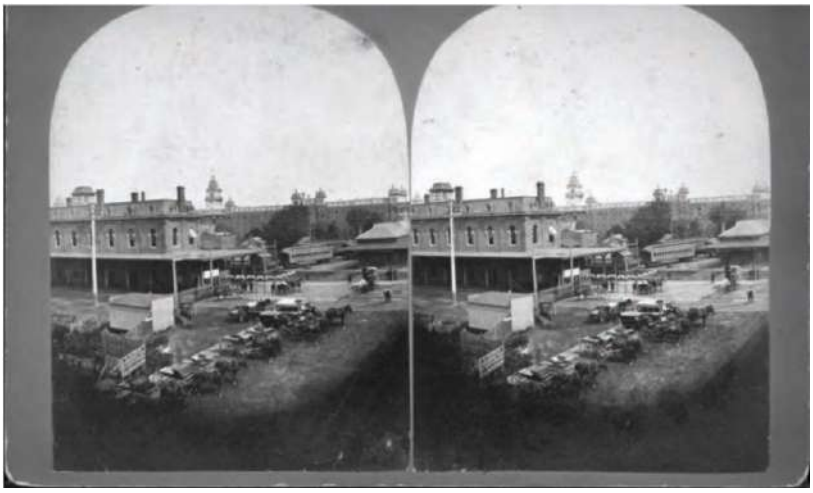

Fig. 7. Fotografía de 1857 lista para ser vista en 3D mediante un visor estereoscópico.

Realidad

El concepto de realidad hace referencia a todo lo existente, es decir, a todo lo que no forma parte de la imaginación y que tiene la propiedad de existir porque puede ser percibido por alguno de los sentidos o por la razón. La realidad es un concepto debatido en filosofía, sociología y psicología, puesto que se han generado debates en diferentes espacios culturales y académicos con respecto a qué es real y qué no. Hoy en día, la realidad está considerada como un conjunto de sujetos y objetos que interaccionan en un sistema; el concepto de realidad ha estado presente en el desarrollo humano y es una de las incógnitas por resolver [6].

Virtualidad

La virtualidad hace referencia a la capacidad de reproducir un efecto, aunque no se produzca en tiempo real o presente. A su vez, se encuentra asociada con lo que tiene existencia de forma aparente, por ejemplo, los medios de mensajería instantánea, las capacitaciones en plataformas digitales, la comunicación por medio de correos electrónicos, todo esto de forma telemática, que se hace en tiempo

real, pero sin presencialidad. En informática se habla de una realidad construida con sistemas digitales, orientada hacia el diseño e implementación de esquemas informáticos y en la interacción persona-computador, respectivamente, como en realidad virtual (RV). Esto quiere decir que los modelos tradicionales de comunicación, de interacción, de entretenimiento y hasta de trabajo se han venido modificando gracias a la era digital que se ha estado generando a escala mundial. Esto lleva a reaprender sobre los contenidos que, en su mayoría, ya no son impresos o transmitidos de forma convencional, sino que para acceder a ellos se requieren habilidades tecnológicas. En la figura 8 se muestran algunos beneficios de la virtualidad [7]

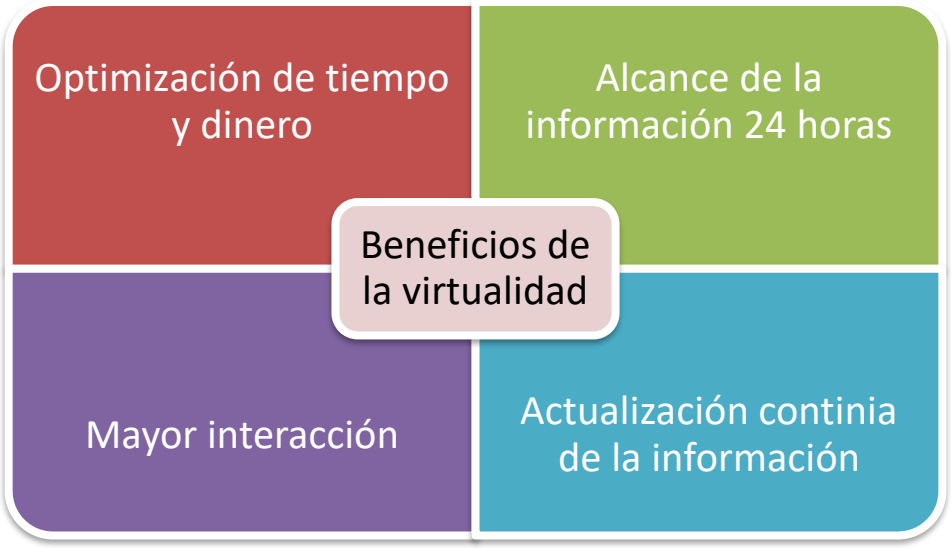

Fig. 8. Beneficios de la virtualidad.

Fig. 9. Fotografía de montaña esférica inmersiva, compatible con las Google Cardboard.

Sistemas de aplicaciones de realidad virtual

Existen diferentes tipos de aplicaciones para el uso de la RV inmersiva:

- Sistema operativo Android: Google Play.
- Sistema operativo iOS: Vr-iPhone.
- Para Apps con sistema operativo Windows Phone: Microsoft Store.
- Videos inmersivos en 360º por medio de YouTube y en Vimeo: Canal de RV.
- Uso de fotografías esféricas inmersivas: Street View y Flickr 360°.

Una vez explorados los diferentes tipos de contenidos que se tienen a disposición, se debe recordar que, para utilizarlos a nivel educativo, de forma inmersiva, además de tener en cuenta las edades mínimas a partir de las cuales es apropiado o no el uso de cada visor de RV, siempre se deberá revisar y valorar previamente la adecuación de los contenidos que se les van a ofrecer a los estudiantes, sobre todo si son menores de edad. El *Pan European Game Information (PEGI)* es un sistema europeo para clasificar el contenido de los videojuegos y otro tipo de software de entretenimiento. Se aplica en 25 países sin tener relación alguna con la Unión Europea; en Colombia no se conoce ningún referente en estos temas, aunque es importante tenerlo para el control de contenidos. Tampoco se sabe de un sistema de control para usos académicos [8].

Realidad Mixta

La realidad mixta (MR) representa un concepto un poco más complejo, pero con el que nos iremos familiarizando poco a poco a lo largo de los próximos años. En la MR lo que hacemos ya no es superponer información sobre el mundo real, sino fusionar el mundo físico con el mundo digital. Esto quiere decir que, si tenemos un elemento, como puede ser una silla modelada en 3D, vamos a poder colocarla en el mundo físico y esa silla va a "ser consciente" del mundo que le rodea: va a entender dónde está el suelo y, si pasa alguien por delante, va a tapar dicha silla. Esto no sucedía con la AR, por lo que ahora la sensación va a ser mucho más inmersiva: le va a afectar la iluminación del entorno y todo se va a ir adaptando de forma que podamos llegar a tener un mundo indistinguible que mezcle lo físico y lo digital. Se trata de una tecnología que está empezando ahora, pero para la que "es fundamental empezar a comprender desde ya todos los conceptos y desarrollos de la VR si queremos ser los primeros en adaptarnos a esta nueva ola que está por llegar".

Realidad Virtual

La realidad virtual (RV) describe al conjunto de tecnologías inmersivas que buscan posicionar al usuario dentro de entornos virtuales simulados por ordenador. Dependiendo del objetivo que se pretenda alcanzar con las simulaciones, es posible que las imágenes sean realistas o no. Para llevarse a cabo, la RV hace uso de dispositivos llamados lentes o cascos de realidad virtual. Estos hacen posible que los usuarios perciban escenarios en 360° con alta definición.

La incorporación de audio y sensores de movimiento, permiten una interacción única con el entorno, lo que le da a la experiencia una característica realista muy útil. Ya que un usuario puedo volverse protagonista de un escenario sin salir de un entorno controlado, las aplicaciones en el entretenimiento solo son limitadas por la imaginación.

A nivel industrial, operadores pueden familiarizarse con entornos de riesgo sin exponerse, médicos pueden practicar procedimientos complejos, equipo técnico asistir a distancia y mucho más. Antes de profundizar más en esta tecnología, es importante no confundirla con la RA (realidad aumentada).

Realidad Aumentada

El termino realidad aumentada (RA) se usa para describir una serie de tecnologías que permiten combinar en tiempo real contenido generado por ordenador con video en directo. Tradicionalmente, se diferencia de la realidad virtual (RV) en que esta implica la creación de entornos 3D completos, mientras que la RA usa diferentes tecnologías de hardware para crear una composición aumentada basada en el mundo real [9].

Tecnología requerida para trabajar con realidad aumentada

Para realizar cualquier tipo de RA visual, es imprescindible un buen ordenador (cualquiera fabricado en los 3 últimos años) con una cámara; en la actualidad muchos ordenadores ya las traen incorporadas, sin embargo, al estar integradas en el equipo resulta complicado moverlas y enfocar de modo que, aunque su equipo incorpore una cámara conviene adquirir una cámara web USB.

La realidad aumentada es la combinación de dos tecnologías, la visión artificial y los motores gráficos.

Motor gráfico

El motor gráfico se ocupa de renderizar los contenidos, típicamente en 3D, que muestra la realidad aumentada. La visión artificial se ocupa de que la ubicación de los contenidos aumentados sea la correcta en la escena, para que la composición sea coherente y entendible por el usuario.

El principal elemento que interviene en la parte gráfica sería el renderizado. Este proceso consiste en la interpretación por parte del ordenador de una escena de tres dimensiones para crear una imagen bidimensional.

La información que se procesa para realizar el render es la geometría del modelo 3D, las características de su superficie (color y material), la iluminación de la escena y la posición de la cámara.

Visión Artificial

Las técnicas de visión artificial que se aplican en realidad aumentada son muy variadas, y se denominan principalmente tecnología de seguimiento o tracking. Existen muchos tipos, por ejemplo, el tracking facial, que permite detectar y seguir la posición de una cara, o el tracking de texturas, que posiciona una imagen de referencia en un sistema de coordenadas tridimensional. Las técnicas de visión artificial más novedosas que se emplean actualmente incorporan sensores activos basados en luz estructurada, tracking SLAM o tracking 3D.

Para que una experiencia de realidad aumentada sea satisfactoria, debe funcionar en tiempo real. Esto quiere decir que cada uno de los dos módulos debe hacerlo también, y ambos son muy exigentes con el uso de los recursos computacionales. Así que el módulo de render debe ser capaz de pintar unas 60 imágenes por segundo, y el módulo de tracking de igual modo, ser capaz de analizar y extraer la información de 60 imágenes por segundo [10].

6. METODOLOGÍA DE INVESTIGACIÓN

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

6.1. MONTAJE DE CONTROLADOR ELECTRÓNICO DE FLUJO Y NIVEL

a. Identificar y definir requerimientos de seguridad industrial del entrenador FPC de flujo y nivel.

Se hizo una lista de los riesgos de seguridad del proceso, tanto para operarios como para los dispositivos que forman parte del entrenador. Se hicieron modificaciones en los circuitos del controlador electrónico para que sus entradas y salidas, tanto digitales como analógicas, estuviesen protegidas contra sobretensiones. Esto se hizo mediante la incorporación de dispositivos de protección ESD como se muestra en Fig. 10. En donde se aprecia una entrada analógica cuyo estándar establece que puede recibir tensiones de entre 0V y 10V. Esta entrada está protegida contra tensiones de hasta 14V por medio del diodo ESD D15.

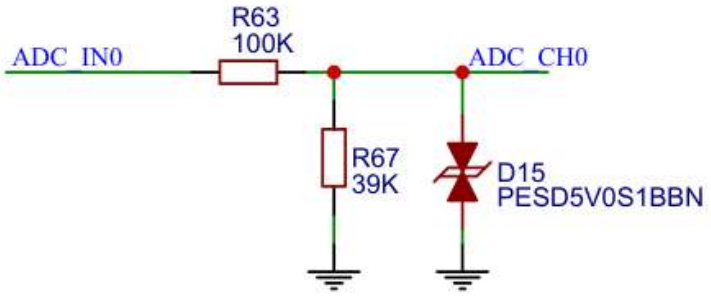

Fig. 10. Entrada analógica con protección ESD por medio de D15.

Adicionalmente, se incorporó a la fuente del controlador de un circuito de protección contra inversiones de tensión de entrada (Fig. 11). En este caso, si el operario invierte las líneas de entrada de la fuente de +24V, el diodo MOSFET de protección evitará que dicha inversión de polaridad pase hacia los demás circuitos que la fuente alimenta.

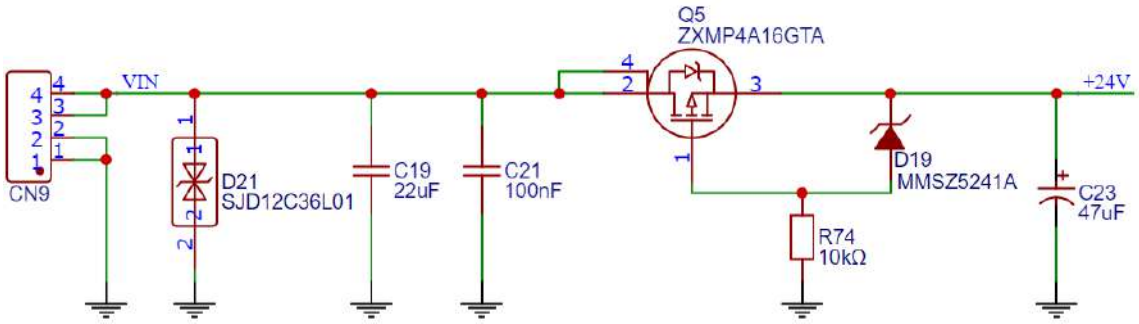

Fig. 11. Circuito de protección contra inversión de polaridad con MOSFET canal P.

Además, se determinó que es importante informar a los operarios y al personal cercano sobre el estado operativo del entrenador, ya que como es un equipo destinado a manipularse de forma remota, existe la posibilidad de que esté siendo utilizado y que las personas que se encuentren en el laboratorio no se enteren de esto.

Para solventar este requerimiento, se dotó al equipo de una luz de estado que mediante blinks (secuencias de apagado/encendido), que indica a las personas que se encuentran cerca sobre el estado operativo de este.

Se utilizó una luz táctil de grado industrial con activación PNP conectada a una salida digital del controlador electrónico cuya codificación de blinks y su relación con el estado del equipo es el siguiente:

Patrón de blinks	Estado
1	Paro
2	Reinicializado
3	Esperando
4	Iniciando
Encendido	Marcha
Apagado	No energizado

Tabla 1. Patrón de blinks de la luz de estado del entrenador.

