Diseño e implementación de un laboratorio remoto para el aprendizaje de controladores PID, enfocado en la operación de bombas

Alvarez Meythaler, Carlos Francisco y Cisneros Salazar, Hugo Andrés

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Ortiz Tulcán, Hugo Ramiro Mgs

08 de febrero de 2022

Alvarez_Cisneros_TrabajoTitulacion_Copyleaks.pdf

Scanned on: 20:22 February 4, 2022 UTC

Overall Similarity Score

Results Found

Total Words in Text

Identical Words	293
Words with Minor Changes	164
Paraphrased Words	826
Ommited Words	1720

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Diseño e implementación de un laboratorio remoto para el aprendizaje de controladores PID, enfocado en la operación de bombas**" fue realizado por los señores **Alvarez Meythaler, Carlos Francisco y Cisneros Salazar, Hugo Andrés** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 08 de febrero de 2022

Firma:

Firmado electrónicamente por:
HUGO RAMIRO
ORTIZ TULCAN

Ing. Ortiz Tulcán, Hugo Ramiro, Mgs

C.C.: 1707721591

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Nosotros, **Alvarez Meythaler, Carlos Francisco** con cédula de ciudadanía n° 0503353450 y **Cisneros Salazar, Hugo Andrés**, con cedula de ciudadanía n° 1723224133, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Diseño e implementación de un laboratorio remoto para el aprendizaje de controladores PID, enfocado en la operación de bombas** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 08 de febrero de 2022

Firma

.....

Alvarez Meythaler, Carlos Francisco

C.C.: 0503353450

Firma

.....

Cisneros Salazar, Hugo Andrés

C.C.: 1723224133

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Nosotros **Alvarez Meythaler, Carlos Francisco** con cédula n° 0503353450 y **Cisneros Salazar, Hugo Andrés** con cédula n° 1723224133, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un laboratorio remoto para el aprendizaje de controladores PID, enfocado en la operación de bombas** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 08 de febrero de 2022

Firma .....

Alvarez Meythaler, Carlos Francisco

C.C.: 0503353450

Firma .....

Cisneros Salazar, Hugo Andrés

C.C.: 1723224133

Dedicatoria

A mis padres, quienes siempre están apoyándome para que pueda alcanzar mis objetivos académicos y personales, quienes con su amor y sabiduría me guían en cada nueva etapa de mi vida, quienes son mi modelo profesional y moral.

A mis hermanos, quienes son mi ejemplo y motivación a seguir esforzándome cada día más, y quienes me han acompañado durante todo este trayecto hasta alcanzar este título de Ingeniero.

Carlos Francisco Alvarez Meythaler

Con mucho cariño, dedico este trabajo de titulación y la profesión de Ingeniero, a mis padres Hugo y Gina, que han estado conmigo incondicionalmente durante todo mi proceso universitario, me han enseñado a ser perseverante y han sido principales responsables de mis éxitos académicos y personales.

A mi hermana Gaby, que ha sido la fuente de motivación en los momentos más complejos de mi vida. Mis éxitos son también de ella. A mi sobrino Joaquin, para que tome como ejemplo que con perseverancia y responsabilidad se puede conseguir cualquier cosa en la vida.

A mi enamorada Romi, por siempre creer en mí, apoyarme incondicionalmente en todos los momentos de mi vida y recordarme constantemente que soy capaz de conseguir todo lo que me proponga.

Hugo Andrés Cisneros Salazar

Agradecimiento

A Dios por haber protegido a mi familia y a mí durante todo este tiempo, además por darme las fuerzas para superar los obstáculos que aparecieron a lo largo de mi carrera universitaria.

A mis padres por su cariño y apoyo incondicional, por ayudarme siempre que tenía un problema y no era capaz de manejarlo solo.

A mis hermanos por ser mi soporte y ayudarme a no darme por vencido en los momentos difíciles.

A mis abuelos, porque me permitían desconectar por un momento mi cabeza de las actividades académicas y aprender de sus experiencias y conocimientos.

A mis amigos José Luis Segovia, Alejandro Hidalgo, José Villamarín y Martín Burbano, por todos los momentos compartidos desde pequeños, porque siempre me apoyaron con sus consejos y mensajes de ánimo.

A mis compañeros y amigos de la universidad: Hugo Cisneros, Jonathan Acosta y Renny Coro, por los conocimientos brindados y experiencias vividas desde que iniciamos la carrera.

A nuestro tutor el Ing. Hugo Ortiz, por la confianza que tuvo en nosotros para sacar adelante el proyecto, y porque siempre estuvo dispuesto a responder nuestras inquietudes.

Por último, agradezco a todos los docentes, porque gracias a ellos, adquirí los conocimientos y habilidades que me permitieron culminar esta meta importante en mi vida.

Carlos Francisco Alvarez Meythaler

La consecución de esta profesión, fue en gran parte gracias al apoyo constante e incondicional de mi familia. Por esta razón, quiero dar gracias a mis padres Hugo y Gina por enseñarme todos los valores que se necesitan para ser una buena persona, por apoyarme todos los días en todo lo que hago, y sobre todo por amarme de la forma en que lo hacen. Hoy puedo ser profesional gracias a todo lo que ellos han hecho por mí.

Quiero dar las gracias a mi hermana Gaby, porque desde que empecé esta carrera universitaria, siempre estuvo conmigo para darme ánimos cuando más lo necesitaba.

A toda mi familia: tíos/as, abuelitos/as, primos/as, sobrino, porque de alguna u otra manera me han apoyado a lo largo de mi vida universitaria. Un especial agradecimiento a mi primo Alex Padilla, quien fue el responsable de que yo siga esta carrera y que ha estado para mi durante todo este proceso.

A mi enamorada Romi, por soportar mis peores comportamientos debido al estrés de la carrera universitaria, por ser leal e incondicional, por estar siempre a mi lado, pero sobre todo por amarme a pesar de cualquier situación. Todo lo que consiga a partir de ahora será de ella también.

A mis amigos Martín Correa y Eduardo Moreno, quienes han sido leales y fieles en todo momento. A mis amigos de la universidad, Jonathan Acosta, Renny Coro y Carlos Alvarez, quienes siempre fueron generosos con sus conocimientos.

Muchas gracias a nuestro tutor de tesis Ing. Hugo Ortiz, quien apoyó con todos sus conocimientos para el desarrollo del proyecto.

A mi universidad y a todos los ingenieros que me dieron clases. Gracias a su exigencia, he forjado una personalidad fuerte la cual de seguro me va a permitir superar cualquier adversidad tanto a nivel profesional como personal.

Finalmente, quiero agradecer a Dios por permitirme culminar una etapa más de mi vida, por haberme protegido durante todo este proceso y por todas las bendiciones recibidas.

Hugo Andrés Cisneros Salazar

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

En el presente trabajo se diseña e implementa un laboratorio remoto, que permita el desarrollo de prácticas enfocadas en el aprendizaje sobre la utilización de un variador de velocidad y la sintonización de controladores PI/PID para el manejo de un sistema de bombas. El laboratorio remoto consta de una maqueta con equipos que representan el sistema de bombeo, y está implementada en el laboratorio de Robótica y PLCs de la universidad. Se accede al manejo de la maqueta mediante una aplicación web desarrollada en Django, donde el estudiante puede registrarse, revisar las prácticas disponibles, agendar un horario y realizar las prácticas visualizando el estado de las variables y la imagen de la maqueta, ambos en tiempo real. En cuanto al acceso remoto, la aplicación fue alojada en la plataforma de servicios Heroku, para que los usuarios accedan al laboratorio desde cualquier parte del mundo mediante un navegador web. El enlace remoto se consigue utilizando una tarjeta Rasberry Pi 4, cuya función es la transmisión de datos desde la aplicación web hacia la maqueta y viceversa. A partir de las pruebas de transmisión realizadas, se determinó que la propuesta de enlace remoto es fiable, continuo y rápido; mientras que, los resultados asociados a la usabilidad y la enseñanza, indican que los estudiantes pudieron desarrollar las prácticas de forma autónoma, y alcanzaron los objetivos de aprendizaje.

PALABRAS CLAVE:

- **LABORATORIO REMOTO**
- **CONTROLADORES PID**
- **PYTHON**
- **HEROKU**

Abstract

In the present work a remote laboratory is designed and implemented, allowing the development of practices focused on learning about the use of a variable frequency drive and PI/PID controllers tuning for a pump system management. The remote laboratory consists of a mockup with equipment that represents the pumping system, and is implemented in the Robotics and PLCs laboratory of the university. The mockup is accessed through a web application developed in Django, where the student can register, review the available practices, schedule a session and perform the practices visualizing the state of variables and the image of the mockup, both in real-time. As for remote access, the application was hosted on the platform as a service Heroku, so that users can access the laboratory from anywhere in the world just using a web browser. The remote link is achieved using a Raspberry Pi 4 board, whose function is to transmit data from the web application to the mockup and vice versa. From transmission testing, it was determined that the proposed remote link is reliable, continuous and fast; while, the results associated with usability and teaching indicate that students were able to develop the practices by themselves, and achieved the learning goals.

KEYWORDS:

- **REMOTE LABORATORY**
- **PID CONTROLLERS**
- **PYTHON**
- **HEROKU**

Capítulo I. Marco Metodológico

Antecedentes

El constante desarrollo de tecnologías de conectividad ha impulsado a varias instituciones académicas a desarrollar prototipos y diseños que incluyen el concepto de conexión remota para laboratorios técnicos, los mismos que permiten el aprendizaje práctico de los usuarios que lo requieran. Antes, un proyecto de este estilo se podía considerar un reto; sin embargo, hoy en día es una necesidad latente debido a la aparición de una pandemia que, de forma inesperada, restringe a los estudiantes el acceso de forma presencial a los laboratorios. Dicho esto, entiéndase como laboratorio remoto a la integración de tecnologías hardware y software que permitan al usuario controlar mediante internet un proceso industrial real, el cual puede estar representado a través de una maqueta que presente dispositivos físicos tangibles (Abrams, 2020).

Es inevitable que, con el paso del tiempo, el constante desarrollo de las tecnologías y la mejora de conocimientos por parte de los seres humanos que están relacionados a la ingeniería desarrollen proyectos sofisticados y de gran nivel; pero, es necesario inicialmente realizar una retrospectiva en el tiempo para evaluar el avance de los laboratorios remotos, tanto a nivel internacional como nacional. En la Universidad Politécnica Antonio José de Sucre ubicada en Venezuela, se desarrolló un laboratorio remoto enfocado al monitoreo y control de las variables de la maqueta de un proceso de alimentos ubicado en su laboratorio. Los estudiantes acudían a realizar la práctica de forma presencial y sólo si el horario no les permitía cumplir con las actividades planteadas, se podía culminar de forma remota, con el objetivo de elaborar informes técnicos y visualizar la maqueta industrial desde cualquier lugar. Es decir, fue una herramienta totalmente auxiliar tanto para el alumnado como para el docente de la institución (Cruz et al., 2010).