Como la luz táctil utilizada es PNP y las salidas del controlador son NPN, fue necesario diseñar un circuito de acople NPN-PNP (Fig. 12) que puede ser utilizado para otros casos en donde se tenga esta incompatibilidad.

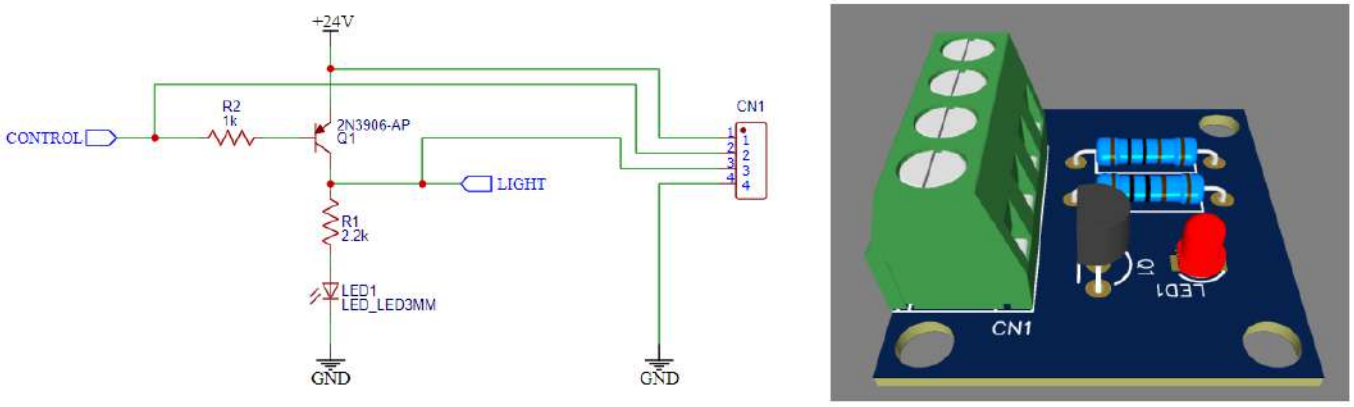

Fig. 12. Circuito de acople digital NPN a NPN.

b. Diseño del PCB utilizando software CAD y encargo para fabricación.

A partir del diagrama esquemático del controlador electrónico se diseñó el PCB utilizando el programa EasyEDA que se ejecuta por medio de un navegador Web y que es de acceso gratuito.

Para el diseño del PCB se consideró desde un inicio hacerlo por encargo, así como el montaje de los componentes de montaje superficial (SMD). Esto implicó que cada uno de los componentes de los circuitos fueron elegidos de la librería de dispositivos de la empresa JLCPCB, quien fue la elegida para fabricarlo y soldar los componentes.

El PCB es multicapa por lo que cuenta con cuatro capas de cobre:

TOP, es la capa donde se interconectan las líneas de alimentación y de comunicación de los diversos dispositivos del circuito (Fig. 13).

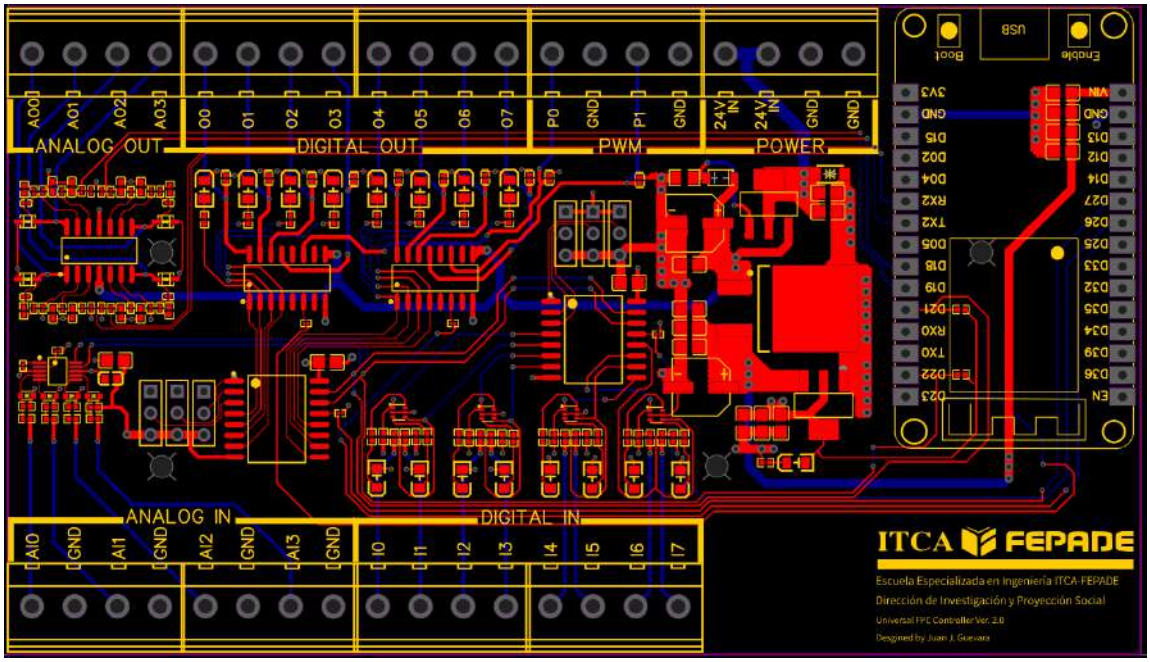

Fig. 13. Capa Top del PCB del controlador electrónico.

En la Fig. 13 se puede identificar fácilmente el área destinada a la fuente de alimentación, ya que representa las zonas de las pistas de cobre más anchas, debido a que por ellas circula una mayor cantidad de corriente. Por otra parte, entre los diversos circuitos integrados se observan pistas mucho más delgadas, de hasta 0.127 mm, que se han destinado a las líneas de comunicación en las cuales circulan corrientes muy pequeñas.

BOTTOM, es la capa que se destinó para interconectar las líneas de entrada y salida (Fig. 14).

La parte inferior del PCB corresponde a las entradas analógicas (8) compatibles con el estándar de 0 – 10V y digitales (8) de +24V tipo NPN.

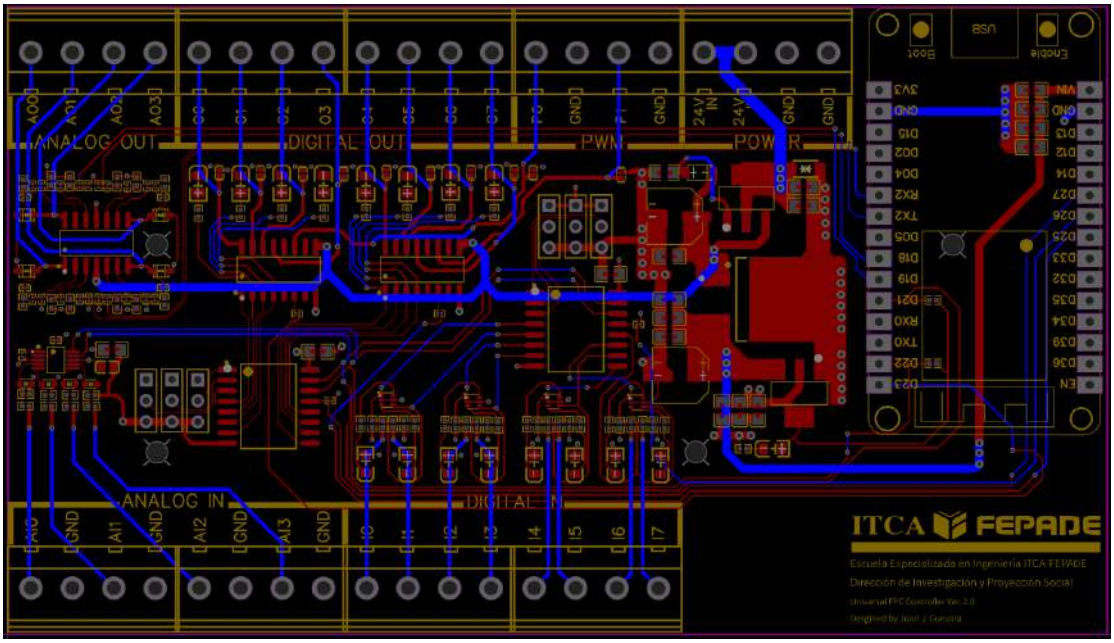

Fig. 14. Capa Bottom del PCB del controlador electrónico.

INNER 1, es la tercera capa que se utilizó para distribuir la tensión de +3.3V a todos los dispositivos que la requerían.

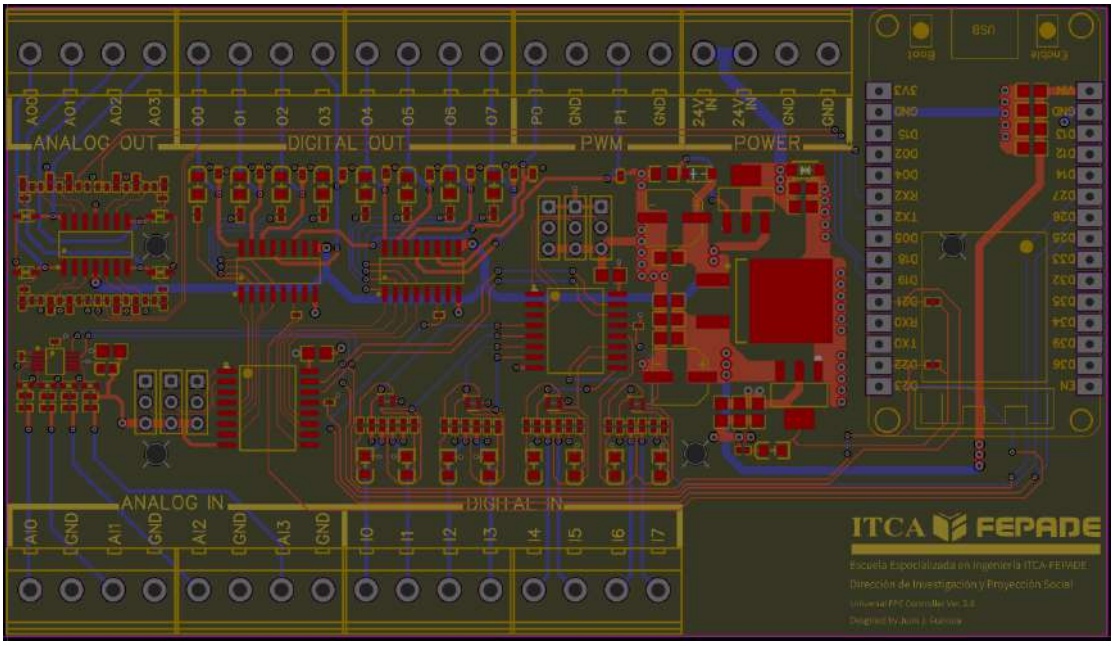

Fig. 15. Capa INNER 1 para fuente de +3.3V del PCB del controlador electrónico.

INNER 2, es la cuarta capa y fue utilizada como referencia a tierra o GND.

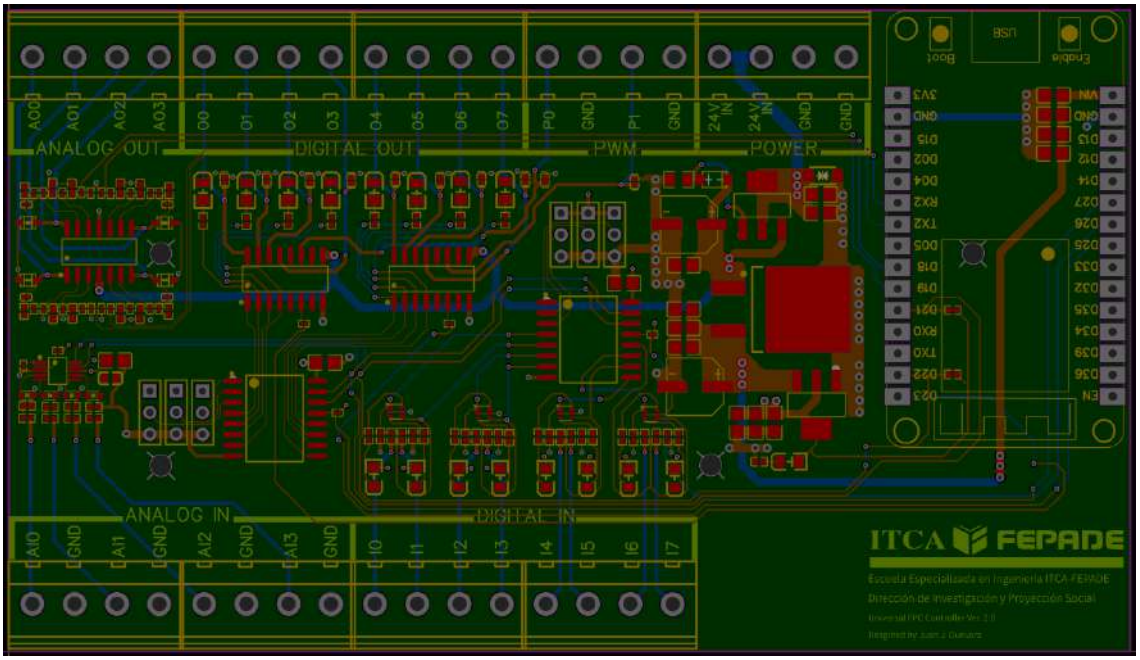

Fig. 16. Capa INNER 2 para GND del PCB del controlador electrónico.

Finalmente, se obtuvo una vista preliminar del PCB en 3D con lo cual se procedió a generar los archivos GERBER que son los utilizados para fabricarlo y son requeridos para la empresa manufacturera.

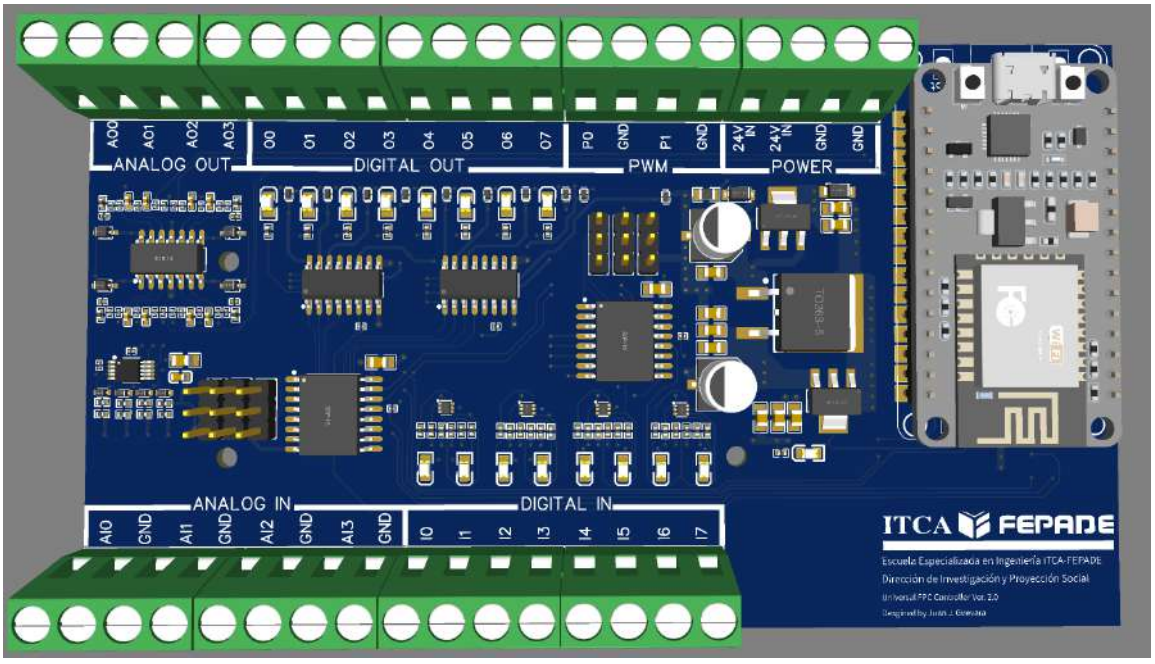

Fig. 17. Vista 3D del PCB del controlador electrónico.

Finalmente, el PCB fabricado y ensamblado fue recibido de JLCPB se muestra en Fig. 18.

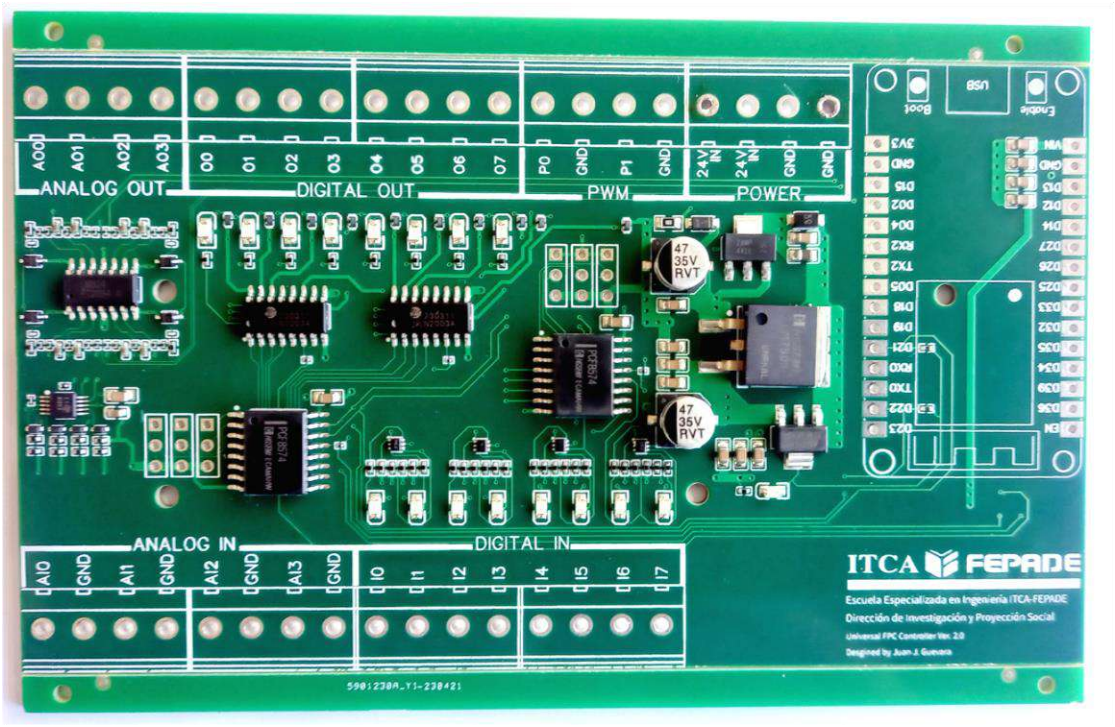

Fig. 18. PCB del controlador electrónico con dispositivos SMD soldados.

c. Montaje de dispositivos electrónicos en el PCB del controlador y pruebas eléctricas de funcionamiento.

En Fig. 18, se aprecia que, desde la fábrica, el circuito PCB tiene soldados únicamente los dispositivos SMD. Fue necesario soldar los dispositivos THT que corresponden a los terminales con tornillo para las entradas y salidas, así como los conectores pin header macho y hembras correspondientes. Las pruebas eléctricas consistieron en realizar mediciones de continuidad para verificar la existencia de corto circuitos y aislamiento de las capas. Posteriormente se energizó el circuito con una fuente de +24V 5A y se invirtieron las polaridades para verificar el funcionamiento de las protecciones.

d. Problemas detectados y mejoras aplicadas.

Como producto de las pruebas eléctricas de funcionamiento se determinaron dos potenciales problemas:

- El expansor de puerto PCF8574T de las entradas interfería con la correcta detección de los estados alto y bajo. Esto se debe a que, por diseño, este circuito integrado, al inicializarse, entra en conflicto con los posibles estados presentes en las entradas.
- El expansor de puerto PCF8574T de las salidas no produce suficiente corriente para activar satisfactoriamente las cargas de salida conectas al circuito integrado driver ULN2003A.

La solución a estos problemas fue sustituir los expansores de entrada y salidas colocando en su lugar un microcontrolador como controlador de entradas y salidas cuyas funciones son:

- Comunicarse con el microcontrolador principal ESP32 WROOM32 bajo demanda cuando se detecte un cambio de estado en las entradas.
- Recibir los cambios de estados del microcontrolador principal para aplicarlos a las salidas digitales.

El microcontrolador elegido fue el RP2040 ya que una investigación exhaustiva de sus especificaciones técnicas [11] determinó la compatibilidad con el microcontrolador principal y viene embebido en una placa de desarrollo Raspberry Pi Pico con suficientes entradas y salidas digitales, así como puertos de comunicación configurables compatibles con el bus I^2C [12]. Se diseñó el diagrama esquemático que se muestra en Fig. 19, en donde se puede apreciar que se dotó al circuito de una fuente reguladora de +5V para ajustar el voltaje de entrada de +24V.

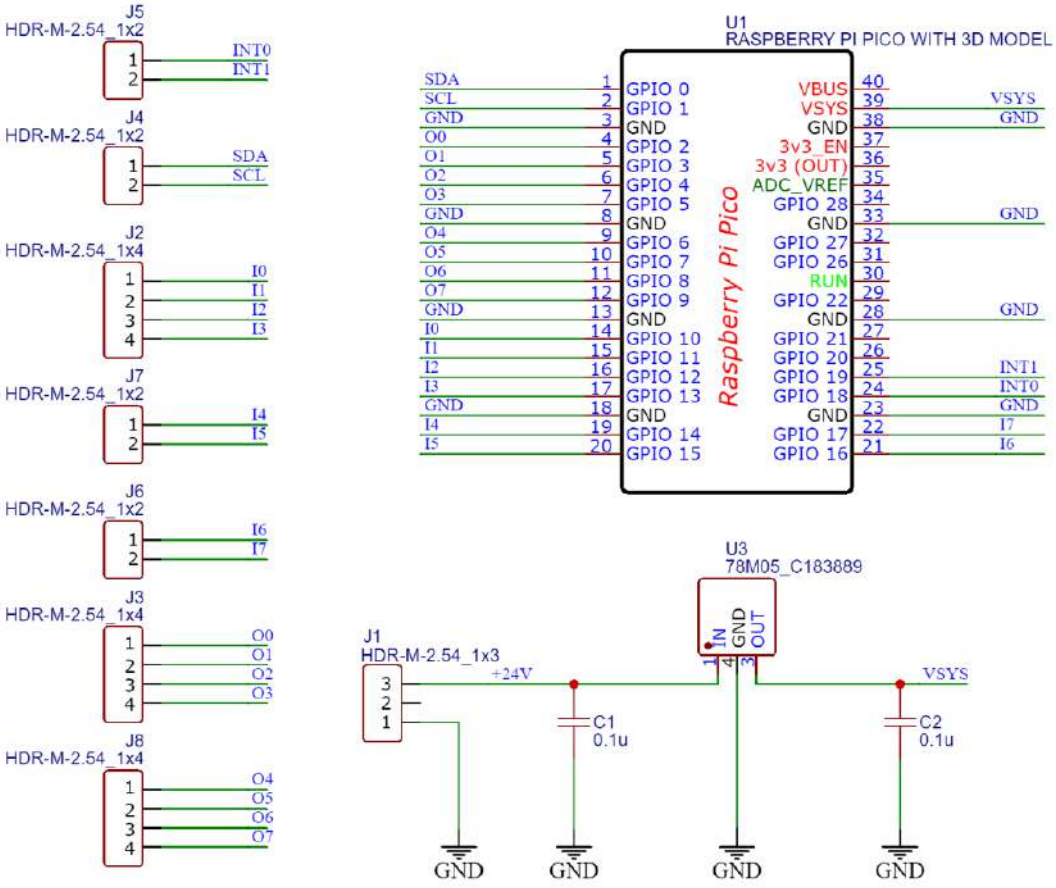

Fig. 19. Diagrama esquemático del controlador de entradas y salidas basado en RP2040.

Finalmente, el circuito del controlador de entradas y salidas se montó con el controlador principal y se realizaron pruebas eléctricas de compatibilidad siendo estas satisfactorias.

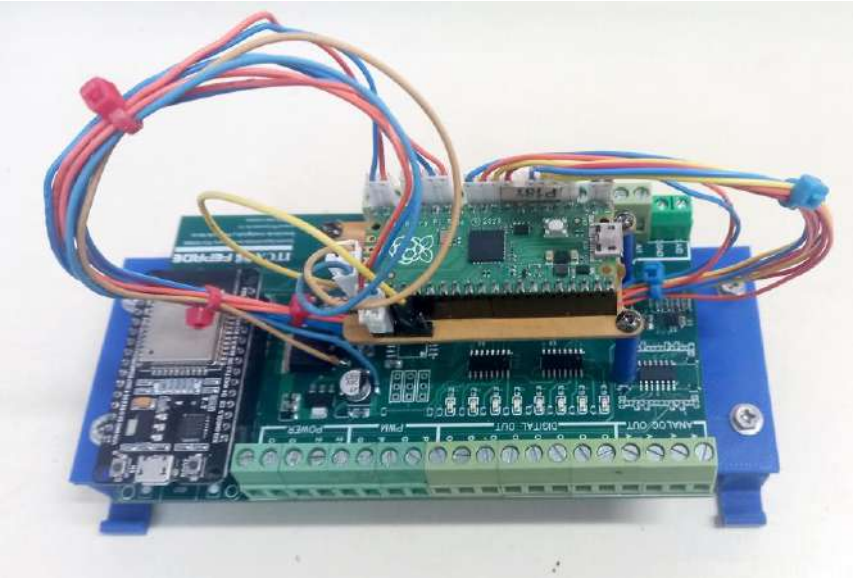

Fig. 20. Montaje del controlador de entradas y salidas en el circuito principal.

6.2. Diseño del Firmware del Controlador Electrónico

El diseño del firmware o programa del controlador se dividió en dos etapas:

a. Diseño del firmware del controlador de entradas y salidas.

El programa se creó utilizando el framework SDK de Raspberry Pi Pico que está basado en el lenguaje C. Si bien la plataforma está diseñada para trabajar con Python, la experiencia obtenida en el proyecto de investigación del año 2021 en donde el firmware del controlador se creó con Micropython, nos demostró que, al ser un lenguaje de muy alto nivel, el código resultante es demasiado grande y es lento para controlar multiprocesos. En cambio, el lenguaje C es el ideal para crear aplicaciones con microcontroladores en donde se manejan múltiples procesos de forma asíncrona.

El firmware de este controlador tiene como función realizar tres tareas:

1. **Monitorear el estado de las entradas digitales**, donde una función temporizada que se ejecuta cada 10 ms se encarga de monitorear constantemente el estado de las entradas digitales. Cuando se detecta un cambio de estado, se activa una interrupción por hardware que avisa al controlador principal que se ha producido un cambio en las entradas y que este debe preguntar al controlador de entradas y salidas, por medio del bus I^2C, cuál de las entradas ha sido la afectada.
2. **Cambiar el estado de las salidas digitales**, donde una función se encarga de cambiar el estado de las salidas cuando el controlador principal ha enviado un comando con la orden de cambio. Esta función se encarga de determinar cuál es la salida afectada y realizar los cambios de estado correspondientes.
3. **Mantener comunicación por el bus I^2C con el controlador principal**, una función manejadora de eventos (*handler*) se encarga de detectar cuando se produce un evento de comunicación que viene desde el máster del bus I^2C (el controlador principal), este controlador de eventos es el encargado de determinar si la acción a realizar está relacionada con la lectura del estado de las entradas con algún cambio en las salidas. Esta función es asíncrona por lo que se ejecuta bajo demanda del controlador principal.