De forma diferente a Venezuela, Cuba le dio otra perspectiva al uso de laboratorios remotos, priorizando su utilización como herramienta primaria para el aprendizaje continuo de los usuarios dentro del campo de la Ingeniería en Control. En esta localidad, se implementó un sistema de laboratorio remoto en donde el usuario podía identificar sistemas, sintonizar técnicas de control y probarlas de forma remota en dispositivos reales. Es importante destacar de este trabajo, que el dispositivo electrónico controlador de todo el sistema fue la tarjeta Arduino y el principal protocolo de comunicación a través de Ethernet (Mar-Cornelio et al., 2019). En adición, es preciso señalar que, en la literatura internacional aparte de los dos trabajos citados, existen una gran cantidad de trabajos relacionados a los laboratorios remotos que incluyen diferentes tecnologías de conectividad, controladores y servidores web (Saavedra, 2016; Villafañe et al., 2019; Jiménez et al., 2010; Vargas et al., 2020).

En el contexto local, el desarrollo de laboratorios remotos se ha limitado a pocos trabajos llevados a cabo por personal o estudiantes de las Instituciones de Educación Superior a manera de prototipo. Una de las pocas excepciones es el laboratorio remoto realizado en la Universidad Particular de Loja UTPL en el año 2017. Sin embargo, el campo que cubre este laboratorio únicamente se enfoca en la asignatura de Física (Carrera, 2017).

Desde hace aproximadamente diez años que se viene desarrollando proyectos de grado a nivel local enfocados al aprendizaje del control, supervisión y monitoreo de pequeños procesos industriales y de la interacción con dispositivos de campo tales como controladores lógicos programables, elementos que se utilizan en los electrofluidos y tecnologías de conectividad. Dentro de esta área, uno de los primeros trabajos de titulación fue realizado en la Universidad del Azuay, el cual se puede resumir en el diseño de un prototipo de laboratorio remoto mediante el cual se pueda realizar prácticas de control y automatización de procesos a través de la programación

remota de un PLC de la marca Siemens S7-1200 vía web y su posterior monitoreo a través de un SCADA desarrollado en LabVIEW (Alvarado & Sánchez, 2011).

La Universidad de las Fuerzas Armadas también ha sido un eje fundamental en el desarrollo de proyectos de grado sobre conectividad remota destacando en su repositorio dos de ellos. En la matriz de Sangolquí se implementó un laboratorio virtual y remoto para desarrollar prácticas de control de procesos y en específico el monitoreo de una planta de transmisión de fluidos. Utilizando el software LabVIEW y una tarjeta de adquisición de datos de National Instruments, diseñaron la interfaz de usuario y la comunicación con la planta. Además, el enlace para el manejo remoto se establece con el software TeamViewer. La deficiencia de este trabajo resulta ser la conectividad en relación a los tiempos de respuesta, contexto dentro del cual se propone mejorar (Arrieta & Piedra, 2015). Otra alternativa diferente a la mencionada se realizó en la sede Latacunga, donde se implementó un sistema de supervisión remoto en el cual los usuarios tenían acceso a la programación básica de un PLC físico ubicado en el laboratorio real. No obstante, el tema central de este trabajo consistió en la conectividad del usuario mediante un sistema de videovigilancia, que le permitía visualizar las salidas del controlador en tiempo real. Para esto, utilizaron un servidor proporcionado por TeamViewer el cual les sirvió de base para realizar toda la conectividad remota entre la web y el laboratorio real de la universidad (Quisnancela & Iza, 2016).

Finalmente, y dentro de lo más reciente, la Escuela Superior Politécnica de Chimborazo dio importancia al uso de un laboratorio virtual de forma complementaria a un laboratorio remoto para el diseño de circuitos electrohidráulicos. La satisfacción por parte de los estudiantes que utilizaron este sistema es notoria y positiva, considerando que este método de aprendizaje también es válido (López-Ortiz et al., 2020).

Es evidente que la situación actual en Ecuador, en el desarrollo de laboratorios remotos, todavía puede mejorar y en base a los trabajos revisados se

puede concluir que se requiere integrar todos los conceptos del manejo de controladores lógicos programables, redes industriales, control de procesos, entre otros; con el objetivo de disponer de un laboratorio que pueda ser controlado adecuadamente de forma remota. Para una correcta integración de estos componentes se requiere de un sistema de conectividad que, utilizando tecnologías vigentes y óptimas, permitan un funcionamiento robusto de la aplicación, todo esto para beneficio de los estudiantes y de la universidad.

Justificaciones e importancia

A raíz de la pandemia originada por el virus SARS-CoV-2, las clases en escuelas, colegios y universidades migró a modalidad de educación en línea. A pesar de que se trataba de una metodología de enseñanza nueva para muchos docentes y estudiantes, con el pasar del tiempo, se fueron acoplando a las condiciones para continuar con el proceso de enseñanza - aprendizaje. Sin embargo, las asignaturas de carácter práctico claramente tienen una limitación con la educación virtual, por la incapacidad de asistir a laboratorios para utilizar directamente las máquinas y equipos. Muchos docentes se apoyan de simuladores para paliar este inconveniente y mantener la calidad de enseñanza en sus asignaturas; no obstante, hay detalles que no pueden ser tratados de forma virtual o su comprensión es mejor cuando el estudiante experimenta y observa los efectos en los elementos reales.

Aún es muy pronto para determinar los efectos que la educación en línea está teniendo en la formación de los estudiantes de ingeniería; sin embargo, se han publicado estudios donde muestran la percepción que los mismos estudiantes tienen respecto al cambio en la modalidad y su efectividad en la enseñanza (Pérez-López et al., 2021), donde la mayoría manifiesta que ha exigido mayor dedicación, pero a pesar del incremento en el tiempo de estudio, no perciben proporcionalidad en el rendimiento académico. Además, si se tratan de clases donde se imparten los conceptos de carácter

práctico (Revilla-Cuesta et al., 2021), la mayoría señala que ha tenido dificultad de comprender los conceptos y creen que no son suficientes para afrontar un problema real. Estos resultados hacen notar que la enseñanza en línea no es igual de efectiva que en clases presenciales, y menos aún al tratar conceptos prácticos, evidenciando la necesidad de buscar alternativas para mejorar la metodología de enseñanza.

Por otro lado, el desarrollo de prácticas presenciales generalmente presenta el inconveniente de tener disponibilidad limitada de laboratorios y número limitado de equipos. La implementación de laboratorios remotos a través de internet, remedia este condicionamiento, aumenta el número de prácticas de laboratorio que pueden realizar los estudiantes y proporciona horas de acceso más extensas y flexibles. Esto, respecto a la enseñanza, provoca un incremento de la dedicación del estudiante a la realización de las prácticas que, de cierto modo, permite una mejora en los resultados de aprendizaje. Adicionalmente, en el contexto de la pandemia provocada por el covid 19, los laboratorios remotos permiten emplear los equipos y programas reales, sin tener que exponer a estudiantes y profesores a contagiarse del coronavirus.

Por estos motivos, en el presente proyecto se diseña e implementa un laboratorio remoto que permita a los estudiantes de la carrera de ingeniería en Electrónica, Automatización y Control, y futuras generaciones de la carrera de Electrónica y Automatización, aprender a sintonizar controladores PID para el manejo de bombas, máquinas muy usadas en procesos industriales cuando se necesita controlar el caudal de algún fluido. Se escoge este tipo de controladores porque en la industria, a pesar de la aparición de nuevas estrategias de control que presentan un gran rendimiento como controladores basados en redes neuronales, lógica difusa, controladores predictivos, entre otros; la gran mayoría de controladores utilizados en la regulación y seguimientos siguen siendo tipo PI o PID.

Alcance del proyecto

El presente proyecto consiste en el diseño e implementación de un laboratorio remoto, que permita el desarrollo de prácticas enfocadas al aprendizaje sobre la utilización de un variador de velocidad y la sintonización de controladores PI y PID para el manejo de bombas en aplicaciones industriales relacionadas al control de presión frente a variaciones en la demanda.

El proyecto se resume en 3 secciones como muestra la Figura 1 las cuales permitirán el diseño del laboratorio remoto y su acceso a un proceso industrial representado mediante una maqueta física para su posterior control.

Figura 1

Secciones definidas para el diseño e implementación del laboratorio remoto

Maqueta

Como el principal objetivo del proyecto es el manejo de bombas en aplicaciones industriales, la maqueta dispone de los equipos necesarios para representar el proceso mencionado anteriormente. Este entorno físico está ubicado en el laboratorio de Robótica y PLCs de la Universidad de las Fuerzas Armadas ESPE sede Matriz en Sangolquí y consta de los siguientes componentes:

- PLC Siemens S7-1200 CPU 1212C AC/DC/RLY

- Variador de velocidad Schneider Electric ATV630U15M3
- Motor trifásico Baldor de ½ hp
- Pulsadores industriales
- Indicadores LED industriales

La importancia de la maqueta radica en la posibilidad que se le otorga al usuario de interactuar con aparatos industriales reales de forma remota sin necesidad de acceder de manera presencial al espacio físico.

El proceso que representa la maqueta consiste en el manejo de 4 bombas para mantener una presión estable independientemente de la demanda que se requiera. Para controlar dicho proceso se utiliza el autómata programable Siemens S7-1200 CPU 1212C; como resultado del control, las bombas que funcionen con su máxima capacidad para cubrir la demanda se visualizan a través de leds industriales, mientras que la bomba que permita satisfacer totalmente la demanda se representa por el motor trifásico Baldor de 1/2 hp y es controlada por el variador de velocidad Schneider Electric ATV630U15M3. Cabe recalcar que este proceso siempre requiere como mínimo del control de una bomba. En adición, se instaló un paro de emergencia como medida de protección en caso de fallo en alguno de los equipos.

Visualización

Esta sección se compone de los elementos que cumplen un rol de supervisión en el funcionamiento de la maqueta, tanto a nivel local como externo al laboratorio. Para el primer enfoque, se dispone de un panel de operador Magelis HMI STU855 de la marca Schneider, el cual muestra información respecto al estado de la maqueta para que, el docente o persona que se encuentre próximo a los equipos, pueda tomar acciones en caso de así considerarlo. También se puede modificar la demanda desde el panel de operador.

En relación a la supervisión remota, se añade una cámara para que el estudiante visualice el desempeño de sus controladores sin necesidad de que se encuentre presente dentro del laboratorio.

Las actividades asociadas a la sección de visualización consisten en el diseño de las interfaces para el panel de operador, establecer la comunicación con el PLC Siemens, y realizar la conexión y configuración de la cámara para que exista un envío de la imagen en tiempo real hacia la interfaz web.