El diagrama de estados del firmware del controlador de entradas y salidas se muestra en la siguiente figura:

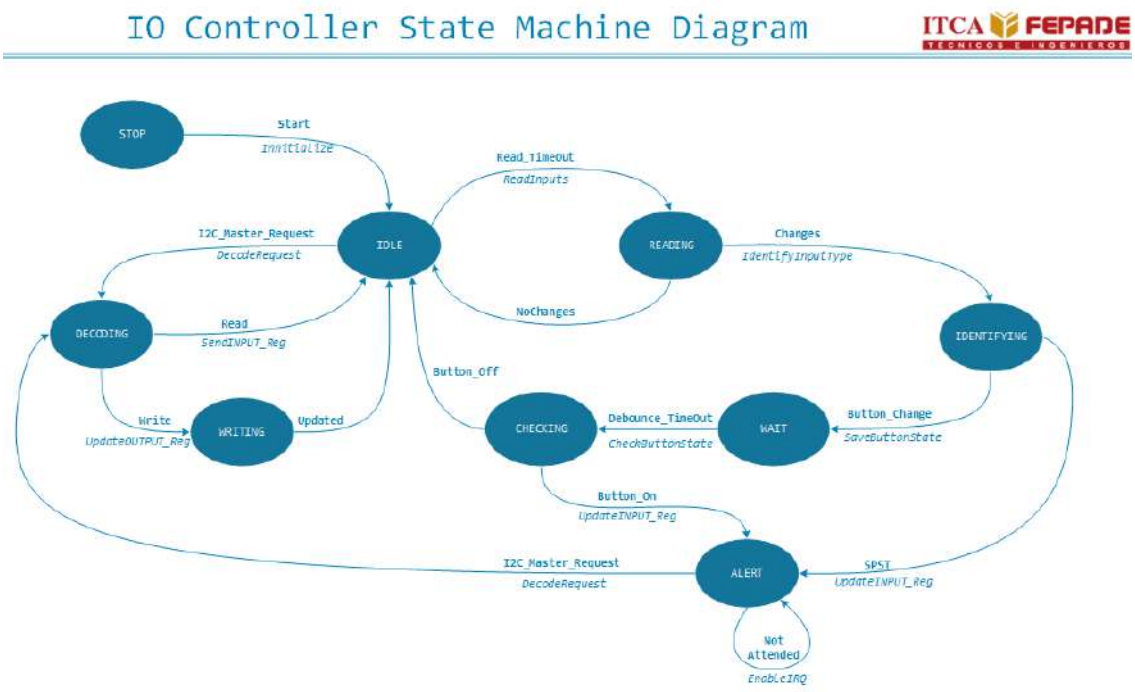

Fig. 21. Diagrama de estados del firmware del controlador IO.

b. Diseño del firmware del controlador principal.

Para la elaboración de este programa se tuvo como principal objetivo el garantizar que las tareas se realizaran, en la medida de lo posible, bajo demanda de forma asíncrona, para acciones que requieren acceso a redes de datos o síncronas lanzadas por medio de un temporizador con un período base de 100 ms. El temporizador principal se inicializa con las siguientes instrucciones:

```
  #define   TIMER_I2C_INTERVAL  100000  //100 mS

  // Timer 0, divisor de reloj 80
  timer = timerBegin(0, 80, true);
  // Adjuntar la función de manejo de interrupción
  timerAttachInterrupt(timer, &timer_i2c, true);
  //Establecer el tiempo de activación
  timerAlarmWrite(timer, TIMER_I2C_INTERVAL, true);
  timerAlarmEnable(timer); // Habilitar la alarma
```

Cada vez que se activa el temporizador se lanza la siguiente función que es donde se activan las banderas que controlan la ejecución de las tareas:

```
void IRAM_ATTR timer_i2c() {
  update_outputs = true;
  read_adc_value = true;
  compute_pid = true;
  check_states = true;
  refresh_states = true;
  modbus_request = true;
}
```

A partir de esta base de tiempos se programaron las funciones como tareas para ser invocadas a diversos intervalos de tiempo de acuerdo con su importancia dentro de todos los procesos.

Las principales tareas del programa principal son las siguientes:

1. **Manejar la comunicación con el controlador de entradas y salidas.**

La comunicación debe se establece a partir de dos condiciones:

- La primera es cuando el controlador de entradas y salidas detecta un cambio de estado en sus entradas digitales. En este caso, activa una interrupción por hardware que pone en alto a PIO26 del ESP32 que corresponde a la interrupción PCF01_INT.

  Mediante la siguiente instrucción:

```
//Función para atender la IRQ del controladdor ES
void IRAM_ATTR irq_request() { read_inputs = true; }
```

  Se define que cuando se active una interrupción interna, la bandera read_inputs se active de manera que en la siguiente verificación de la lectura de las entradas se realiza la transacción correspondiente por el bus I^2C.

```
  if(read_inputs) {
    c_i2c_read++;
    if(c_i2c_read == I2C_READ_INTERVAL) {
      read_i2c_inputs();
      c_i2c_read = 0;
      read_inputs = false;
    }
  }
```

  Como se aprecia en el código anterior, la constante `I2C_READ_INTERVAL` establece el intervalo de tiempo de lectura del estado de las entradas desde el controlador de entradas y salidas, el valor de esta constante es 5 lo que equivale a que se ejecuta cada 500 ms. La función read_i2c_inputs que se invoca tiene la siguiente estructura:

```
void read_i2c_inputs() {
  Wire.requestFrom(I2C_ES_CONTROLLER, 1);  //Leer un byte
  while(Wire.available()) {
    input_reg = Wire.read();
    //Mostrar registro, solo para efectos de prueba
    printf("Entradas: %d\n", input_reg);

    //Verificar si hay cambios
    if(input_reg != input_reg_bk) { find_input_change(input_reg); }
  }
  read_inputs = false;
}
```

  De esta manera, se consigue identificar los cambios en las entradas solo cuando estas ocurran, liberando espacio en el proceso del microcontrolador principal.

- La segunda condición se da cuando se cambia el estado de las salidas como parte del proceso que se está controlando ya sea de forma automatizada (establecido en los estados de máquina del proceso) o por acción directa del usuario.

  La mejor manera de programar la acción de cambio en las entradas es utilizar registros de estado actual y anterior de las salidas, de manera que se comparen cada cierto intervalo de tiempo y, en caso de existir diferencias, se procede a indicar al controlador de entradas y salidas que cambie el estado de la salida correspondiente.

  Una condición se utiliza para determinar si ya es tiempo de verificar si deben aplicarse cambios en las salidas siguiendo la misma lógica que para el caso de las entradas:

```
  if(update_outputs) {
    c_i2c_outputs++;
    if(c_i2c_outputs == I2C_OUTPUT_INTERVAL) {
      find_output_change(output_reg);
      update_i2c_slave_outputs();
      c_i2c_outputs = 0;
    }
    update_outputs = false;
  }
```

  La constante `I2C_OUTPUT_INTERVAL` es utilizada para definir el intervalo de tiempo para realizar la verificación, experimentalmente se comprobó que el valor más efectivo para esta constante es 2, lo que equivale a que la comprobación se realiza cada 200 ms. Ahora, la función `find_output_change` se encarga de verificar si existen cambios:

```
void find_output_change(unsigned char reg) {

  //printf("Buscando cambios\n");

  if(P_BOMB) { reg = reg | 0B00000001; }
  else { reg = reg & 0B11111110; }

  if(P_SV1) { reg = reg | 0B00000010; }
  else { reg = reg & 0B11111101; }

  if(P_SV1_LIGHT) { reg = reg | 0B00000100; }
  else { reg = reg & 11111011; }

  if(P_SV2) { reg = reg | 0B00001000; }
  else { reg = reg & 0B11110111; }

  if(P_SV2_LIGHT) { reg = reg | 0B00010000; }
  else { reg = reg & 0B11101111; }

  if(P_UPLS_LIGHT) { reg = reg | 0B00100000; }
  else { reg = reg & 0B11011111; }

  if(P_STATUS_LIGHT_R) { reg = reg | 0B01000000; }
  else { reg = reg & 0B10111111; }

  output_reg = reg;
}
```

En caso de existir cambios, la función `update_i2c_slave_outputs` se encarga de aplicarlos comunicándose convenientemente con el controlador de entradas y salidas por medio del bus I^2C.

```
void update_i2c_slave_outputs() {

  //Verificar si hay cambios que aplicar en las salidas vía esclavo i2c
  if(output_reg != output_reg_bk) {

    //Enviando valor para actualizar salidas
    Wire.beginTransmission(I2C_ES_CONTROLLER);
    Wire.write(output_reg);
    Wire.endTransmission();

    output_reg_bk = output_reg;
  }
  update_outputs = false;
}
```

Como puede concluirse, al igual que en el caso de las entradas, las salidas únicamente se cambian bajo demanda liberando al procesador de repetir instrucciones de forma innecesaria.

2. Leer las entradas analógicas.

El controlador fue diseñado para manejar hasta cuatro entradas analógicas con una resolución de 16 bits lo que equivale a 65536 valores discretos. A nivel de hardware el convertidor ADC es un circuito integrado ADS1115 cuyo circuito se muestra en Fig. 22.

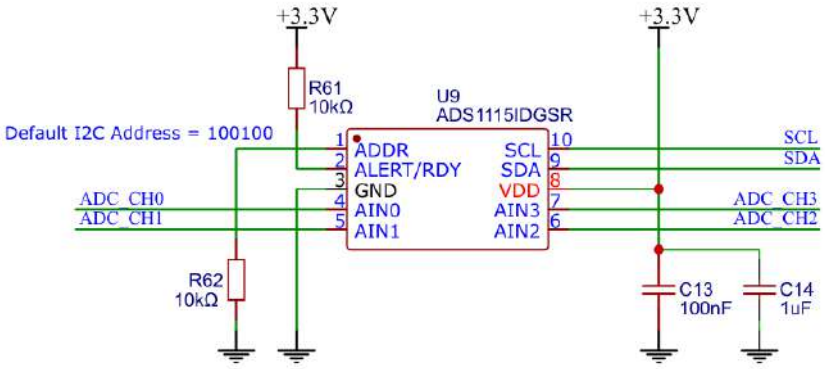

Fig. 22. Circuito convertidor de analógico a digital de 16 bits con ADS1115.

El circuito integrado trabaja con niveles de tensión de hasta 3.3V por lo que fue necesario que cada entrada fuese dotada de un circuito resistivo divisor como el mostrado en Fig. 10, este circuito adapta los voltajes de entrada de entre 0V y 10V a 0V y 3.3V respectivamente.

Para controlar al circuito conversor, se utilizó la librería ADS1X15 de Rob Tillaart que es de acceso y uso libre [13]. La bandera `read_adc_value` se utiliza para indicar cuando es momento de leer las entradas analógicas. Cada entrada debe haber sido previamente activada y configurada lo cual depende de cuantos circuitos estén conectados al controlador.

La lectura de los canales analógicos se realiza cuando el tiempo transcurrido es igual al establecido en la constante `I2C_ADC_INTERVAL` que por defecto es de 10, lo que equivale a lecturas cada un segundo.

```
  if(read_adc_value) {
    c_adc++;
    if(c_adc == I2C_ADC_INTERVAL) {
      read_adc_channels();
      c_adc = 0;
    }
    read_adc_value = false;
  }
```

La función que realiza la lectura de los valores analógicos de acuerdo con los canales que se encuentren activos:

```
void read_adc_channels() {
  float f = ADS.toVoltage(1);
  //Caudalímetro conectado al canal 0
  if(adc_ch0_enable) {  //Sensor de caudal o flujo
    //Resultado de la conversión es un entero de 16 bits
    adc_ch0 = ADS.readADC(0);
    float v = (float)(adc_ch0 * f); //Convertir a Voltaje
    v = v * 3.58;
    if(v <= 1.011) { v = 1.011; }
    //Escalamiento de voltaje a litros/minuto
    v = scale_flow(v, FLOW_CONST, FLOW_OFFSET, FLOW_DIVIDER);
    //Redondeando y actualizando el registro de flujo en tiempo real.
    actual_flow = round_value(v, 10); //10 significa que tiene un decimal
    int_actual_flow = (int)(actual_flow * 10);
  }

  //Sensor ultrasónico conectado a canal 1
  if(adc_ch1_enable) {
    adc_ch1 = ADS.readADC(1);
    float v = (float)(adc_ch1 * f); //Convertir a Voltaje
    v = v * 3.58;
    //Escalamiento de voltaje a centimetros
    v = scale_level(v, LEVEL_CONST, LEVEL_OFFSET, LEVEL_DIVIDER);
    //Redondeando y actualizando el registro de flujo en tiempo real.
    //1, significa que no tiene decimales
    actual_level = round_value(v, 1);
    int_actual_level = (uint16_t)(actual_level * 10);
  }
  //No se utiliza
  if(adc_ch2_enable) {
    adc_ch2 = ADS.readADC(2);
    float v = adc_ch2 * f;
  }
  //No se utiliza
  if(adc_ch3_enable) {
    adc_ch3 = ADS.readADC(3);
    float v = adc_ch3 * f;
  }
  read_adc_value = false;
}
```

En el código mostrado, se observa que los canales 0 y 1 se encuentran activados y convirtiendo la señal del medidor de flujo y sensor ultrasónico respectivamente. Se observa que la lectura sólo se realizará si los canales están habilitados y para el caso de las señales que se están leyendo se realizan acciones de escalamiento y redondeo. La instrucción `int_actual_flow = (int)(actual_flow * 10);` así como

`int_actual_level = (uint16_t)(actual_level * 10);` tienen como finalidad convertir los valores decimales del flujo y nivel a enteros de 16 bits que son los tipos de datos compatibles con el protocolo MODBUS que es el utilizado para comunicarse con el gateway industrial.

3. Controlar las salidas analógicas y de alta velocidad PWM.

El controlador posee dos salidas de alta velocidad PWM y cuatro analógicas. Las salidas analógicas proveen señales desde 0V a 10V a dispositivos actuadores, para el caso del entrenador son utilizadas por la válvula proporcional de flujo. Para generar voltajes analógicos se utiliza el circuito doble filtro paso bajos y un amplificador de voltaje que se muestra en Fig. 23.

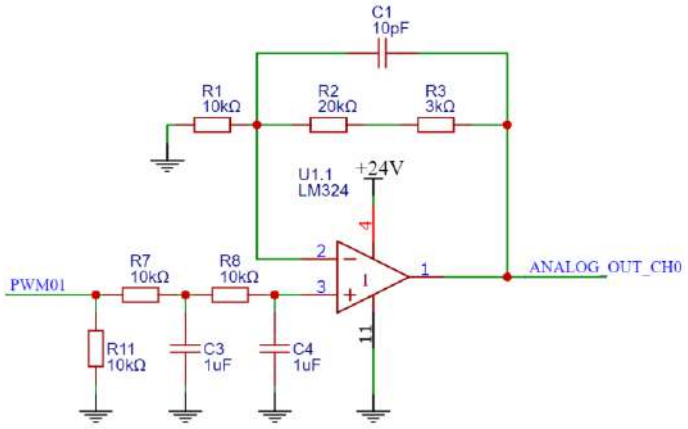

Fig. 23. Circuito convertidor de señales PWM a voltaje de 0 a 10V.

El conjunto de resistencias y capacitores conectados en la entrada no inversora del amplificador de operaciones se utilizan para convertir la señal PWM de 1 KHz generada por el microcontrolador ESP32 a voltajes de 0 a 3.3V, posteriormente el circuito amplificador basado en el LM324 se encarga de elevar los niveles de tensión hasta los 10V, que es compatible con el estándar de la industria. Para aumentar o disminuir el voltaje de salida analógico se varía convenientemente el ciclo de trabajo de la señal PWM, la función `setup_analog_pwm_module` se encarga entre otras cosas de definir la frecuencia de la señal PWM, así como su resolución.

```
void setup_analog_pwm_module(uint16_t f, uint16_t r) {

    //Estabelcer frecuencia y resolución del ciclo de trabajo en bits,
    //ej. 8 bits = 0 - 255
    frequence = f; resolution = r;

    //Asignando frecuencia y resolución a los canales
    ledcSetup(pwmChannel0, frequence, resolution);
    ledcSetup(pwmChannel1, frequence, resolution);
    ledcSetup(pwmChannel2, frequence, resolution);
    ledcSetup(pwmChannel3, frequence, resolution);
    ledcSetup(pwmChannel4, frequence, resolution);
    ledcSetup(pwmChannel5, frequence, resolution);

    //Asociar a un gpio
```

```
    ledcAttachPin(ANA_CH0, pwmChannel0);
    ledcAttachPin(ANA_CH1, pwmChannel1);
    ledcAttachPin(ANA_CH2, pwmChannel2);
    ledcAttachPin(ANA_CH3, pwmChannel3);
    ledcAttachPin(PWM_CH0, pwmChannel4);
    ledcAttachPin(PWM_CH1, pwmChannel5);

    //Poner CT en 0%
    ledcWrite(pwmChannel0, 0);
    ledcWrite(pwmChannel1, 0);
    ledcWrite(pwmChannel2, 0);
    ledcWrite(pwmChannel3, 0);
    ledcWrite(pwmChannel4, 0);
    ledcWrite(pwmChannel5, 0);
}
```

Para habilitar/deshabilitar los canales de salida analógicos y de alta velocidad se utiliza la función `enable_analog_pwm_channel` que tiene la siguiente estructura:

```
void enable_analog_pwm_channel(bool type, uint8_t channel, bool state) {
//Type = true, indica si es una salida analógica, si es false es pwm
    if(type) {  //Es salida analógica
        switch (channel)
        {
        case 0:
            ana_ch0_enable = state;
            if(!ana_ch0_enable) { ledcWrite(pwmChannel0, 0); }
            break;
        case 1:
            ana_ch1_enable = state;
            if(!ana_ch1_enable) { ledcWrite(pwmChannel1, 0); }
            break;
        case 2:
            ana_ch2_enable = state;
            if(!ana_ch2_enable) { ledcWrite(pwmChannel2, 0); }
            break;
        case 3:
            ana_ch3_enable = state;
            if(!ana_ch3_enable) { ledcWrite(pwmChannel3, 0); }
            break;
        default:
            break;
        }
    }
    else {  //Es salida pwm
        switch (channel)
        {
        case 0:
            pwm_ch0_enable = state;
            if(!pwm_ch0_enable) { ledcWrite(pwmChannel4, 0); }
            break;
        case 1:
            pwm_ch1_enable = state;
            if(!pwm_ch1_enable) { ledcWrite(pwmChannel5, 0); }
            break;
        default:
            break;
        }
    }
}
```

Las salidas de alta velocidad PWM cuentan con la función `set_dc_pwm` que se encarga de establecer el ciclo de trabajo. Finalmente, el ciclo de trabajo de la señal PWM de la salida analógica conectada a la válvula proporcional es manejada por el controlador PID.

4. Mantener comunicación con el Gateway Industrial VBOX.

Como el Dashboard de control del entrenador se encuentra en VNET, y esta a su vez se conecta al gateway industrial VBOX se hace necesario que el controlador ESP32 se comunique con este, utilizando algún protocolo industrial. Para este proyecto se utilizó MODBUS TCP ya que se dispone de la librería modbus-esp8266 creada por Andres Sarmento y que es de código y uso abierto [14].

MODBUS TCP funciona por sobre TCP/IP con registros de intercambio que para el caso de este proyecto son tres:

- Registros bobina (COILS), son registros de lectura escritura de 1 bit, se utilizan para establecer estado verdadero (encendido) y falso (apagado). Que se utilizaron para activar o desactivar acciones, actuadores todo o nada, etc.
- Registros variables (IREG), son registros que pueden almacenar valores enteros de 16 bits (0 a 65535) y son útiles para enviar/recibir datos de variables cuyos valores cambian en función del tiempo como flujo y nivel.
- Registros de retención (HREG), son registros que almacenan valores enteros pero que cuyo contenido no varía demasiado durante la ejecución del programa. Fueron utilizados para el almacenamiento de los valores de las variables que controlan el proceso.

Los registros poseen una dirección de acceso que debe configurarse y coincidir tanto en VNET como en el controlador. A continuación, se muestran los registros definidos en el controlador, su tipo y dirección de acceso:

```
//Registros IREG
const uint16_t      SP_M        =   200;    //SP memoria
const uint16_t      PV          =   210;    //Real Time Level
const uint16_t      P_M         =   220;    //Proportional Mem
const uint16_t      I_M         =   230;    //Integral Mem
const uint16_t      D_M         =   240;    //Derivative Mem
const uint16_t      FLOW        =   250;    //Flujo
const uint16_t      PUMP        =   260;    //Water pump power on/off
const uint16_t      SV1_M       =   270;    //Solenoid Valve 1 Status
const uint16_t      SV2_M       =   280;    //Solenoid Valve 2 Status
const uint16_t      UP_LS       =   290;    //Up Water Sensor
const uint16_t      DN_LS       =   300;    //Down Water level Sensor
const uint16_t      START_M     =   320;    //Plant Start Status
const uint16_t      STOP_M      =   330;    //Plant Stop Status
const uint16_t      RESET_M     =   340;    //Plant Reset Status
const uint16_t      PID_O       =   500;    //Porcentaje de salida del
                                            //controlador PID
const uint16_t      ESTADO      =   510;    //Estado del proceso
//Bobinas
const uint16_t      EMERGENCY   =   350;    //Plant Emergency flag
const uint16_t      SV1         =   360;    //Solenoid Valve 1 ON/OFF
const uint16_t      SV2         =   370;    //Solenoid Valve 2 ON/OFF
const uint16_t      START       =   400;    //Plant Start Command
const uint16_t      STOP        =   410;    //Plant Stop Command
const uint16_t      RESET       =   420;    //Plant Reset Command
```

```
//Registros HREG
const uint16_t      SP          =   430;    //Desired Water level in cm
const uint16_t      P           =   440;    //PID Proportional gain
const uint16_t      I           =   450;    //PID Integral gain
const uint16_t      D           =   460;    //PID Derivative gain
```

Los registros mostrados que finalizan con la letra M (ejemplo SP_M) son registros de memoria del controlador y representan el valor actual de ese registro, son utilizados como retroalimentación para el Dashboard en VNET. De esta manera, el usuario puede estar seguro de que el controlador posee el valor deseado.

5. Realizar cálculos del controlador PID.

El controlador PID se implementó con una librearía de Arduino creada por Brett Beauregard y distribuida bajo licencia de MIT [15] que implementa (1).

$$Output = K_p e(t) + K_I \int e(t)dt + K_d \frac{d}{dt} e(t) \qquad (1)$$

De acuerdo con (1), para realizar control, el valor de K_p debe ser de un valor superior a cero, pero tanto K_I como K_d sí pueden ser cero. Con esto, se logra un control proporcional siempre y cuando el Set Point sea un valor superior a cero, pero dentro de los límites del tanque de agua en donde se realiza la acción de control, cuyo nivel no debe exceder los 250 mm.

La inicialización del objeto se realiza con las siguientes instrucciones:

```
//Variables para PID
bool compute_pid = false;
double Setpoint, Setpoint_new, Setpoint_old, Input, Output;
//Especificar parámetros iniciales del controlador
double Kp=0, Kp_new = 0, Kp_old = 0, Ki=0, Ki_new = 0, Ki_old = 0, Kd=0,
Kd_new = 0, Kd_old = 0;
//Objeto PID
PID objPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
//Establecer el modo de realización de los cálculos
objPID.SetMode(AUTOMATIC);
//Establecer límites PID
objPID.SetOutputLimits(0, 100);
```

En el código mostrado, inicialmente, la bandera `compute_pid` es false, por lo que el controlador estará desactivado. También se establece el límite de la salida que es 100, esto significa que sus valores estarán comprendidos entre 0 y 100, lo que correspondería al grado de apertura de la válvula proporcional.
La constante `PID_INTERVAL` posee un valor por defecto de 10, lo cual significa que los cálculos se realizan cada 1 segundo. Esto es verificado cada 100 ms en la siguiente función dentro de `loop()`.

```
//Calcular PID
  if(compute_pid) {
    c_pid++;
    if(c_pid == PID_INTERVAL) {
      //Calcular el valor de salida
      Input = actual_level;
      objPID.SetTunings(Kp, Ki, Kd);
      objPID.Compute();
      c_pid = 0;
    }
    compute_pid = false;
  }
```

Si la bandera `compute_pid` es true, entonces significa que el estado del proceso es "ejecutando" de manera que el controlador debe empezar a realizar los cálculos a partir de los valores de las variables K_p, K_I, K_d y SP (variable `Input` en el código mostrado). SP se actualiza cada un segundo con el nivel actual en el tanque en donde se está realizando la acción de control.

6. Manejar los estados de máquina del proceso.

En Fig. 24 se muestran las variables del proceso del entrenador de flujo y nivel.

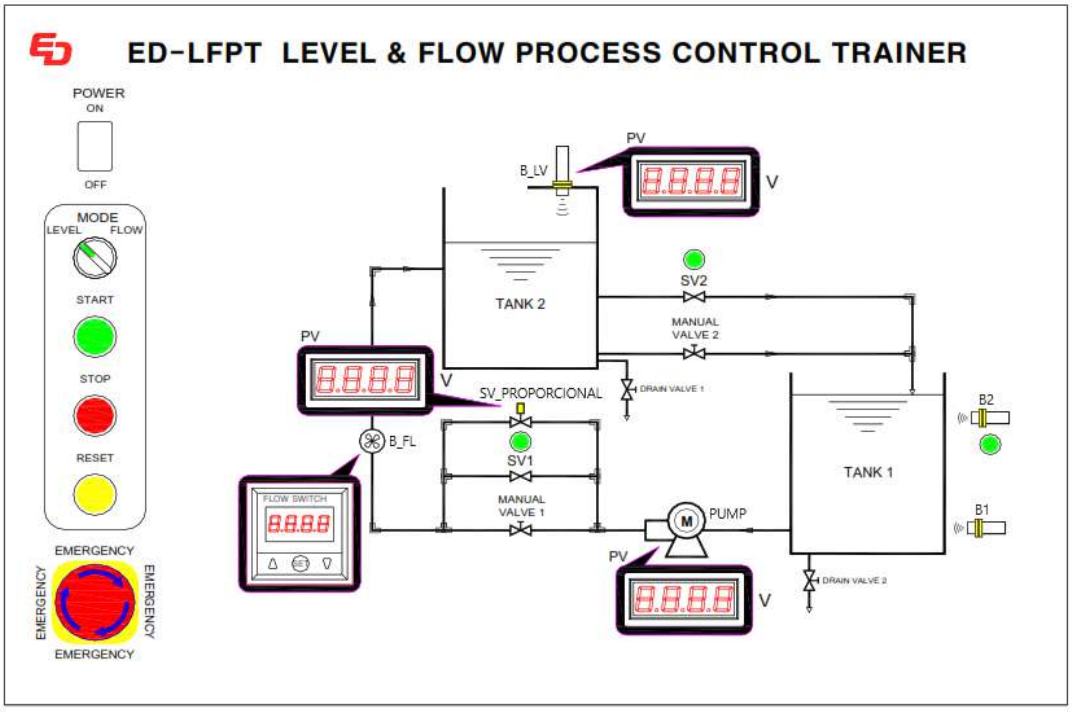

Fig. 24. Variables del proceso.

A partir de las variables y del análisis del proceso se creó el diagrama secuencial que se observa en Fig. 25:

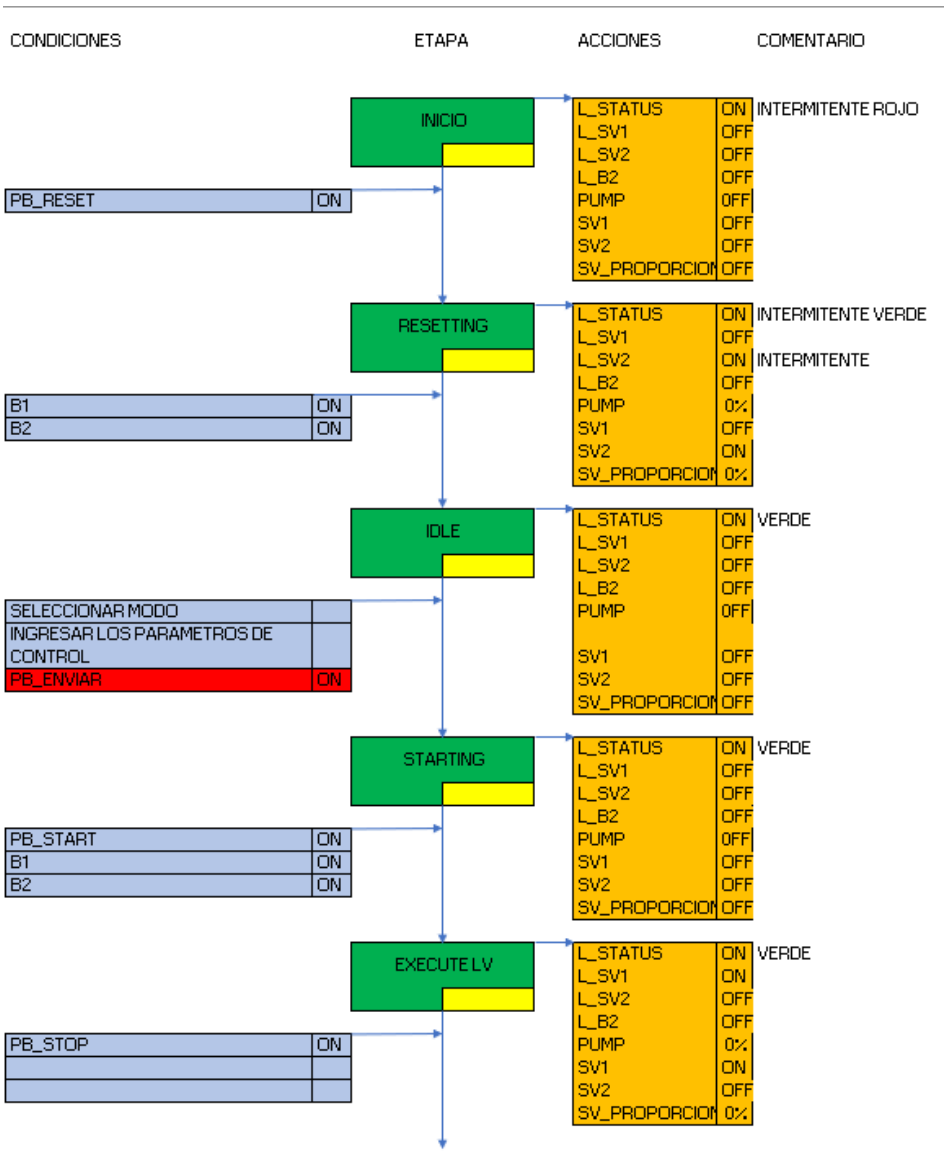

Fig. 25. Diagrama secuencial que muestra los estados del proceso.