Acceso remoto

El usuario tiene acceso a una interfaz web a través de un servidor implementado mediante una tarjeta Raspberry PI 4. En consecuencia, el laboratorio remoto se resume en dicho entorno web, ya que desde aquel sitio el usuario puede realizar las siguientes acciones:

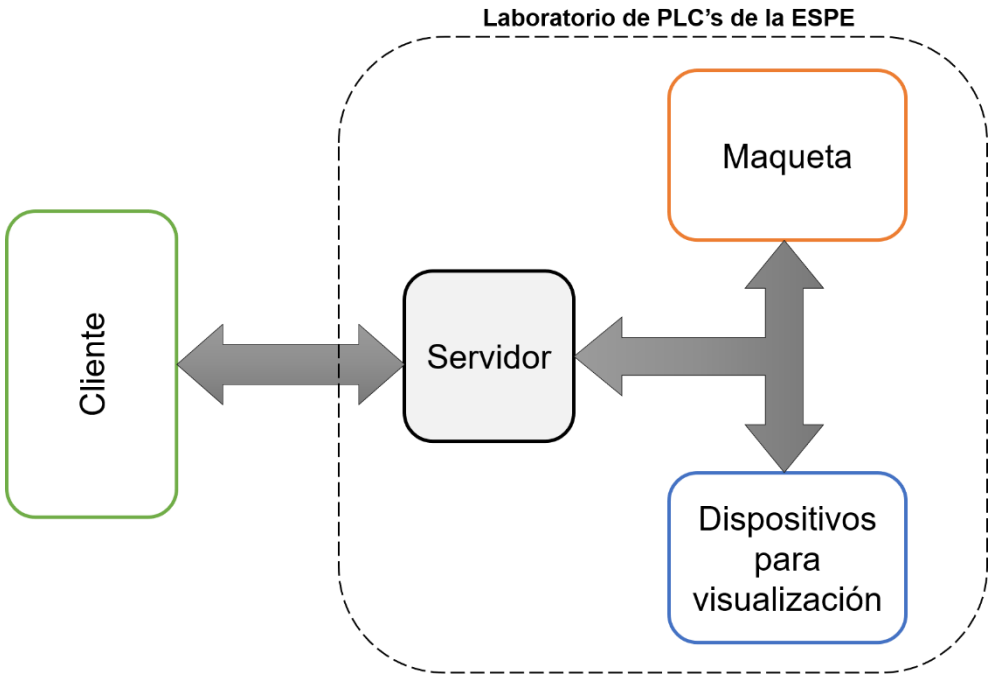
- Configurar un variador de velocidad
- Establecer la demanda requerida.
- Seleccionar tipos de controladores: PI, PID.
- Sintonizar los parámetros de los controladores.
- Visualizar la maqueta en tiempo real.
- Visualizar la dinámica de la planta (demanda) y la velocidad de la bomba controlada.

Debido a que el funcionamiento de la maqueta se desarrolla en tiempo real, el acceso a la misma debe ser organizado y por horarios. Para esto, la interfaz web permite la opción al usuario de registrarse para autorizar el control de la planta. De esta forma, el acceso al laboratorio remoto será en una hora establecida y un estudiante a la vez.

Finalmente, en la Figura 2 se indica el diagrama de bloques del laboratorio remoto implementado en el laboratorio de Robótica y PLC de la Universidad de las Fuerzas Armadas ESPE sede Matriz.

Figura 2

Diagrama de bloques del laboratorio remoto implementado en el laboratorio de Robótica y PLC de la Universidad de las Fuerzas Armadas ESPE sede Matriz



Objetivos

Objetivo general

Implementar un laboratorio remoto para el aprendizaje en sintonización de controladores PID enfocado al manejo de bombas dentro del laboratorio de Robótica y PLC de la Universidad de las Fuerzas Armadas ESPE mediante la integración de equipos industriales y tecnologías de software.

Objetivos específicos

- Diseñar la red para intercomunicar a los equipos industriales que conformarán la maqueta en representación de una aplicación enfocada al manejo de bombas.

- Diseñar el enlace de comunicación que permita la interacción del usuario con la maqueta física por medio de un dispositivo remoto.
- Desarrollar la interfaz de aprendizaje que permita el ingreso mediante usuario y contraseña, el registro para acceso restringido por horarios y la realización de prácticas de sintonización de controladores PID a través de entornos visuales.
- Implementar el entorno de hardware conformado por los equipos industriales de la maqueta y los elementos que habilitan el acceso remoto.
- Realizar pruebas de funcionamiento que permitan el análisis y evaluación del laboratorio remoto en aspectos de conectividad y control de la maqueta.

Capítulo II. Marco Conceptual

Laboratorio remoto

El laboratorio remoto es la integración de tecnologías hardware y software cuyo objetivo es que un usuario realice un proyecto o experimento práctico desde cualquier lugar a través de internet tal y como se lo realizaría en un laboratorio presencial (Daley, 2020).

Características

Se pueden resumir en las siguientes,

- Aplicación basada en un entorno cliente – servidor. Por el lado del cliente, una computadora es conectada al Internet a través de un navegador web; mientras que, del lado del servidor existen dos componentes: el servidor de laboratorio y el servidor web. El servidor de laboratorio implica la conexión de un computador a los componentes hardware del experimento real y opcionalmente conectado a una cámara web. Por otro lado, el servidor web está conectado al servidor del laboratorio y este permite el acceso de los clientes a la configuración y modificación del experimento (Riman et al., 2011).
- Multidisciplinario. Los laboratorios remotos son aplicables para cualquier rama y disciplina ya sea de la ingeniería como de otro entorno. Ejemplos de la misma pueden ser la Ingeniería Mecánica, Electrónica o bien las Ciencias Experimentales en un nivel inferior.
- Distintos medios de interconexión como Internet o Intranet. Aun así, el medio más común y más utilizado de los laboratorios remotos es el acceso a través de Internet y mediante un navegador web.

Ventajas

Un laboratorio remoto proporciona las siguientes ventajas:

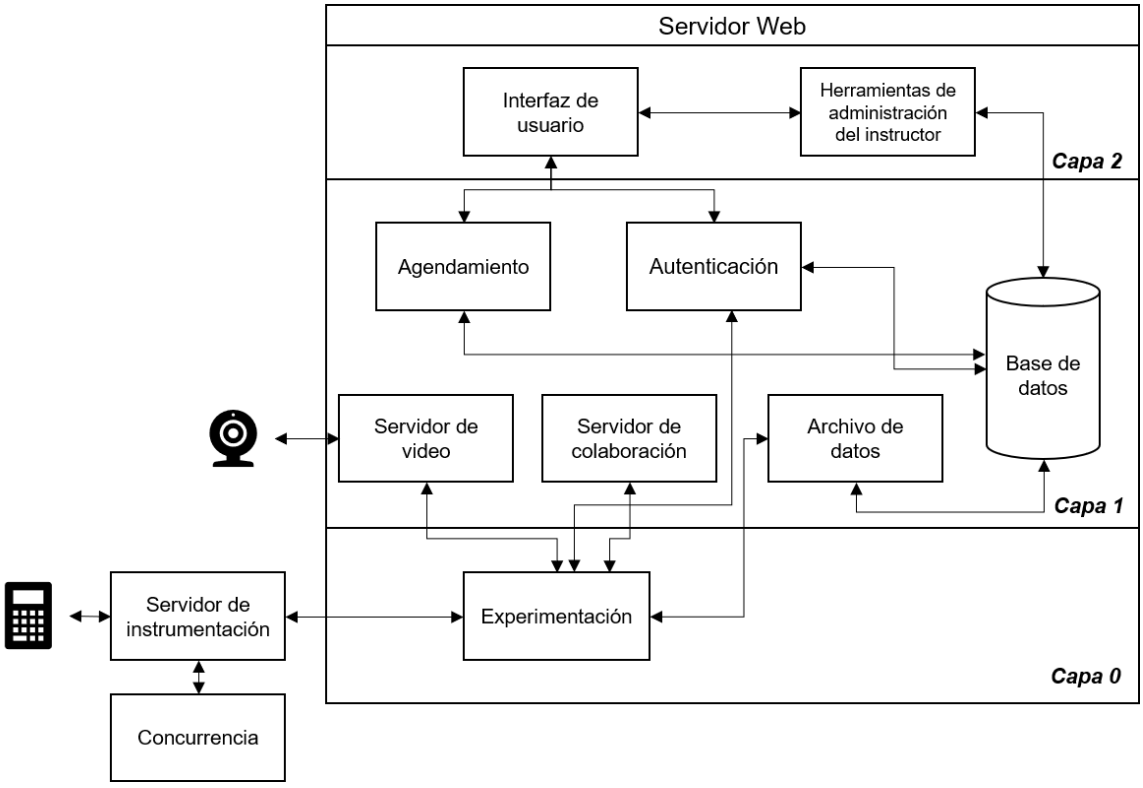
- Evita transporte de los usuarios a laboratorios presenciales y en consecuencia costos por movilización.
- Accesibilidad para personas discapacitadas.
- Incentiva el autoaprendizaje de los usuarios que son alumnos.
- Visualización del hardware en tiempo real a través de software.
- Compartimiento de recursos de las universidades u organizaciones en equipos de alto costo.
- Almacenamiento de información proveniente de los experimentos para obtener resultados y analizarlos de tal forma que se puedan hallar conclusiones.
- Demostración de experimentos en entornos reales en talleres, conferencias o seminarios.

Servicios

Existen por lo menos 8 modelos referentes a los laboratorios remotos y los cuales son expuestos por Mougharbel et al. (2006). Sin embargo, los mismos autores establecieron un nuevo modelo mejorado el cual engloba todos los servicios que un laboratorio remoto podría tener (Riman et al., 2011). En la Figura 3, se observa el modelo mencionado.

Figura 3

Modelo de laboratorio remoto



Nota. Adaptado de *A remote Lab Experiments Improved Model*, por Riman, Hajj y Mougharbel, 2011, International Journal of Online Engineering 7(1), doi:10.3991/ijoe.v7i1.1460.

Interfaz de usuario. Está presente en cualquier laboratorio remoto y consiste en una aplicación web que conecta al usuario con el experimento real que se esté ejecutando. Dispone usualmente de lo siguiente:

- Ingreso de nombre de usuario y contraseña.
- Modificación de contraseña.
- Añadimiento, modificación o verificación de horarios programados.
- Descarga de datos obtenidos anteriormente.
- Iniciar y realizar el proyecto o experimento.

- Lectura de información general proporcionada por el instructor.
- Envío de informes de laboratorio.

Herramientas de configuración del instructor. Son páginas web que ayudan al instructor para que puedan configurar el experimento de forma remota. Pueden realizar las siguientes funciones:

- Publicar información del experimento de laboratorio.
- Añadir, modificar o eliminar experimentos de laboratorio mediante base de datos.
- Publicar anuncios en cualquier momento.
- Añadir, modificar o borrar usuarios o grupos de usuarios mediante bases de datos.
- Asignar horarios disponibles para experimentos.
- Publicar calificaciones de los reportes de laboratorio.
- Iniciar y supervisar un experimento con los usuarios.

Autenticación. Permite acceder al laboratorio remoto solo a estudiantes registrados. Para esto los usuarios se registrarán con un nombre de usuario y contraseña, esta información debe ser ingresada antes de realizar la práctica.

Planificación o agendamiento. Este aspecto debería implementarse para evitar conflictos de congestión de usuarios en el acceso al laboratorio, se debería tomar en cuenta lo siguiente:

- Cada experimento tiene un horario de acceso para los usuarios.
- Los horarios son establecidos por el instructor.
- Los horarios se manejan en una base de datos y se comprueban en el inicio de sesión de un usuario.

Servidor de video. El experimento se puede ver mediante una video cámara. Esto va a depender del diseñador del laboratorio remoto; pero, se detallan las opciones de este servidor:

- Se puede utilizar una o varias cámaras para visualizar en tiempo real el experimento.
- El usuario puede escoger qué cámara utilizar.
- Es posible diseñar una página web para mostrar la salida de la cámara.

Archivado de datos. Los estudiantes podrían guardar los resultados del experimento en la base de datos con un cierto nombre especificado. Esta información se guardaría y se la podría extraer en cualquier momento. Además, se la podría visualizar en cualquier formato de archivo (.txt, .doc, .xlx).

Base de datos. Capaz de almacenar y extraer:

- perfiles de usuario
- información de instructor
- lista de experimentos
- estudiantes registrados
- horarios de los experimentos
- datos guardados
- datos de acceso mediante inicio de sesión
- entre otros y según considere el diseñador del laboratorio remoto

Experimentación. Cambiar los parámetros de un experimento acorde a la rama a la cual se esté experimento. Por ejemplo, en la Ingeniería Electrónica se podría modificar parámetros de un controlador PID.

Acceso remoto

El acceso remoto es la capacidad de una persona autorizada para acceder a una computadora, dispositivo o red desde una distancia geográfica a través de internet u otro método de telecomunicaciones (Rosencrance, 2021).

El acceso remoto se logra con una combinación de software, hardware y conectividad de red.

Servidor web

Se utiliza un servidor web para almacenar y entregar el contenido del sitio web. Estos pueden ser imágenes, textos, datos de aplicaciones, videos y muchos más según la solicitud del cliente. El navegador web solicita datos del sitio web, lo que ocurre cuando el usuario hace clic en un enlace en particular o hace clic para descargar un documento que se ha mostrado en el navegador (Upadhyay, 2020).

Un servidor web no es más que software y hardware que utiliza el Protocolo de Transferencia de Hipertexto, comúnmente conocido como HTTP, y algunos otros protocolos que responden a las solicitudes de los clientes realizadas en la *World Wide Web*. El trabajo principal que realiza el servidor web es mostrar el contenido del sitio web, lo que se está almacenando, luego procesando y finalmente entregando las páginas web al usuario que lo ha solicitado.

El hardware del servidor web se conecta a Internet, lo que permite el intercambio de datos con otros dispositivos relacionados con él. El software del servidor web controla cómo el usuario evalúa los archivos que se han alojado.

Se accede al software del servidor web utilizando el nombre de dominio de un sitio web. Esto asegura que el contenido del sitio se entregue al usuario que lo solicitó. La parte de software del servidor web también se compone de varios componentes y tiene al menos un único servidor HTTP.

Hardware. Las computadoras utilizadas como servidor web generalmente tienen más memoria, controladores de disco duro más grandes y más rápidos y procesadores más rápidos que las computadoras de escritorio o portátiles típicas. Aunque la capacidad de procesamiento y de memoria del hardware depende de la cantidad de peticiones que se tiene previsto tendrá el servidor.

Dentro del mismo hardware, se puede tener varios servidores virtuales o host virtuales, los cuales consisten en particiones de los recursos que dispone el hardware para mantener el software de varios servidores, pudiendo administrarse de forma independiente.

Raspberry Pi. Raspberry Pi es la denominación a la línea de computadores de placa única fabricados por la Fundación Raspberry Pi. La primera versión fue lanzada en 2012, y desde entonces han ido evolucionando en las prestaciones de los ordenadores. La CPU de la primera versión de la Raspberry Pi era de un solo núcleo a 700 MHz y 256 MB de RAM, mientras que el modelo más reciente tiene una CPU de cuatro núcleos a más de 1,5 GHz y hasta 8 GB de RAM (Red Hat, Inc., 2021).

Se pueden utilizar para instalar servidores HTTP que procesen la aplicación del lado del servidor, y darle un enlace para el acceso público, con el fin de poder acceder desde cualquier parte a la aplicación web.

Servicios en la nube. Se basan en la utilización de una red de servidores remotos que se utilizan por medio del internet y que hacen que una aplicación sea accesible desde cualquier dispositivo, además de permitir administrar, manejar y almacenar información en bases de datos y servidores. Existen tres tipos de servicios en la nube:

- SaaS (software como servicio)

Este tipo de servicio aloja el software en servidores de propietarios y el usuario puede disponer del mismo por medio de la web.

- PaaS (plataforma como servicio)

Las plataformas como servicio son aquellas que permiten al usuario acceder a un entorno en el que pueden crear aplicaciones y muchos de los sitios soportan varios lenguajes de programación (Llordachs, 2021).

Las plataformas de servicio en la nube más importantes actualmente son: Heroku de Salesforce.com, Velneo Cloud de Velneo, App Engine de Google, Azure de Microsoft, Cloud Applications de Oracle, Cloud de IBM, entre otros.

- IaaS (infraestructura como servicio)

El propietario de los servicios entrega software y aplicaciones por el internet. Los clientes acceden a los mismos a través de un registro en el software por medio de la web o de la interfaz de programación de los propietarios.

Lenguajes de programación para diseño de aplicaciones web. En primer lugar, una aplicación web es una herramienta donde un usuario accede a un servidor web en una determinada red y por medio de cualquier tipo de navegador. La red típicamente puede variar entre internet o intranet. Para ingresar a la aplicación web no es necesario de ninguna instalación en la computadora, ya que se ingresa a través del navegador y además se programa en lenguaje HTML (Valarezo et al., 2018). A continuación, algunos de ellos:

GO. Es un lenguaje compilado, estructurado, concurrente y orientado a objetos que se guía en C. Está disponible en varios sistemas como Linux o Mac OSX y en formato binario para sistemas como Windows. Una de sus principales características es que es un lenguaje de código abierto que incluye simplicidad, seguridad y rapidez en la programación.

Python. Quizá el lenguaje más poderoso para la creación de aplicaciones web, dispone de estructuras de datos bastante eficientes y además una perspectiva de programación orientada a objetos. La sintaxis de este lenguaje permite a los

desarrolladores implementar menos líneas de código con respecto a otros lenguajes y una de sus ventajas es que no es necesario compilar. En adición, es un software de código libre.

Java. A diferencia de Python, un programa desarrollado en Java debe compilarse para que el código resultante en bytecodes sea entendido por una máquina virtual. La sintaxis es similar a la de C++, sin embargo, no tienen relación.

C#. Este lenguaje une las características de otros lenguajes de programación como C++, Java o Visual Basic y los integra en uno solo. El compilador de este lenguaje es el más depurado de los que incluye el .NET Framework SDK (Valarezo et al., 2018).

PHP. Es un lenguaje general que se encuentra del lado del servidor y que generalmente sirve para diseño de desarrollo web con contenido dinámico. Es capaz de combinar código HTML y PHP en un solo archivo de un servidor definido por el desarrollador. Soporta a varias bases de datos, el cual es un parámetro importante actualmente para el desarrollo de aplicaciones web. Para el correcto funcionamiento se requiere la instalación de un servidor Apache. Su sintaxis es tomada de los lenguajes C, Java y Perl (Arbeláez et al., 2011).

JavaScript. Es un lenguaje interpretado y uno de los más utilizados para el diseño de páginas web, debido a que no es necesario compilar el programa para poder ejecutarlo. La diferencia con PHP es que los procesos se ejecutan del lado del cliente; es decir, en el navegador, y en consecuencia, no se requiere de intercambio de datos con un servidor. Adicionalmente, este lenguaje se asemeja a Java y C.

Ruby. Aunque el rendimiento de este lenguaje es mejor con Unix, puede trabajar en varios sistemas operativos. Ruby es un lenguaje de programación dinámico con sintaxis compleja, pero con una gran cantidad de librerías de clase. Por lo tanto, está enfocado a la programación orientada a objetos; sin embargo, también es apto para programación funcional o imperativa.

HTML5. Fue creado para establecer compatibilidad en la publicación de las aplicaciones en la web. Se caracteriza por incluir una gran cantidad de componentes dinámicos en forma de etiquetas, que configuran la aplicación en la web.

La sintaxis admite tanto HTML como XHTML para la programación. La primera es más apta para aplicaciones poco exigentes, mientras que la segunda tiene un énfasis más académico y riguroso.

Bases de datos. Dentro del contexto de laboratorios remotos, se tiene que hablar de un servidor de base datos el cual es un tipo de software que sirve para organizar información a través de tablas, índices y registros. A nivel de hardware, un servidor de base de datos está configurado para servir consultas a clientes remotos o locales que piden información pero que también modifican las tablas, índices o registros de la base de datos desde un servidor web (Borges, 2019).

Los programas que pueden gestionar las bases de datos se los conoce como sistemas gestores de bases de datos (SGBD). SQL es el lenguaje de consulta más utilizado, Microsoft SQL Server y Oracle Database son soluciones privadas muy utilizadas, y en la web MySQL, PostgreSQL y SQLite son soluciones libres muy comunes y que se detallarán a continuación (Casado, 2021). A continuación, algunas de ellas:

MySQL. Es un sistema de gestión de base de datos de código abierto que trabaja con datos relacionales, por lo tanto, usa tablas múltiples que se interconectan entre ellas para almacenar información y ordenarla de forma eficiente. Es utilizado en aplicaciones web que utilizan lenguajes como PHP, Ruby o Python (Robledano, 2019).

Las principales características de este sistema son:

- Su arquitectura es cliente y servidor, es decir, el cliente puede realizar consultas por medio del sistema de registro para obtener, modificar y guardar datos.

- Tiene compatibilidad con SQL.
- Permite automatizar tareas dentro de la base de datos, es decir, cuando se da cierto evento, se lanza otro para modificar registros.
- Permite configurar vistas personalizadas lo que resulta una ventaja cuando se trabaja con grandes bases de datos.

Las sentencias más utilizadas en MySQL son:

- SELECT: sirve para consultar datos.
- DISTINCT: elimina duplicados de las consultas.
- WHERE: para añadir condiciones en la consulta de datos.
- AND/OR: añade varias condiciones en la consulta.
- ORDER BY: ordena resultados de la consulta.
- INSERT: inserta datos.
- UPDATE: actualiza datos que ya existen.
- DELET: borra datos.

PostgreSQL. Es un sistema de gestión de base de datos de código abierto, de datos relacionales y está orientado a objetos. Sus principales características son:

- Alta concurrencia o conocido como acceso concurrente multiversión. PostgreSQL posibilita que mientras es escribe en la tabla, se pueda acceder a la misma sin bloquear la lectura (Platzi, 2015).
- Los usuarios tienen la opción de crear sus propios tipos de datos.
- Permite incluir disparadores, es decir, que se realice una acción cuando se dé un evento en específico.
- Se puede utilizar varios lenguajes de programación como C, C++, Java, Python, Ruby, entre otros.
- Herencia de tablas.

- Una de las herramientas para administrar las bases de datos en PostgreSQL se la denomina pgAdmin, la cual permite ejecutar comandos SQL.

SQLite. Es un sistema de gestión de base de datos relacional de código libre que es utilizada para guardar información principalmente en dispositivos con poca capacidad de hardware como los celulares, sin embargo, también se puede utilizar en computadores de escritorio o portátiles. Al implementar el estándar SQL92, soporta las consultas más sencillas hasta las más complejas del lenguaje SQL (Rómmel, 2021).

Las principales características son:

- La base de datos se ubica en un solo archivo.
- Su versión más actual y estable es la 3 y en esta se permite implementar bases de datos de hasta 2 Terabytes.
- Soporta datos numéricos de 64 bits y texto en formato UTF-8/16.
- Está construida en C, lo que permite migrar a diversas plataformas de sistemas operativos. SQLite funciona en plataformas como Windows, Mac OSX, Linux, entre otros.
- Los lenguajes de programación que se pueden utilizar para este sistema son Java, Perl, Python, PHP, Ruby, C#, entre otros.