Los estados que se muestran en Fig. 25 son los siguientes:

- **INICIO** (paro), es el estado en el cual entra la máquina (entrenador) cuando es energizada por primera vez o cuando se activa el interruptor de paro (STOP). En esta condición, el indicador luminoso de estado hace blink de forma continua. Se puede pasar a estado de inicio en cualquiera de los otros cuatro estados de máquina.
- **REINICIANDO**, si la máquina se encuentra en estado de INICIO se puede activar el estado REINICIANDO. En este estado, se descarga el líquido del tanque 2 y se detiene (se pasa al siguiente estado) hasta que tanto los sensores capacitivos B1 (nivel inferior) y B2 (nivel superior) se encuentran activados simultáneamente. Para descargar el líquido del tanque 2, se abre la electroválvula SV2 lo cual es indicado por la intermitencia (blinks) del indicador de estado en el tablero de principal (Fig. 23). En el estado reiniciando, el indicador luminoso de la máquina hace dos blinks.

- **ESPERA**, después del proceso de reinicio la máquina espera a que el operario establezca los parámetros mínimos de control que son el establecimiento del nivel deseado (SP) y el control proporcional (P). Mientras esta condición no se cumpla solamente se puede pasar a estado de inicio. El indicador de estado hace tres blinks.
- **INICIANDO**, es un estado temporal con transición automática al siguiente y su finalidad es indicar al operario que la máquina ya se encuentra lista para entrar en estado de ejecución o marcha. El indicador de estado hace cuatro blinks.
- **EJECUTANDO** (marcha), en este estado se ejecuta el proceso industrial tratando de hacer llegar el nivel del líquido en el tanque 2 al nivel establecido en el SP. El controlador PID funciona y su salida de control se pasa a la salida analógica correspondiente para controlar el grado de apertura de la válvula proporcional de flujo. El operario puede actualizar en cualquier momento los parámetros de control (variables SP, P, I, D) para poner a prueba el sistema. El indicador de estado se encuentra encendido.

Para cada uno de los estados de la máquina se crearon funciones que aplican las condiciones indicadas en Fig. 25, en el anexo 1, se muestra la estructura de estas funciones.

6.3. Diseño de la Interfaz Gráfica (Dashboard) de Control del Proceso

El diseño del Dashboard se hizo utilizando las herramientas que proporciona la plataforma VNET que es donde el gateway VBOX intercambia los datos del proceso.

Inicialmente, se crearon todas las etiquetas (tags) en VNET que corresponden a los registros MODBUS que se crearon en el controlador.

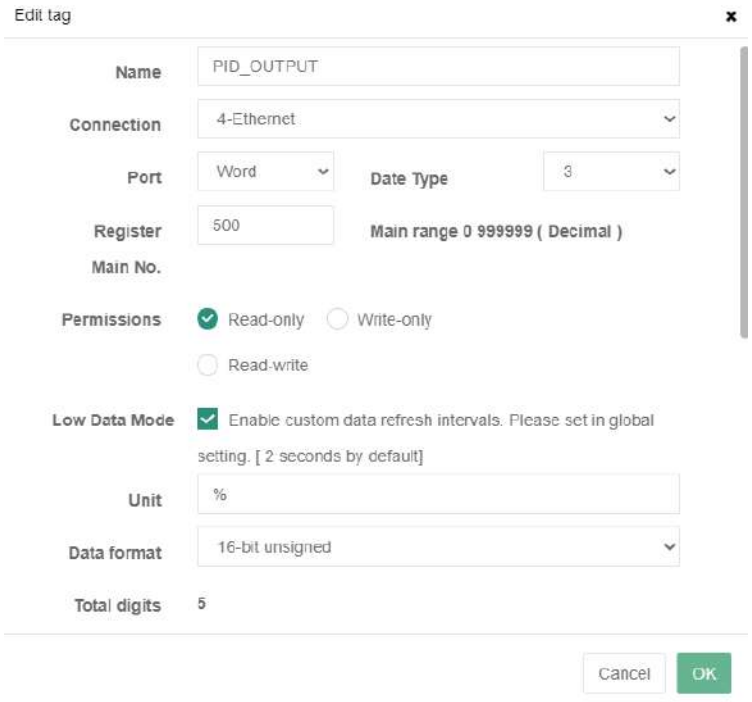

Fig. 26. Creación de un tag MODBUS en VNET.

A continuación, se verificó que todos los tags tuviesen conectividad activa desde VNET hacia el controlador electrónico.

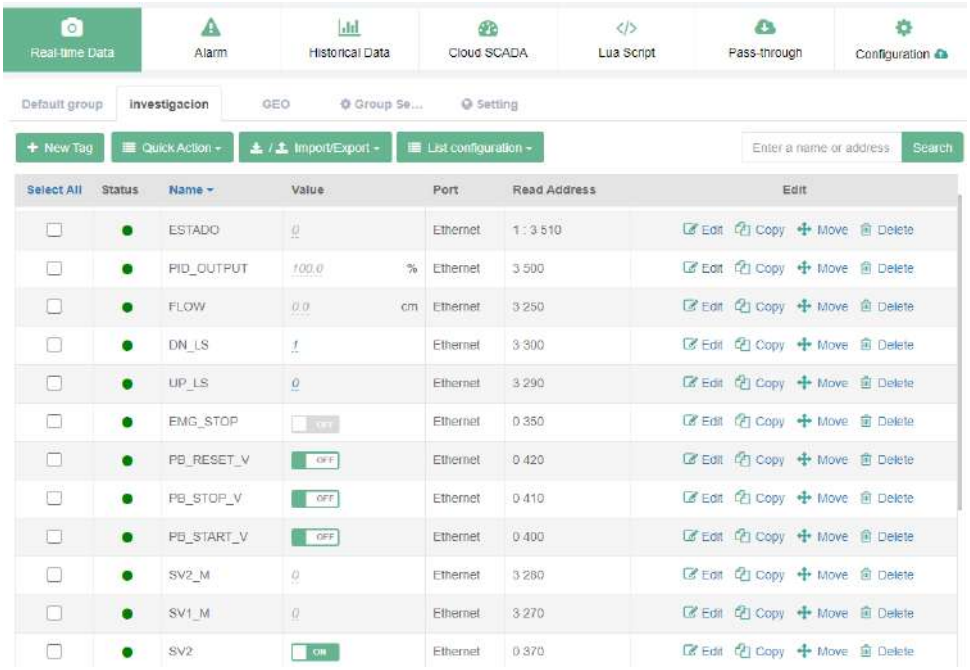

Fig. 27. Lista de tags MODBUS con conectividad al controlador electrónico desde VNET.

Posteriormente, se diseñó un Dashboard de pruebas (Fig. 28) cuyo objetivo fue comprar el funcionamiento de cada uno de los módulos del firmware del controlador a medida que estos fueron incorporándose.

Fig. 28. Dashboard de pruebas del proceso en VNET.

El uso del Dashboard de pruebas fue importante ya que, por medio de su implementación se detectó que, por ejemplo, era necesario crear variables de memoria que retroalimenten al usuario remoto sobre los datos almacenados en las variables. Esto permite que el usuario esté seguro de que el controlador ha

recibido los parámetros que se han enviado desde VNET. Finalmente, se elaboró el Dashboard del control del proceso que se muestra en Fig. 29, el cual incorpora todos los elementos de control correspondientes.

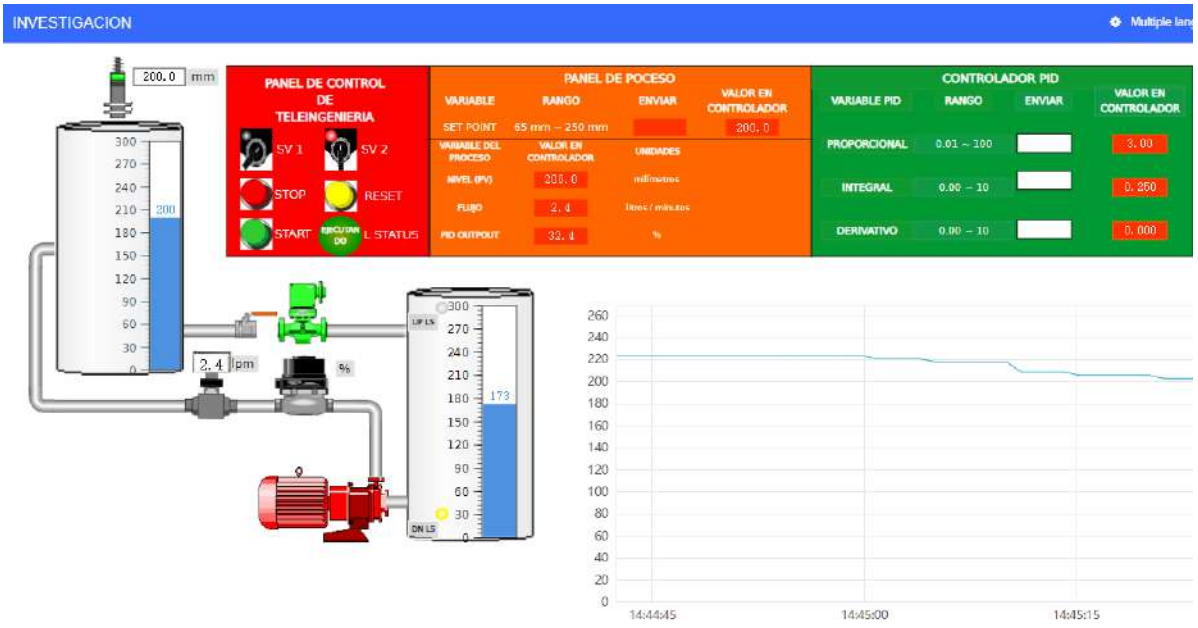

Fig. 29. Dashboard del control del proceso industrial.

6.4. Montaje del Entrenador

Una vez finalizado el proceso de creación del firmware del controlador electrónico, se procedió a realizar el montaje final.

Primeramente, se diseñó un base compatible con riel DIN para montar la tarjeta electrónica del controlador y se montó en el tablero de control y potencia del entrenador.

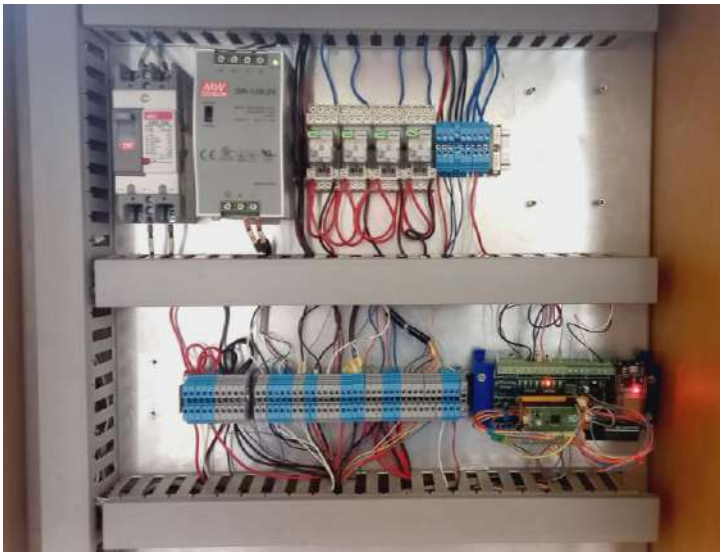

Fig. 30. Montaje final del panel de control y potencia del entrenador.

Posteriormente se utilizaron perfiles metálicos de estantería para crear una estructura en la cual montar las cámaras IP y los dispositivos de la red de datos como se muestra en Fig. 31.

Fig. 31. Estructura para montaje de cámaras y dispositivos de red de datos del entrenador.

Finalmente, se montaron los dispositivos de la red de datos, así como la instalación eléctrica correspondiente respectiva.

Fig. 32. Montaje de dispositivos de red de datos e instalación eléctrica.

6.5. PROGRAMACIÓN DE MÓDULOS DE ENTRENAMIENTO Y SIMULACIÓN CON COMPONENTES 3D Y RA

El diseño de la aplicación se estructuró para ofrecer una interfaz que facilite la interacción con las distintas opciones disponibles. Se han implementado componentes específicos para el visor de la plataforma de teleingeniería, permitiendo la visualización y manipulación de datos relevantes, realidad mixta y realidad aumentada.

Fig. 33. Interfaz de Inicio de aplicación.

Fig. 34. Interfaz de menú opciones de aplicación.

Se programó el módulo remoto donde se encuentra el visor a la plataforma de Teleingeniería.

Fig. 35. Interfaz visor de plataforma de Teleingeniería.

Para desarrollar los módulos se desarrolló el script datos que manipula todo el modelo 3D.