Cliente web

Un cliente web es todo lo que se muestra en una aplicación que puede ser accedida desde el navegador web. Esto incluye lo que ve el usuario, como texto, imágenes y el resto de la interfaz de usuario, junto con las acciones que realiza una aplicación dentro del navegador del usuario (Cloudflare, 2021).

El cliente web es un software que le ayuda a interactuar con los servidores web y recopilar los datos procesados de los servidores. Los lenguajes de marcado como HTML y CSS son interpretados por el navegador del lado del cliente.

Equipos industriales

En el contexto de un laboratorio remoto, los equipos industriales son aquellos componentes que conforman e integran el hardware de todo el sistema. Generalmente, dentro de un experimento son aquellos que componen una maqueta. Ejemplos de equipos industriales son las bombas eléctricas, molinos, trituradores, motores, PLCs, DCS, routers, sensores, entre otros. Sin embargo, de forma conceptual, se definen los equipos que serán utilizados en el presente trabajo.

Controlador Lógico Programable (PLC)

Según la NEMA (Asociación Nacional de Fabricantes Eléctricos) un controlador lógico programable es un instrumento electrónico, el cual utiliza memoria programable para almacenar instrucciones que implementen ciertas funciones tales como operaciones lógicas, secuencias de acciones, registro y control de tiempos, contadores y cálculos que controlen por medio de módulos de E/S tanto analógicos como digitales en diferentes máquinas y procesos (Belloso, 2021).

Existen diversos fabricantes de PLCs, pero las más importantes y utilizadas en la industria son:

- Allan Bradley
- Siemens
- Omron
- Schneider Electric
- Mitsubishi
- Delta
- Festo
- Bosch
- ABB

Variador de velocidad

Se define como un regulador industrial ubicado entre la alimentación de energía y un motor. Por lo tanto, un variador regula la velocidad de motores eléctricos para que la energía eléctrica que alimenta al motor se adapte a la demanda real de una aplicación, bajando la potencia del motor entre un 20 y 70%. Entonces, la energía eléctrica circula por el variador y regula la energía antes de que llegue al motor para luego modificar frecuencia o tensión acorde a la aplicación que se esté realizando. Un ejemplo que es muy común de los variadores de velocidad es la reducción de potencia de salida de una bomba a través del control de velocidad de un motor el cual o permite que funcione a una velocidad mayor a la que se necesita (ABB, 2021).

Las marcas más comunes de variadores de velocidad son,

- Siemens
- Allen Bradley
- Schneider Electric
- Delta
- Omron
- ABB
- WEG

Motor trifásico

Son máquinas eléctricas que convierten la energía eléctrica en mecánica mediante interacciones electromagnéticas y que funcionan con la potencia de la corriente alterna trifásica. Estos motores son impulsados por tres corrientes alternas de igual frecuencia las cuales sus máximos picos se alternan. Generalmente, tienen una potencia de hasta 300kW y velocidades entre 900 hasta 3600 RPM, pero estos parámetros van a depender de los fabricantes (CLR, s.f.).

Las principales marcas de motores trifásicos son,

- TECO
- ABB Baldor
- WEG
- Sumitomo

Panel de operador HMI

Es la interfaz cuyo objetivo es la interacción y comunicación entre el operador y el proceso productivo controlado. Entonces, es un panel en donde el operario puede modificar parámetros para controlar, observar, coordinar y supervisar un proceso industrial. Un HMI transforma las variables de un proceso industrial en información que sirva y sea procesable (Aula21, 2020; Autycom, 2019).

Las funcionalidades del HMI se resumen en la siguiente lista:

- Visualización de información en tiempo real.
- Supervisión de entradas y salidas de máquinas.
- Seguimiento de tiempo de producción.
- Gráficos visuales que muestren estados de equipos industriales como válvulas, motores, o niveles y parámetros de un proceso.
- Gestionar alarmas.
- Conexión con sistemas SCADA y MES.
- Conexión con controladores lógicos programables y sensores.
- Apagado de maquinaria.
- Aumento de frecuencia de producción.

Switch industrial

Son dispositivos utilizados en las redes industriales que permiten conectar y servir de puente entre redes. Para conectar las redes, el switch industrial aprende y

almacena las direcciones IP de los dispositivos que compongan las redes a las que se conecta.

Estos dispositivos disponen características de aislamiento eléctrico para resistir ante fluidos. Además, permite hacer conexiones Ethernet e incluso fibra óptica (Logicbus, s.f.).

Protocolos de comunicación

MODBUS

MODBUS es un protocolo de comunicación usado comúnmente en la automatización de procesos dado que puede transmitir datos a través de líneas seriales o vía Ethernet. Actualmente es un protocolo abierto, aunque la marca “MODBUS” haya sido registrada por Schneider Electric USA, Inc. (actual propietario de la marca Modicon) (Mostia, 2019).

El protocolo serie MODBUS funciona con el modelo maestro/esclavo; es decir, existe un maestro quién es el que controla las transacciones de datos con uno o varios esclavos que responden a las peticiones para leer o escribir datos en ellos. MODBUS TCP, o también denominado MODBUS TCP/IP, utiliza el modelo cliente/servidor.

La trama MODBUS se basa en una unidad de datos de aplicación (ADU). Esta ADU se forma mediante la dirección del esclavo o servidor, la unidad de datos de protocolo (PDU) y una comprobación de errores. La PDU contiene el código de función del mensaje y los datos del mensaje.

En una red serie MODBUS estándar, hay un maestro y hasta 247 esclavos, cada uno con un número identificador único. MODBUS TCP, en cambio se implementa en una red Ethernet y el intercambio de datos de un cliente se dirigen a un servidor mediante la dirección IP. Existe varias variedades de MODBUS, entre ellas se encuentran: RTU, ASCII, TCP/IP y UDP/IP.

Los modos de transmisión RTU y ASCII son de tipo serial y se diferencian en la forma en que el mensaje es codificado. En MODBUS RTU, los bytes se envían uno tras otro sin espacio entre ellos, además, por cada 8 bits de mensaje, se envía un bit de inicio, un bit de paridad y un bit de parada, con lo cual, se envía un total de 11 bits. Los mensajes en MODBUS RTU hacen el reconocimiento de errores a través de una suma de comprobación de redundancia cíclica (CRC) (MODICON Inc., 1996).

Por otro lado, MODBUS ASCII utiliza los caracteres ASCII para la transmisión de datos. Inicia cada mensaje con la codificación del carácter ASCII ":" y el finaliza con "/". Por cada 7 bits de mensaje se añade un bit de inicio, un bit de paridad y un bit de parada, con lo cual, se envía un total de 10 bits. Los mensajes en MODBUS ASCII hacen el reconocimiento de errores a través de una suma de control de redundancia longitudinal (LRC).

MODBUS TCP o TCP/IP es básicamente MODBUS RTU envuelto en un paquete Ethernet (IEEE 802.3), en lugar de direccionar el mensaje a través del identificador de esclavo, utiliza la dirección IP mediante el protocolo de transacción TCP/IP. Utiliza el puerto 502 para la comunicación.

DeviceNet

DeviceNet es una implementación del Protocolo Industrial Común (CIP) para redes de comunicación industrial. Fue desarrollado por Allen-Bradley en 1994, aunque su tecnología fue transferida a Open DeviceNet Vendor Association (ODVA) quien es la que brinda soporte al protocolo. Además, utiliza el protocolo Controller Area Network (CAN) para la capa 2 del modelo OSI (Weg, 2008).

DeviceNet utiliza el modelo productor/consumidor, lo que implica que, al enviar la información, esta se distribuye de forma simultánea a todos los receptores, quienes deciden leer o no el mensaje. La ventaja de este modelo es que provoca un mejor

aprovechamiento del ancho de banda y los tiempos de respuesta se ven reducidos. Una red DeviceNet puede tener hasta 64 dispositivos (Gómez, 2003).

DeviceNet define dos tipos de mensajes, los de entrada/salida (datos de proceso) y los explícitos (configuración y parametrización). También dispone de mecanismos para reconocer direcciones duplicadas y para aislar nodos en caso de que ocurra fallos críticos en la red.

Profibus

Profibus es una tecnología de bus de campo inteligente donde todos los dispositivos se acoplan a una línea central para comunicar información de manera eficiente (Real Time Automation, Inc, 2021). Las redes Profibus utilizan tres capas independientes del modelo OSI (1, 2 y 7). Existen varias versiones de Profibus los cuales se diferencian en los tipos de mensajería que manejan en la capa de aplicación. Profibus admite intercambio de datos cíclicos y acíclicos, diagnóstico, manejo de alarmas y mensajería isócrona.

La capa de enlace de datos combina el modelo maestro-esclavo y el paso de tokens. En una red maestro-esclavo, la secuencia en la transacción es igual al protocolo MODBUS RTU, siendo los maestros generalmente controladores, y los esclavos, sensores y actuadores. Por otro lado, en una red con paso de tokens, se pasa una señal de "token" entre los nodos, y solo el nodo con el token puede comunicarse.

Finalmente, Profibus, en su capa física, puede tener tres tipos de medios: cableado de par trenzado estándar (RS485), transmisión de fibra óptica para alcanzar velocidades mayores, y un sistema de seguridad mejorada llamado Manchester Bus Power o MBP, el cual se recomienda en situaciones en las que el entorno químico es propenso a explosiones.

Profibus FMS. Profibus Fieldbus Message Specification (FMS) es la versión inicial de este protocolo. Fue diseñado para transmitir mensajes complejos entre

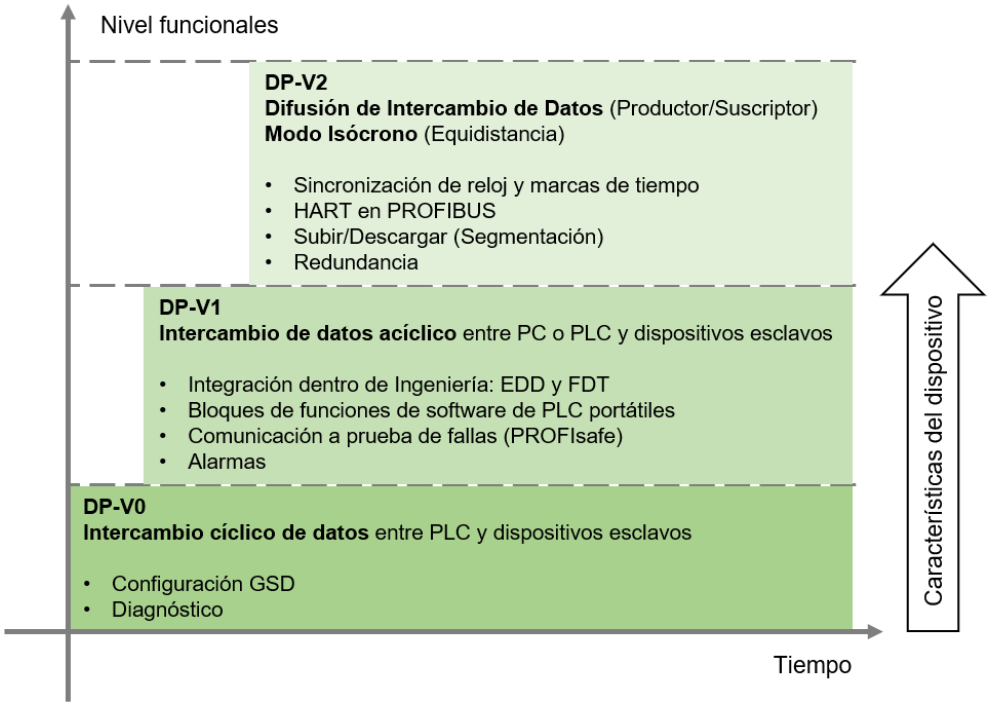
controladores programables y computadores. A pesar de que Profibus FMS cumplía con este propósito, no era lo suficientemente flexible para el envío de mensajes menos complejos en una red amplia.