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

public class datos : MonoBehaviour
{
    public Transform agua1;
    public GameObject recipiente1;
    public Transform agua2;
    public GameObject recipiente2;

    public static double h;//Nivel deseado (setpoint) SP
    public static double nivelActual = 0.35;// nivel actual PV
    public static double t = 0; // Tiempo actual
    public static double F1 = 0.000082126; // F1: Flujo de entrada
    public float velocidadLlenado = 0.01f; // Ajusta la velocidad de llenado
    public static double error;
    public static double error_acum = 0;
    public static double error_ant = 0;
    public static double tiempo_ant = 0;
    public static double nAnterior = 0;
    public static double incremento = 0;
    public static double error_der = 0;
    public static List<double> puntos = new List<double>();
    public static double time = 0;
```

```
public TMP_InputField sp;
    public TMP_InputField pv;
    public TMP_InputField kp;
    public TMP_InputField ki;
    public TMP_InputField kd;
    public TMP_InputField Result;

    public GameObject botonCongelar;
    public GameObject botonReiniciar;

    public AudioSource audioAgua1;

    void PausarJuego()
    {
        if (Time.timeScale == 0)
        {
            // Si ya está pausado, reanudar el juego
            Time.timeScale = 1;
        }
        else
        {
            // Si no está pausado, pausar el juego
            Time.timeScale = 0;
        }
    }

    void ReiniciarEscena()
    {

        // Reinicia las variables importantes a sus valores iniciales
        h = 0; // Nivel deseado (setpoint) SP
        nivelActual = 0.35; // Nivel actual PV
        t = 0; // Tiempo actual
        F1 = 0.000082126; // Flujo de entrada
        error = 0;
        error_acum = 0;
        error_ant = 0;
        tiempo_ant = 0;
        nAnterior = 0;
        incremento = 0;
        error_der = 0;
        puntos.Clear();
        time = 0;
// Restablece las escalas y posiciones de los objetos en la escena (ajusta según tu
escena)
        agua1.localScale = new Vector3(1, 0.35f, 1);
        agua2.localScale = new Vector3(1, 2.7f, 1);
        agua1.position = new Vector3(agua1.position.x, agua1.position.y,
agua1.position.z); // ajusta según tu escena
        agua2.position = new Vector3(agua2.position.x, agua2.position.y,
agua2.position.z); // ajusta según tu escena

        // Restablece los valores de los InputFields
        sp.text = "0";
        pv.text = "0.35";
        kp.text = "0";
        ki.text = "0";
        kd.text = "0";
        Result.text = "";
```

```
// Reactiva el proceso
        procesoIniciado = false;

        // Obtiene el índice de la escena actual y la recarga
        int escenaActual = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(escenaActual);
    }

    static double formatNumber(double n)
    {
        return double.Parse(n.ToString("N12"));
    }

    static double int_euler(double h, double dh)
    {
        return h + dh;
    }

    static double obtenerNivelActual(double h, double F1)
    {
        // Rango de integración
        double xi = 0;
        double xf = 240;
        int n = (int)((xf - xi) / 0.001) + 1;
        double[] X = new double[n];
        for (int i = 0; i < n; i++)
        {
            X[i] = xi + i * 0.001;
        }

        // Parámetros del tanque
        double d = 0.012; // Diámetro del tubo de salida del agua
        double g = 9.81; // Gravedad (m/s ^2)
        double f = 8;    // Factor de corrección por resistencias de tubería,
válvula y accesorios
        double b = 0.185; // Base del tanque en metros
        double p = 0.25;  // Profundidad del tanque en metros
        double hs = 0.035; // Altura de la salida del tanque respecto a la base

        // Calcular h
        double A = b * p;
        double At = (Mathf.PI / 4) * Mathf.Pow((float)d, 2f);
        double dh = (1.0 / A) * (F1 - (At * Mathf.Sqrt((float)(2.0 * g * (h - hs))
/ (float)f)));

        h = int_euler(h, dh);

        return formatNumber(h);

}

    IEnumerator InvokePIDWithDelay()
    {
        yield return new WaitForSeconds(2); // Espera 2 segundos
        PID(); // Llama a la función PID después del retraso de 2 segundos
    }

    bool procesoIniciado = false;
```

```
public void IniciarProceso()
    {
        if (!procesoIniciado)
        {
            h = double.Parse(sp.text);
            StartCoroutine(EsperarInicioPID());
        }
    }
    IEnumerator EsperarInicioPID()
    {
        yield return new WaitForSeconds(2);  // Espera 2 segundos
        InvokeRepeating("PID", 0f, 2f);
        procesoIniciado = true;
    }
      void PID()
    {
        h = double.Parse(sp.text);
        nAnterior = formatNumber(nivelActual);
        time = t;
        error = h - nivelActual; // Calcular el error actual
        error_acum += error * (t - tiempo_ant); // Calcular la acumulación del
error a lo largo del tiempo
        error_der = (error - error_ant) / (t - tiempo_ant + 0.0001); // Calcular la
tasa de cambio del error + 0.0001 (tolerancia)
        double u = double.Parse(kp.text) * error + double.Parse(ki.text) *
error_acum + double.Parse(kd.text) * error_der; // Calcular la señal de control
        F1 = u * 0.0135384;
        // Actualizar las variables del controlador
        nivelActual = obtenerNivelActual(h, F1); // Obtener el nivel actual del
tanque
        incremento = Mathf.Abs((float)((nAnterior - nivelActual) / 10.0));
        const double umbralTolerancia = 0.00001;//0.00001
        bool flag = true;
        while (Mathf.Abs((float)(nAnterior - nivelActual)) >=
(float)umbralTolerancia)
        {//inicio del while principal

if (nAnterior != nivelActual && time > 0)
            {
                if (nAnterior < nivelActual)
                {
                    nAnterior += incremento;
                    // Console.WriteLine("Cambiar posición en Y0: " + nivelActual);
                    Result.text = nivelActual.ToString();
                }
                else
                {
                    // Console.WriteLine("Cambiar posición en Y1: " + nivelActual);
                    nAnterior -= incremento;
                    Result.text = nivelActual.ToString();
                }
                flag = false;
            }
            else
            {
                flag = false;
            }
            time += 0.1; // Incrementa 100 ms
            puntos.Add(nAnterior);
        }//final while
```

```
//Console.WriteLine("Altura a utilizar Y2: " + nivelActual);
        Result.text = nivelActual.ToString();
        if (flag)
        {
            time += 0.1; // Incrementa 100 ms
        }
        error_ant = error;
        tiempo_ant = t;
        t++;
    }

    void Start()
    {

botonReiniciar.GetComponent<UnityEngine.UI.Button>().onClick.AddListener(ReiniciarE
scenaDesdeBoton);
        // Agrega un listener al botón para manejar el clic

botonCongelar.GetComponent<UnityEngine.UI.Button>().onClick.AddListener(PausarJuego
);
        DesactivarInputField(pv);
        pv.text = "0.35";
        InvokeRepeating("PID", 0f, 2f);
    }

        void DesactivarInputField(TMP_InputField inputField)
    {
        inputField.interactable = false;  // Desactivar la capacidad de edición
        inputField.textComponent.color = Color.gray;  // Cambiar el color del texto
para indicar que está desactivado (opcional)
        inputField.selectionColor = Color.clear;  // Eliminar el color de selección
(opcional)
        inputField.navigation = new Navigation { mode = Navigation.Mode.None };  //
Desactivar la navegación (opcional)
    }

    public float escaladoSuavizado = 0.03f; // Velocidad de transición de escalas

    void ActualizarEscalas()
    {
        // Escala mínima y máxima para limitar los valores de la escala
        float escalaMinima = 0.35f;
        float escalaMaxima = 3.05f;

        // Asegurarse de que el nivel actual esté dentro del rango deseado
        nivelActual = Mathf.Clamp((float)nivelActual, escalaMinima, escalaMaxima);

        // Obtener la escala actual de los cubos
        Vector3 escalaActualAgua1 = agua1.transform.localScale;
        Vector3 escalaActualAgua2 = agua2.transform.localScale;

        float  nivelActualFloat  =  Mathf.Clamp((float)nivelActual,  escalaMinima,
escalaMaxima);

        float escalaAgua1Y = nivelActualFloat;
        float escalaAgua2Y = escalaMaxima - escalaAgua1Y;

// Ajustar a los límites
        escalaAgua1Y = Mathf.Clamp(escalaAgua1Y, escalaMinima, escalaMaxima);
        escalaAgua2Y = Mathf.Clamp(escalaAgua2Y, escalaMinima, escalaMaxima);
```

```
// Calcular nuevas escalas
        Vector3 nuevaEscalaAgua1 = new Vector3(escalaActualAgua1.x, escalaAgua1Y,
escalaActualAgua1.z);
        Vector3 nuevaEscalaAgua2 = new Vector3(escalaActualAgua2.x, escalaAgua2Y,
escalaActualAgua2.z);
        // Aplicar transición suave en la escala usando interpolación lineal
        agua1.transform.localScale = Vector3.Lerp(agua1.transform.localScale,
nuevaEscalaAgua1, Time.deltaTime * escaladoSuavizado);
        agua2.transform.localScale = Vector3.Lerp(agua2.transform.localScale,
nuevaEscalaAgua2, Time.deltaTime * escaladoSuavizado);
        // Verificar si la escala de agua1 ha alcanzado su destino antes de ajustar
la posición de agua2
        if (Mathf.Abs(escalaAgua1Y - escalaActualAgua1.y) > 0.001f)
        {
            float bordeInferiorAgua1 = agua1.transform.position.y -
(escalaActualAgua1.y / 2.0f);
            float bordeInferiorAgua2 = agua2.transform.position.y -
(escalaActualAgua2.y / 2.0f);
            // Ajustar la posición vertical en relación al borde inferior del cubo
para agua1
            Vector3 nuevaPosicionAgua1 = new Vector3(agua1.transform.position.x,
bordeInferiorAgua1 + escalaAgua1Y / 2.0f, agua1.transform.position.z);
            agua1.transform.position = Vector3.Lerp(agua1.transform.position,
nuevaPosicionAgua1, Time.deltaTime * escaladoSuavizado);
            // Ajustar la posición vertical en relación al borde inferior del cubo
para agua2
            Vector3 nuevaPosicionAgua2 = new Vector3(agua2.transform.position.x,
bordeInferiorAgua2 + escalaAgua2Y / 2.0f, agua2.transform.position.z);
            agua2.transform.position = Vector3.Lerp(agua2.transform.position,
nuevaPosicionAgua2, Time.deltaTime * escaladoSuavizado);
        }
    }

    void ReiniciarEscenaDesdeBoton()
    {
        // Cancela la invocación de ActualizarEscalas para evitar que se active
después del reinicio
        CancelInvoke("ActualizarEscalas");
        CancelInvoke("PID");
        procesoIniciado = false;

        agua1.localScale = new Vector3(1, 0.35f, 1);
        agua2.localScale = new Vector3(1, 2.7f, 1);
        agua1.position = new Vector3(agua1.position.x, agua1.position.y,
agua1.position.z); // ajusta según tu escena
        agua2.position = new Vector3(agua2.position.x, agua2.position.y,
agua2.position.z); // ajusta según tu escena
                // Obtiene el índice de la escena actual y la recarga
        int escenaActual = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(escenaActual);

        // Reactiva la actualización después de restablecer las escalas y
posiciones
        InvokeRepeating("ActualizarEscalas", 0f, 2f);
      }
    void Update()
    {
        ActualizarEscalas();
    }
}
```

Se incorporó soporte para tecnologías de realidad mixta, lo que implica la integración de elementos virtuales en el entorno real del usuario. Esto se logra mediante el uso de dispositivos que combinan información del mundo físico con objetos y datos generados digitalmente. Se usó como base el script llamado datos.

Fig. 36. Interfaz de realidad mixta en la aplicación.

Asimismo, se ha incluido funcionalidad de realidad aumentada, que enriquece la experiencia del usuario al superponer información adicional o elementos virtuales en el mundo real a través de la cámara del dispositivo.

Fig. 37. Interfaz de realidad aumentada en la aplicación.

6.6. Programación de Dashboard de Análisis de Datos Con Bussiness Intelligence

Se implementó un Dashboard para el análisis de datos y se desarrolló el código:

```
<template>
  <!-- 'reservas_diarias'=> $reservas[0]->num_reservas,
       'tiempo_usado'=> $tiempo[0]->tiempo,
       'veces_usado'=> [
          'total_FPC' => $n->total_FPC,
          'total_FN' => $n->total_FN
       ],
       'jornadas' => [
          'diurna'=> $jornadas->diurna,
          'nocturna'=> $jornadas->nocturna
       ] -->
  <div class="fluid">
    <div class="row">
      <div class="col-4">
        <div class="btn btn-block btn-dark btn-sm p-3">
          <h3><i class="bi bi-calendar-check"></i></h3>
          <h1>{{reservas}}</h1>
          <p class="text-center fw-lighter description-home">
            N° de prácticas reservadas este día
          </p>
        </div>
      </div>
      <div class="col-4">
        <div class="btn btn-block btn-info btn-sm p-3">
          <h3><i class="bi bi-alarm"></i></h3>
          <h1>{{tiempo}}</h1>
          <p class="text-center fw-lighter description-home">
            N° de horas de práctica reservadas este día
          </p>
        </div>
      </div>
<div class="col-4">
        <div class="btn btn-block btn-info btn-sm p-3">
          <h3><i class="bi bi-alarm"></i></h3>
          <h1>{{tiempo}}</h1>
          <p class="text-center fw-lighter description-home">
            N° de horas de práctica reservadas este día
          </p>
        </div>
      </div>
```

```
<div class="col-4">
          <div class="btn btn-block btn-dark btn-sm p-3">
            <h3><i class="bi bi-cpu"></i></h3>
            <h1>{{fpc}}</h1>
            <p class="text-center fw-lighter description-home">
              N° de reservas para el entrenador FPC este día
            </p>
          </div>
        </div>
        <div class="col-4">
          <div class="btn btn-block btn-info btn-sm p-3">
            <h3><i class="bi bi-water"></i></h3>
            <h1>{{fn}}</h1>
            <p class="text-center fw-lighter description-home">
              N° de reservas para el entrenador de Flujo y Nivel este día
            </p>
          </div>
        </div>
        <div class="col-4">
          <div class="btn btn-block btn-dark btn-sm p-3">
            <h3><i class="bi bi-moon-stars"></i></h3>
            <h1>{{nocturna}}</h1>
            <p class="text-center fw-lighter description-home">
              N° de prácticas reservadas en jornada nocturna
            </p>
          </div>
        </div>
        <div class="col-4">
          <div class="btn btn-block btn-info btn-sm p-3">
            <h3><i class="bi bi-cloud-sun"></i></h3>
            <h1>{{diurna}}</h1>
            <p class="text-center fw-lighter description-home">
              N° de prácticas reservadas en jornada diurna
            </p>
          </div>
        </div>
</div>
  </div>
</template>
```