Profibus DP. Profibus Decentralized Periphery (DP) es mucho más simple y rápido que la primera versión. Actualmente, Profibus DP se utiliza en la inmensa mayoría de los perfiles de aplicación.

Las funciones del protocolo de comunicación Profibus DP se distribuyen en tres niveles de rendimiento: DP-V0, DP-V1 y DP-V2 (PROFIBUS Nutzerorganisation e.V., 2016). En la Figura 4 se resume los servicios de comunicación que se engloban en cada uno de los niveles.

Figura 4

Protocolo Profibus DP, niveles de rendimiento



Nota. Adaptado de *PROFIBUS System Description*, por PROFIBUS Nutzerorganisation e.V., 2016.

La versión DP-V0 proporciona la función básica del protocolo de comunicación, en particular, la comunicación cíclica y el diagnóstico específico de dispositivo, módulo y canal para una rápida localización de fallas. La versión DP-V1 complementa DP-V0 con funciones para la comunicación acíclica, para funciones como parametrización, operación, monitorización y gestión de alarmas. La versión DP-V2 añade funciones que son necesarias para el control del tráfico. Estas incluyen funciones para la comunicación entre esclavos y sincronización de ciclos.

Los dispositivos Profibus se dividen en tres clases según sus funciones: maestro clase 1, maestro clase 2 y esclavo. El maestro clase 1 utiliza la comunicación cíclica para intercambiar datos de proceso con los esclavos. Los maestros clase 2 se utilizan la comunicación acíclica para configurar los parámetros del dispositivo. Por último, el esclavo es un nodo de comunicación pasivo que reacciona a las solicitudes del maestro enviando un mensaje de respuesta.

Profibus PA. Es un protocolo diseñado para la automatización de procesos, que se caracteriza por un funcionamiento intrínsecamente seguro, grandes longitudes de cable y alimentación de dispositivos a través del cable de bus. Profibus PA define las funciones y parámetros de los dispositivos de control de procesos, como transmisores, actuadores, válvulas y analizadores. Estas funciones y parámetros se utilizan para adaptar los dispositivos a las respectivas condiciones de aplicación y proceso.

Ethernet/IP

Ethernet/IP es un protocolo de capa de aplicación, dado que utiliza las especificaciones Ethernet tradicional para las demás capas, es decir, utiliza el protocolo de control de transporte (TCP), el protocolo de Internet (IP) y las tecnologías de señalización y acceso a medios de las tarjetas de interfaz Ethernet convencionales (Real Time Automation, Inc, 2021).

El hecho de funcionar en base a tecnologías de PC estándar significa que Ethernet/IP puede funcionar de la misma manera con todos los dispositivos Ethernet estándar disponibles en el mercado, como computadoras, dispositivos de comunicación, entre otros. Otra ventaja de basarse en una plataforma de tecnología estándar, es que asegura que este protocolo avanzará a medida que las tecnologías base avancen.

Ethernet/IP surge a partir de un estándar ampliamente utilizado en DeviceNet y ControlNet llamado Control and Information Protocol (CIP). Este estándar organiza como una colección de objetos a todos los dispositivos acoplados a la red. Además, define el acceso, el comportamiento del objeto y las extensiones que permiten acceder a dispositivos muy dispares utilizando un mecanismo común.

Todos los dispositivos en una red Ethernet/IP presentan sus datos a la red como una serie de valores llamados atributos agrupados con otros datos similares en conjuntos de atributos llamados “objetos”.

Hay dos tipos de mensajes que se transfieren entre un dispositivo de escáner Ethernet/IP (abre conexiones e inicia transferencias de datos) y dispositivos de adaptador Ethernet/IP (proporciona datos a los escáneres): mensajes explícitos (asíncronos, según sea necesario) y mensajes de entrada/salida (mensajes de datos que se transfieren continuamente).

EtherNet/IP proporciona un tiempo de respuesta mejorado y un mejor rendimiento que DeviceNet y ControlNet. Adicionalmente, conecta dispositivos desde el nivel de campo hasta el nivel de control y nivel empresarial con una interfaz de capa de aplicación consistente.

Profinet

Profinet es protocolo de comunicación basado en estándares internacionales diseñado para intercambiar datos entre controladores y dispositivos dentro de un

entorno de automatización. Apareció a principios de la década de 2000 y es la variante de Ethernet industrial más utilizado actualmente (Ayllon, 2021).

Profinet define la comunicación cíclica y acíclica entre componentes, incluyendo diagnósticos, seguridad funcional, alarmas e información adicional. Utiliza como medio de físico, conectores y cables de Ethernet estándar, permitiendo coexistir dentro de la misma infraestructura de red otros protocolos de Ethernet; de hecho, Profinet puede emplear otros protocolos para complementar la red, como OPC UA, SNMP, MQTT o HTTP.

Profinet clasifica los dispositivos en tres tipos: controladores IO, dispositivos IO y supervisores IO. Los controladores IO son dispositivos que ejecutan un programa de automatización. Los controladores IO son funcionalmente similares a un maestro de clase 1 de Profibus, e intercambian datos con los dispositivos IO (sensores o actuadores). En términos de Profibus, los dispositivos IO son similares a los esclavos. Por último, los supervisores IO son dispositivos de puesta en marcha, monitorización o análisis de diagnóstico como HMIs, PCs, entre otros. Estos dispositivos son similares a los Maestros Profibus de clase 2 (Real Time Automation, Inc., 2021).

Para garantizar un rendimiento adecuado, PROFINET suministra datos a través de los siguientes canales de comunicación:

TCP/IP (o UDP/IP). Profinet puede emplear la comunicación TCP/IP o UDP/IP para las tareas que no son de tiempo crítico, como acciones de configuración o parametrización. Debido a la latencia y la fluctuación añadidas asociadas a la comunicación basada en IP, este método no es adecuado para las tareas en las que el tiempo es crítico.

Profinet Real-Time (RT). Emplea un canal de tiempo real para suministrar datos de forma rápida y determinista (entregar los mensajes exactamente en el momento en que se esperan) para las aplicaciones de tiempo crítico. Las tramas Ethernet estándar

tienen un campo para especificar tipo de protocolo utilizado (EtherType). La comunicación Profinet RT está configurada para tener un EtherType de 0×8892, haciendo que la trama se dirija a la aplicación Profinet directamente, mejorando la velocidad de comunicación significativamente.

El rendimiento global dependerá del diseño de la red, pero generalmente, puede lograr tiempos de ciclo de 250 µs a 512 ms. Además, no necesita un hardware o una configuración especial para utilizar el mecanismo de tiempo real Profinet.

Profinet Isochronous Real-Time (IRT). Profinet IRT crea reglas específicas para el tráfico Profinet y mejora las utilizadas en la conmutación Ethernet. Estos cambios eliminan los retrasos de datos variables y permite una comunicación determinista alcanzando tiempos de ciclo de entre 31,25 µs y 1 µs. Profinet IRT es opcional y sólo se requiere en aplicaciones específicas de alto rendimiento, como máquinas de embalaje y prensas de impresión.

Red sensible al tiempo (TSN). TSN hace referencia a un conjunto de normas IEEE 802 que hacen que Ethernet sea determinista por defecto. Esta nueva tecnología se asienta en la capa 2 del modelo OSI, añadiendo definiciones para garantizar el determinismo y el rendimiento en las redes Ethernet (PI North America, 2021). Busca garantizar una latencia limitada para las comunicaciones de tiempo crítico considerando el aumento en el tráfico de la red que conllevan las tendencias industriales actuales, como la Industria 4.0 y el Internet Industrial de las Cosas (IIOT).

Dentro de la automatización de procesos, la automatización de fábricas y el control de movimiento, Profinet sobre TSN es utilizado por PLCs, sensores remotos, entradas/salidas remotas, controladores de movimiento y accionamiento, así como por controladores numéricos computarizados. Puede manejar hasta 1024 dispositivos y lograr tiempos de ciclo de 31,25 µs.

Protocolo S7

S7 es un protocolo propietario de Siemens de capa de aplicación que implementa funciones para intercambiar datos entre PLCs de la serie Simatic S7 y otros dispositivos que admitan esta comunicación. Se puede utilizar como interfaces eléctricas Ethernet Industrial, Profibus o MPI. Tanto ISO como TCP/IP son compatibles como protocolos de transporte en el caso de Ethernet Industrial (Schneider Electric, 2021).

Este protocolo tiene las ventajas de ser aplicable a todas las zonas de datos S7, permitir la transferencia de datos desde 1 byte hasta 64 kbytes dependiendo del servicio utilizado (PUT/GET, BSEND/BRCV o USEND/URCV) y de la CPU S7 utilizada, y generar una baja carga del procesador y del bus en la transmisión de grandes cantidades de datos (Siemens, 2019).

Puede utilizar los modelos maestro-esclavo o cliente-servidor, donde la CPU (maestro/cliente) envía solicitudes S7 al dispositivo de campo (esclavo/servidor). Estas solicitudes se utilizan para consultar o enviar datos al dispositivo o emitir ciertos comandos. En los PLCs de gama alta se implementa una función denominada entrada/salida de datos cíclicos, que se asemeja al modelo productor/consumidor (Miru, 2017).

El protocolo S7 está orientado a funciones/comandos, lo que significa que una transmisión consta de una solicitud S7 y una respuesta adecuada. El número de la transmisión en paralelo y la longitud máxima de una PDU se negocia durante el establecimiento de la conexión. Si el tamaño de un comando no cabe en una PDU, debe dividirse en múltiples PDUs. La PDU consta de tres partes: encabezado, parámetros y datos (opcional).

Capítulo III. Diseño

Esta sección contiene todos los componentes necesarios para diseñar el laboratorio remoto, partiendo por describir las funcionalidades que ofrece al usuario tanto para el aprendizaje del estudiante como en el manejo de los equipos para el docente. Luego se explica cómo funciona el sistema principal relacionado al manejo de bombas, variables del sistema y al análisis matemático. Después se indica todos los equipos utilizados para la elaboración de la maqueta, su utilidad y qué tecnologías son empleadas para integrarles y comunicarse entre ellos. Se elabora la lógica de funcionamiento del PLC mediante diagramas GRAFCET, además de enlistar todos los recursos que se utilizan del mismo. A continuación, se muestra la distribución de los elementos dentro de las interfaces de usuario, y herramientas de programación que se utilizaron para el desarrollo de la aplicación web, a fin de que la navegación sea intuitiva y que el estudiante logre sus objetivos de aprendizaje. Por último, se detalla la estrategia utilizada para poner a disposición la aplicación web de forma remota, al igual que el intercambio de información entre la maqueta y el cliente.