```
<script>
export default {
   data() {
      return {
         reservas : '0',
         tiempo : '0',
         fpc : '0',
         fn : '0',
         diurna : '0',
         nocturna : '0'
      };
   },   mounted () {
      this.makeDashboard()
   },   methods: {
      async makeDashboard() {
         // Reservas diarias
         let res = await axios.get("/api/reservas");
         let info = res.data
         this.reservas = info.reservas_diarias
         this.tiempo = info.tiempo_usado
         this.fpc = info.veces_usado.total_FPC
         this.fn = info.veces_usado.total_FN
         this.diurna = info.jornadas.diurna
         this.nocturna = info.jornadas.nocturna
      }
   }
}
</script>
```

Fig. 38. Interfaz de Dashboard de análisis de datos.

7. RESULTADOS

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

- Un circuito controlador electrónico de propósito general que permite la gestión a distancia de un entrenador FPC de flujo y nivel.
- Un panel de control Web que permite monitorear y realizar ajustes a distancia de un control de procesos industriales de flujo y nivel.
- Un manual de prácticas para el laboratorio de Teleingeniería de control de Flujo y Nivel.
- Un manual de instalación y uso de laboratorio de Teleingeniería de control de Flujo y Nivel.

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

- Un Simulador virtual con Realidad Aumentada.
- Un Manual de Usuario.
- Un Manual de Instalación.

8. CONCLUSIONES

- Es posible diseñar un simulador con realidad aumentada de un proceso industrial de flujo y nivel utilizando herramientas de cálculo diferencial y obtener resultados similares a los que se producen en los dispositivos reales.
- Se puede diseñar un controlador electrónico a medida de los requerimientos cumpliendo requisitos de control, de señales de entrada-salida y demás características de los controladores industriales existentes en el mercado.
- Se pueden monitorear y manipular las variables de proceso de forma remota utilizando el protocolo de comunicación industrial MODBUS TCP implementándolo en un controlador electrónico utilizando librerías de código abierto.
- La plataforma con ejecución basada en la nube VNET facilita el trabajo de diseño de los Dashboard de control de procesos, permitiendo a los ingenieros concentrarse en los mecanismos de comunicación y actualización de los datos.
- La realidad aumentada ofrece una experiencia de entrenamiento inmersiva y visualmente rica. La simulación de un controlador de flujo y nivel a través de esta tecnología proporciona a los usuarios una comprensión más práctica y profunda de los conceptos teóricos.
- Es posible interactuar directamente con el entorno virtual, ajustar parámetros y observar las respuestas en tiempo real, lo que fomenta la participación. Esto puede aumentar la retención del conocimiento y mejorar las habilidades de resolución de problemas.
- La realidad aumentada y mixta facilita el aprendizaje autónomo al permitir que los usuarios practiquen y adquieran habilidades en su propio tiempo y ritmo. Además, puede ser accesible en ubicaciones remotas, lo que es beneficioso para la formación descentralizada.

- Al simular un controlador de flujo y nivel mediante realidad aumentada, se pueden reducir los costos asociados con la formación en equipos físicos. Además, se optimiza el tiempo de formación al proporcionar una experiencia de aprendizaje más eficiente y centrada en resultados.

9. RECOMENDACIONES

- Aplicar otra técnica para eliminar el uso de marcadores en realidad mixta y realidad aumentada.
- Utilizar cámaras de video que proporcionen herramientas para enfocar de forma más efectiva el video y mejorar la experiencia del usuario.

10. GLOSARIO

A

Adquisición de datos

Son los productos y/o procesos utilizados para recopilar información para documentar o analizar un fenómeno.

B

Bases de datos de tipo relacional

Es un tipo de base de datos que cumple con el modelo relacional, que es el modelo más utilizado actualmente para implementar las Bases de Datos ya planificadas.

Big Data

Es un concepto que hace referencia al almacenamiento de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos.

Big Table

Es un sistema de gestión de base de datos creado por Google con las características de ser: distribuido, de alta eficiencia y propietario.

Business Intelligence

Es un término genérico que incluye las aplicaciones, la infraestructura y las herramientas, y las mejores prácticas que permiten el acceso y el análisis de la información para mejorar y optimizar las decisiones.

M

MongoDB

Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

N

NoSQL

Not Only SQL y son sistemas de almacenamiento que no cumplen con el esquema entidad-relación Proveen un sistema de almacenamiento mucho más flexible y concurrente y permiten manipular grandes cantidades de información de manera mucho más rápida que las bases de datos relacionales.

P

Presión diferencial

La presión diferencial se define como la diferencia de las medidas de la presión entre dos puntos en un sistema. Esta medida es importante en aplicaciones que tienen la funcionalidad de la presión, tales como la instrumentación meteorológica, los aviones y los automóviles.

R

Retroalimentación

Como retroalimentación se designa el método de control de sistemas en el cual los resultados obtenidos de una tarea o actividad son reintroducidos nuevamente en el sistema con el fin de controlar y optimizar su comportamiento.

Robot industrial

Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas.

S

Sistema telemático

Es aquel que está formado por equipos informáticos conectados unos a otros mediante una red de telecomunicaciones, que consiste en una red de nodos ordenados para la comunicación tanto para cortas como para largas distancias.

V

Visor VR

Un casco de realidad virtual, también llamado gafas de realidad virtual, visor de realidad virtual o HMD (del inglés head-mounted display), es un dispositivo de visualización similar a un casco, que permite reproducir imágenes creadas por ordenador sobre una pantalla muy cercana a los ojos o proyectando la imagen directamente sobre la retina de los ojos. En este segundo caso el casco de realidad virtual recibe el nombre de monitor virtual de retina.

11. REFERENCIAS BIBLIOGRÁFICAS

[1] S. Soria, "Capítulo 1. Introducción a los sistemas automáticos industriales", en *Sistemas automáticos industriales de eventos discretos*, Alpha Editorial, 2013.

[2] W. Dunn, *Fundamentals of Industrial Instrumentation and Process Control*, 2a ed. Estados Unidos, 2018.

[3] Pepperl+Fuchs, "Sensores capacitivos", Pepperl+Fuchs. Consultado: el 26 de enero de 2022. [En línea]. Disponible en: https://www.pepperl-fuchs.com/global/es/classid_144.htm

[4] "Medidores de flujo: ¿qué son y cómo funcionan? | Badger Meter". Consultado: el 26 de enero de 2022. [En línea]. Disponible en: https://www.badgermeter.com/es-es/blog-es-es/medidores-de-flujo-que-son-y-como-funcionan/

[5] F. Navarro, A. Martínez, y J. Martínez, *Realidad virtual y Realidad Aumentada. Desarrollo de Aplicaciones*. Grupo Editorial RA-MA. Consultado: el 6 de febrero de 2024. [En línea]. Disponible en: https://www.ra-ma.es/libro/realidad-virtual-y-realidad-aumentada_83635/, https://www.ra-ma.es/libro/realidad-virtual-y-realidad-aumentada_83635/

[6] J. Cortés Trujillo, *Realidad virtual en los procesos de enseñanza en la educación superior*. Corporación Universitaria Minuto de Dios. UNIMINUTO, 2019.

[7] "Realidad virtual, aumentada y mixta. Qué son y diferencias.", Editeca. Consultado: el 11 de febrero de 2022. [En línea]. Disponible en: https://editeca.com/realidad-virtual-aumentada-y-mixta-que-son-y-en-que-se-diferencian/

[8] "Realidad Virtual, ¿qué es y para qué sirve? ▷ 9 Aplicaciones", EDS Robotics. Consultado: el 11 de febrero de 2022. [En línea]. Disponible en: https://www.edsrobotics.com/blog/realidad-virtual-que-es/

[9] Mullen, Tony, *Realidad Aumentada: Crea tus propias Aplicaciones*. España: Anaya, 2011.

[10] "Todo sobre la Realidad Aumentada", Innovae. Consultado: el 6 de febrero de 2024. [En línea]. Disponible en: https://www.innovae.com/la-realidad-aumentada/

[11] Raspberry Pi Foundation, "Raspberry Pi Documentation - RP2040". Consultado: el 12 de septiembre de 2023. [En línea]. Disponible en: https://www.raspberrypi.com/documentation/microcontrollers/rp2040.html

[12] "Raspberry Pi Documentation - Raspberry Pi Pico and Pico W". Consultado: el 22 de febrero de 2024. [En línea]. Disponible en: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[13] R. Tillaart, "RobTillaart/ADS1X15". el 29 de enero de 2024. Consultado: el 22 de febrero de 2024. [En línea]. Disponible en: https://github.com/RobTillaart/ADS1X15

[14] "modbus-esp8266/examples/TCP-ESP at master · emelianov/modbus-esp8266", GitHub. Consultado: el 22 de febrero de 2024. [En línea]. Disponible en: https://github.com/emelianov/modbus-esp8266/tree/master/examples/TCP-ESP

[15] "Improving the Beginner's PID – Introduction « Project Blog". Consultado: el 27 de febrero de 2024. [En línea]. Disponible en: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

12. ANEXO - FUNCIONES DE CONTROL DE LOS ESTADOS DE MÁQUINA DEL PROCESO

```
void estado_PARO() {

  printf("Estado: PARO\n");

  P_BOMB = false;
  P_SV1 = false;
  P_SV1_LIGHT = false;
  P_SV2 = false;
  P_SV2_LIGHT = false;
  P_UPLS_LIGHT = P_UPLS;

  mb.Ireg(SV2_M, 0);
  mb.Ireg(SV1_M, 0);


  if(P_STATUS_LIGHT_R) {
    P_STATUS_LIGHT_R = false;
  }
  else {
    P_STATUS_LIGHT_R = true;
  }

}

bool restart_blink = true;
uint8_t restart_blink_count = 0;

void estado_REINICIANDO() {

  printf("Estado: REINICIANDO\n");

  P_BOMB = false;
  P_SV1 = false;
  P_SV1_LIGHT = false;
  P_SV2 = true;

  mb.Ireg(SV2_M, 1);
  mb.Ireg(SV1_M, 0);

  if(P_SV2_LIGHT) { P_SV2_LIGHT = false; } else { P_SV2_LIGHT = true; }
  P_UPLS_LIGHT = P_UPLS;

  //Blinking two times
  if(restart_blink == true) {
    restart_blink_count++;
    if(restart_blink_count <= 4) {
      if(P_STATUS_LIGHT_R) { P_STATUS_LIGHT_R = false; }
      else {P_STATUS_LIGHT_R = true;}
    }
    else { restart_blink = false; restart_blink_count = 0; }
  }
  else {
    restart_blink_count++;
    if(restart_blink_count <= 4) { P_STATUS_LIGHT_R = false; }
    else { restart_blink = true; restart_blink_count = 0; }
  }


  //Condición de salida
  if(P_DNLS == 1 & P_UPLS == 1) {

    //Salir del estado
```

```
    REINICIANDO = false;
    ESPERA = true; //Pasando a ESPERA
    actual_state = 2;
  }

}



bool waiting_blink = true;
uint8_t waiting_blink_count = 0;

void estado_ESPERA() {

  printf("Estado: ESPERANDO\n");

  P_BOMB = false;
  P_SV1 = false;
  P_SV1_LIGHT = false;
  P_SV2 = false;
  P_SV2_LIGHT = false;
  P_UPLS_LIGHT = P_UPLS;

  mb.Ireg(SV2_M, 0);
  mb.Ireg(SV1_M, 0);

  //Blinking three times
  if(waiting_blink == true) {
    waiting_blink_count++;
    if(waiting_blink_count <= 6) {
      if(P_STATUS_LIGHT_R) { P_STATUS_LIGHT_R = false; }
      else {P_STATUS_LIGHT_R = true;}
    }
    else { waiting_blink = false; waiting_blink_count = 0; }
  }
  else {
    waiting_blink_count++;
    if(waiting_blink_count <= 4) { P_STATUS_LIGHT_R = false; }
    else { waiting_blink = true; waiting_blink_count = 0; }
  }


  //Evaluando para salir
  if(Kp > 0 & Setpoint > 0) { //Solo se pasa al siguiente estado si se han
establecido: SP y Kp
    //Salir del estado y pasar a INICIANDO
    ESPERA = false;
    INICIANDO = true;
    actual_state = 3;  //Iniciando
  }

}


bool starting_blink = true;
uint8_t starting_blink_count = 0;
void estado_INICIANDO() {

  printf("Estado: INICIANDO\n");

  P_BOMB = false;
  P_SV1 = false;
  P_SV1_LIGHT = false;
  P_SV2 = false;
  P_SV2_LIGHT = false;
  P_UPLS_LIGHT = P_UPLS;
```

```
  mb.Ireg(SV1_M, 0);
  mb.Ireg(SV2_M, 0);


  //Blinking four times

  if(starting_blink == true) {
    starting_blink_count++;
    if(starting_blink_count <= 8) {
      if(P_STATUS_LIGHT_R) { P_STATUS_LIGHT_R = false; }
      else {P_STATUS_LIGHT_R = true;}
    }
    else { starting_blink = false; starting_blink_count = 0; }
  }
  else {
    starting_blink_count++;
    if(starting_blink_count <= 4) { P_STATUS_LIGHT_R = false; }
    else { starting_blink = true; starting_blink_count = 0; }
  }

}

void estado_EJECUTANDO() {

  if(first_time_start) {  //Hacer esto solo la primera vez que EJECUTANDO se activa
    P_SV1 = false;
    P_SV1_LIGHT = false;
    P_SV2 = false;
    P_SV2_LIGHT = false;

    first_time_start = false;
  }

  P_STATUS_LIGHT_R = true;
  P_BOMB = true;
  P_UPLS_LIGHT = P_UPLS;

  double t_output = Output * 2.5;

  set_dc_analog_out(0, t_output);

}
```

ITCA FEPADE 55

SEDE CENTRAL Y CENTROS REGIONALES EL SALVADOR

La Escuela Especializada en Ingeniería ITCA-FEPADE, fundada en 1969, es una institución estatal con administración privada, conformada actualmente por 5 campus: Sede Central Santa Tecla y cuatro centros regionales ubicados en Santa Ana, San Miguel, Zacatecoluca y La Unión.

1. SEDE CENTRAL SANTA TECLA

Km. 11.5 carretera a Santa Tecla, La libertad.
Tel.: (503) 2132-7400

2. CENTRO REGIONAL SANTA ANA

Final 10a. Av. Sur, Finca Procavia.
Tel.: (503) 2440-4348

3. CENTRO REGIONAL ZACATECOLUCA

Km. 64.5, desvío Hacienda El Nilo sobre autopista a Zacatecoluca.
Tel.: (503) 2334-0763 y 2334-0768

4. CENTRO REGIONAL SAN MIGUEL

Km. 140 carretera a Santa Rosa de Lima.
Tel.: (503) 2669-2298

5. CENTRO REGIONAL LA UNIÓN

Calle Sta. María, Col. Belén, atrás del Instituto Nacional de La Unión
Tel.: (503) 2668-4700

www.itca.edu.sv