Funcionalidades del laboratorio remoto

Las funciones del laboratorio remoto a implementar con el presente proyecto tienen como objetivo primordial que el usuario pueda realizar prácticas de laboratorio a distancia, manejando un sistema que está implementado en una maqueta y situado en el laboratorio de Robótica de la Universidad de las Fuerzas Armadas ESPE sede Matriz; es decir, desde cualquier lugar con la única condición de que tenga acceso a internet. Tomando en cuenta que el laboratorio remoto resulta ser una aplicación web, las funcionalidades del mismo se clasifican bajo los siguientes apartados.

- **Registro de usuario**

Se tiene la opción de registrar un nuevo usuario con nombres, apellidos, usuario y contraseña. Esta identificación sirve para acceder con un nombre único al laboratorio remoto y evitar colapsos en los agendamientos del laboratorio.

- **Inicio de sesión de usuario**

Una vez que se tenga registrado un nuevo usuario, la interfaz dispone de una validación de ingreso. Aquí el usuario digita su nombre de identificación y su contraseña para acceder al entorno del laboratorio remoto.

- **Agendamiento de una práctica**

Dentro del entorno del laboratorio remoto, se dispone de una sección dentro de la cual el usuario puede agendar una práctica en un horario específico para que nadie más pueda acceder al laboratorio en esa fecha y hora. De esta forma se evita que dos estudiantes puedan interactuar al mismo tiempo con la maqueta, lo cual podría provocar algún fallo en los equipos.

El estudiante puede agendar una práctica con una semana de anticipación; además, cada día tiene 4 franjas horarias; la primera de 7:15 a 9:15, la segunda comprendida entre las 9:30 hasta 11:30, la tercera desde las 12:00 hasta las 14:00 y la última de 14:15 a 16:15.

- **Selección de prácticas de laboratorio**

Aparte de reservar una fecha y horario específico, el laboratorio remoto posee 3 prácticas de laboratorio distintas las cuales pueden ser escogidas por el usuario en su agendamiento. Para facilitar la reserva, la interfaz tiene una sección en la cual se describen las tres prácticas de laboratorio; las mismas que se indican a continuación:

1. Configuración del variador de frecuencia y puesta en marcha del motor

El laboratorio remoto permite al usuario configurar los parámetros básicos y relevantes de un variador de frecuencia, que son: los tiempos de aceleración y desaceleración, la referencia o setpoint de frecuencia en Hz y las especificaciones nominales del motor. Después, permite inicializar y poner en marcha al motor. Durante el funcionamiento del motor, se puede visualizar las gráficas de las características de funcionamiento del mismo, es decir, la corriente, el voltaje, la potencia y el torque. La interfaz muestra al variador y al motor a través de una cámara en tiempo real. También el usuario puede parar el motor en cualquier momento.

2. Manejo de bombas eléctricas con demanda fija

Esta práctica está enfocada en el sistema del manejo de bombas eléctricas, el mismo que está implementado en la maqueta. Se permite visualizar en tiempo real el funcionamiento del sistema a través de una cámara, así como también controlarlo de forma remota. Se puede escoger un controlador PI o PID, sintonizar sus parámetros de funcionamiento (Kp, Ti, Td) y dar arranque al sistema. En la interfaz se puede ver gráficas e indicadores de las variables de interés del sistema, de tal manera que se pueda analizar su comportamiento. Cabe indicar que en esta práctica la demanda es fija y no cambia.

3. Manejo de bombas eléctricas con demanda variable

Esta práctica es un nivel más avanzado que la anterior porque tiene la opción de modificar la demanda del sistema; de esta forma, el usuario puede analizar el comportamiento ante estas variaciones. Todas las funcionalidades y opciones que proporciona la práctica 2 son las mismas en cuanto a selección del controlador, sintonización de sus parámetros, puesta en marcha del sistema y visualización de gráficas que permitan el análisis de las variables de interés.

- **Cambio de horario de la práctica reservada**

El usuario cuando se encuentre dentro del entorno del laboratorio remoto tiene la opción de cambiar la fecha, la hora y el tipo de práctica que reservó anteriormente, dejando disponible la anterior para que otros usuarios puedan acceder.

- **Almacenamiento en base de datos**

Los datos del usuario, la hora, fecha y el tipo de práctica reservado por el usuario son reservados en una base de datos con el objetivo de no colapsar al laboratorio remoto.

- **Visualización de prácticas en tiempo real**

La maqueta integrada con todos los componentes está en red con una cámara web lo que permite que la interfaz remota muestre el funcionamiento del sistema en tiempo real al usuario. Esto permite que la experiencia, retroalimentación y aprendizaje sea como que el usuario realiza la práctica de forma presencial.

- **Exportar las gráficas**

Cada práctica de laboratorio presenta gráficas en tiempo real de sus variables más importantes, como, por ejemplo, de velocidad, corriente, potencia o voltaje del motor. Los estudiantes pueden capturar estas gráficas en cualquier momento con el objetivo de tener recursos para futuros informes o análisis.

- **Anuncios por parte del profesor**

El profesor tiene acceso a una cuenta diferente a la de los estudiantes en la cual puede publicar anuncios para las tres prácticas de laboratorio descritas anteriormente. Estos mensajes pueden ser avisos, recomendaciones, indicaciones, entre otros.

Descripción del sistema

El sistema para el laboratorio remoto está enfocado en el manejo de 4 bombas para mantener una presión estable de $6\ bar$ independientemente de la demanda que se

requiera. Para esto es necesario definir un esquema y una técnica de control para que, como resultado del mismo, se determine cuántas bombas funcionen al máximo de su capacidad y además la bomba que permita satisfacer la demanda requerida. La bomba mencionada que no funciona en su máxima capacidad, está representada a través de un motor trifásico.

El tipo de sistema es en lazo cerrado para que exista una realimentación de la señal de salida y que en definitiva permita el control de la presión del sistema independientemente de la demanda requerida.

La variable que es controlada es la presión del sistema, para lo cual el usuario define el tipo de controlador a utilizarse, teniendo como opciones el controlador Proporcional Integral (PI) y el Proporcional Integral Derivativo (PID).

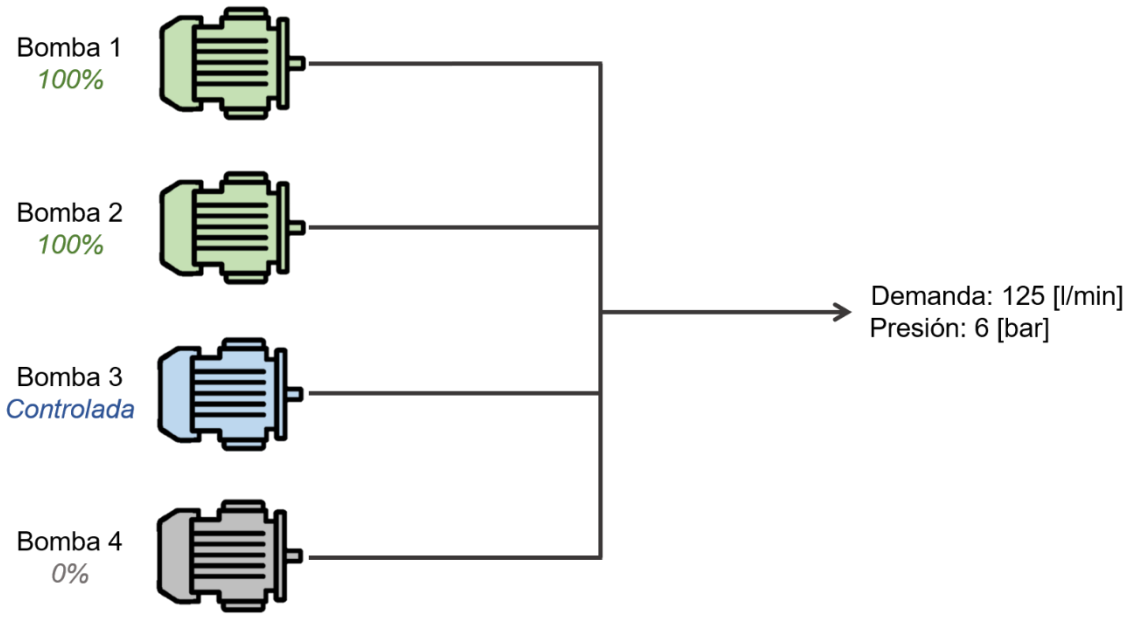
Al tratarse de un sistema enfocado en el manejo de bombas, y al no disponer uno de estos componentes de forma física (la bomba está representada por un motor eléctrico), se define las especificaciones técnicas basado en bombas comerciales. Por consiguiente, las características de una bomba eléctrica definidas para este sistema son:

- Potencia: 3/4 HP
- Caudal máximo: 50 l/min

Sin embargo, el sistema total consta de 4 bombas eléctricas, entonces el caudal máximo del mismo es de 200 l/min. Después, con esta información se determina cuántas bombas funcionan al 100% de su capacidad y cuál bomba debe ajustar su velocidad para cumplir con la demanda y mantener una presión estable.

Figura 5

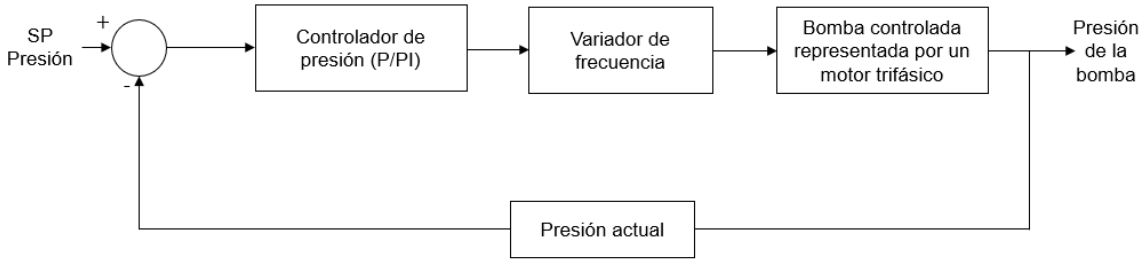
Activación de las bombas en caso de demanda = 125 [l/min]



Por ejemplo y como se observa en la Figura 5 si el sistema requiere de una demanda de 125 l/min, las bombas eléctricas que funcionarían a su máxima capacidad serían la primera y la segunda; es decir, con una demanda de 50 l/min y una presión de 6 bar. Por otro lado, la tercera bomba, únicamente tendría una demanda de 25 l/min y tendría que llegar a una presión de 6 bar a través de un controlador que es descrito gráficamente en la Figura 6.

Figura 6

Diagrama que describe el control de presión de la bomba que no funciona al 100% de su capacidad



Parámetros del sistema

Acorde al diagrama de la Figura 6, es importante definir todos los parámetros asociados al sistema tales como las variables controladas, las variables manipuladas, sus unidades y límites.

Variable controlada

La variable a controlar es la presión cuya referencia puede tomar valores de 0 a 10 bar. Se considera este valor porque una tubería de servicio común debería soportar una presión máxima de trabajo de al menos 15 kg/cm^2 que es equivalente a 14.71 bar (Válvulas Arco, 2019).

No obstante, y como se describió anteriormente, a priori se busca que el sistema mantenga una presión estable de 6 bar; por lo tanto, este valor resulta idóneo para esta aplicación.

Variable manipulada

El sistema como tal consiste en el manejo de bombas eléctricas; sin embargo, al estar representada a través de una maqueta, la bomba eléctrica controlada es reemplazada por un motor trifásico. De esta forma, la variable manipulada es la velocidad del motor. La variable manipulada siempre tiene que ver con el actuador del sistema, en este caso el actuador resulta ser el variador de frecuencia que está conectado al motor trifásico. El valor máximo que puede tomar la velocidad es de 60 Hz debido a que la velocidad máxima del motor de la maqueta es de 1725 rpm.

En la Tabla 1 se resumen los parámetros descritos para las variables del sistema.

Tabla 1

Variables del sistema

	Variable	Unidades	Límites
Controlada	Presión	bar	$0 - 10$
Manipulada	Velocidad	Hz	$0 - 60$

Análisis matemático para el cálculo de la presión en la bomba controlada

Basado en el diagrama de bloques mostrado en la Figura 6, es necesario obtener la presión de la bomba para establecer el lazo de control; sin embargo, dado que se tiene una maqueta que representa el funcionamiento de las bombas, pero no se dispone de una red de tuberías por donde el fluido sea transportado y sensado, es necesario obtener esta magnitud de forma matemática.

El objetivo de una bomba es transferir energía a un fluido para desplazarlo dentro de una red de tuberías. La fórmula mostrada en (1) calcula la potencia BHP proporcionada al fluido en hp, a partir de la gravedad específica SG, el caudal Q expresado en l/min, la presión P indicada en bares y el rendimiento total de la bomba η, el cual es la relación entre el rendimiento hidráulico η_h, el rendimiento mecánico η_m y el rendimiento eléctrico η_e (Gargil, S.L., 2021).

$$BHP = \frac{SG \cdot Q \cdot P}{450 \cdot \eta} \tag{1}$$

Si de la ecuación (1) despejamos la presión, obtenemos una expresión en función del caudal que fluye a través de la bomba y la potencia BHP. Basado en la Figura 5, la primera variable se obtiene restando el caudal de las bombas que funcionan al 100% de la demanda solicitada. En el caso de la potencia BHP, ciertamente no es posible obtenerla dado que no se está trabajando con ningún fluido; sin embargo, se puede aproximar utilizando la potencia eléctrica del motor, valor que se puede obtener

desde el variador de velocidad. Con respecto al caudal, se utiliza la demanda necesaria de dicha bomba para cubrir con la demanda total del sistema. Por lo tanto, con la ecuación (2) obtenemos la presión de la bomba siendo Pot la potencia eléctrica del motor en hp y Q_d la demanda necesaria de la bomba controlada en l/min.

$$P = \frac{450 \cdot \eta \cdot Pot}{SG \cdot Q_d} \tag{2}$$

En cuanto al caudal, normalmente sería medido utilizando algún sensor sea tipo ultrasónico, inductivo, vortex, entre otros (ifm electronic, 2021). En este caso, se va a aproximar el caudal de salida a partir de la cilindrada de la bomba. La cilindrada consiste en el volumen de fluido desplazado en cada revolución que realiza el eje de la bomba. El valor de la cilindrada depende del tipo de bomba que se disponga y, según ello, las dimensiones de los componentes y el cuerpo de la bomba. Existe también otra forma de aproximar el valor de la cilindrada basado en el caudal máximo y velocidad máxima que alcanza el eje del motor (Hidráulica Hidraoil, 2021). La fórmula (3) permite obtener el caudal aproximado Q_{ap} en función de la cilindrada V_B, la velocidad de giro del motor n, y el rendimiento volumétrico η_v que consiste en la relación entre el caudal real y el teórico que produce una bomba. Estas pérdidas se producen por diversos factores tales como la viscosidad del fluido, la presión de trabajo, la temperatura, entre otros; además, generalmente está entre el 80% 95%.

$$Q_{ap} = \frac{V_B \cdot n \cdot \eta_v}{1000} \tag{3}$$

Si de esta fórmula despejamos la cilindrada y reemplazamos los términos de caudal y velocidad del motor por sus condiciones máximas tendremos una expresión la cual nos permite calcular la cilindrada a partir de condiciones nominales definidas para el sistema, tales como caudal máximo del sistema (200 l/min – 50 l/min por cada bomba) y velocidad máxima del motor utilizado en la maqueta (1725 rpm).

$$V_B = \frac{1000 \cdot Q_{max}}{n_{max} \cdot \eta_v} \quad (4)$$

Evaluando en (4) las condiciones nominales del laboratorio remoto y suponiendo un rendimiento volumétrico del 95% tenemos que la cilindrada es de 30.51 ml/rev.

Conociendo la cilindrada, se puede calcular el caudal utilizando la fórmula (3), únicamente se necesita conocer la velocidad de giro del motor en la maqueta, valor el cual es posible conocer a partir del variador.

Equipos y componentes de la maqueta

Para la elaboración de la maqueta se dispone de los equipos necesarios para emular el proceso industrial a fin de que el estudiante pueda experimentar con el manejo y control de equipos industriales, y pueda observar y monitorear el rendimiento, tanto de forma local como remota. A continuación, se indica todos los equipos y componentes utilizados en la elaboración de la maqueta y su utilidad dentro del laboratorio remoto.

PLC Siemens S7-1200

Para controlar el proceso enfocado en el manejo de bombas se optó por utilizar un controlador de gama media debido a su flexibilidad con respecto a la programación que está basada en diagrama *ladder*. Además, el software de Siemens, TIA PORTAL, incluye bloques de controladores PID los cuales fueron implementados para el control del proceso emulado en la maqueta.

El CPU de este PLC es el 1212C con número de referencia 6ES7 212-1BE40-0XB0. Las especificaciones técnicas de este equipo se detallan en la Tabla 2.

Tabla 2

Especificaciones técnicas del PLC Siemens S7-1200 1212C 6ES7 212-1BE40-0XB0

Especificación	Descripción
Versión de firmware	V4.2
Fuente de alimentación	120/240V AC
Entradas digitales	#8 24V DC
Entradas analógicas	#2 entradas de voltaje de 0 a +10V
Salidas digitales	6 salidas de relé 2A
Protocolos de comunicación	Soporta PROFINET IO
Protocolos (Ethernet)	• TCP/IP • DHCP • SNMP • DCP • LLDP
Protocolos adicionales	MODBUS
Número de contadores	#6
Controlador PID	Si

Nota. Adaptado de *Datasheet 6ES7212-1BE40-0XB0*, por Siemens, 2021, Automation24, https://media.automation24.com/datasheet/se/6ES72121BE400XB0_en.pdf

De forma específica, el controlador S7-1200 permite controlar la demanda y presión del sistema. Por lo tanto, en base a una demanda requerida, el controlador indica a través de sus salidas cuantas bombas funcionan al 100% y la velocidad de la bomba a controlar, para mantener una presión estable.

En la Figura 7 se muestra el autómata lógico programable utilizado.

Figura 7

PLC Siemens S7-1200 CPU 1212C AC/DC/RLY

Nota. Tomado de *SIMATIC S7-1200 CPUs,* por Siemens, 2021, https://new.siemens.com/in/en/products/automation/systems/industrial/plc/s7-1200.html

Panel de operador Schneider Magelis HMISTU855/HMIS5T

La manipulación de los parámetros del sistema como la referencia en la demanda y las ganancias del PID de presión, el monitoreo respecto al funcionamiento del motor y la simulación de las condiciones del sistema, se realiza de forma local a través de un panel de operador, en concreto el modelo Magelis HMI STU855 de la marca Schneider Electric. Esta terminal táctil tiene un tamaño de 5.7 pulgadas con 65536 colores y 16 niveles de brillo. Adicionalmente, se cuenta con un panel posterior HMIS5T donde se encuentran los puertos de comunicación y comandos. Las especificaciones técnicas del panel de operador se encuentran en la Tabla 3. En las Figuras Figura 8 y Figura 9 se muestran la terminal táctil HMISTU855 y el panel posterior HMIS5T respectivamente.

Tabla 3

Especificaciones técnicas del panel de operador Schneider Magelis HMI STU855

Especificación	Descripción
Sistema operativo	Harmony
Procesador	ARM9
Tamaño de panel	5.7 pulgadas
Resolución	320 x 240 pixeles
Panel táctil	Análogo
Voltaje de alimentación	24V DC
Memoria	16 MB de aplicación
	64 kB para datos de respaldo
Puertos	USB 2.0 tipo A
	USB 2.0 tipo mini B
	COM1
	RJ45
	RS232C/RS485

Nota. Adaptado de *Ficha técnica Magelis STU 5,7" Color touch 22mm - HMISTU855*, por Schneider Electric, 2021, https://www.se.com/ar/es/product/download-pdf/HMISTU855.

Figura 8

Panel de operador Schneider Magelis HMI STU855

Nota. Tomado de *Ficha técnica Magelis STU 5,7" Color touch 22mm - HMISTU855*, por Schneider Electric, 2021, https://www.se.com/ar/es/product/download-pdf/HMISTU855.

Figura 9

Panel posterior HMIS5T

Nota. Tomado de *Ficha técnica Magelis STU módulo posterior - HMIS5T*, por Schneider Electric, 2021, https://www.se.com/ar/es/product/download-pdf/HMIS5T.

Variador de frecuencia Schneider Altivar ATV630U15M3

Es el equipo que permite el control del motor de corriente alterna utilizado para emular a una bomba eléctrica. Está integrado con el PLC Siemens S7-1200 para manejar y controlar la velocidad del motor. A pesar de disponer un controlador PID integrado dentro de sus funcionalidades, para el presente trabajo no se utiliza este recurso. El control viene directamente del autómata S7-1200 descrito anteriormente.

Las especificaciones técnicas del variador se describen en la Tabla 4 y en la Figura 10 se observa el variador de velocidad utilizado en la maqueta.

Tabla 4

Especificaciones técnicas del variador de frecuencia Schneider Altivar ATV630U15M3

Especificación	Descripción
Frecuencia de alimentación	50 a 60Hz
Voltaje de alimentación	200 a 240V
Potencia del motor en kW	1.5kW – carga normal
	0.75kW – carga pesada
Potencia del motor en HP	2Hp – carga normal
	1Hp – carga pesada
Protocolos de comunicación	Serie MODBUS
	Ethernet
	MODBUS TCP

Figura 10

Variador de frecuencia Schneider ATV630U15M3

Nota. Tomado de *Variador de velocidad, Altivar Process ATV600, ATV630, 1.5kW/2 hp, 200...240 V, IP21/UL tipo 1,* por Schneider Electric, 2021, https://www.se.com/co/es/product/ATV630U15M3/variador-de-velocidad-altivar-process-atv600-atv630-1-5kw-2-hp-200-240-v-ip21-ul-tipo-1/

Motor Baldor CM3538

Este motor es utilizado para emular el funcionamiento de la bomba controlada del sistema para cumplir con la demanda indicada. Sus especificaciones técnicas se detallan en la Tabla 5, siendo necesario que la velocidad nominal sea similar a la de la bomba escogida para el sistema, con el fin de que la salida porcentual de los bloques PID coincida con el rango de velocidades y, por ende, el caudal que entrega. El variador

Altivar ATV630U15M3 permitirá controlar el giro del motor. En la Figura 11 se puede observar el motor utilizado para la maqueta del laboratorio remoto.

Tabla 5

Especificaciones técnicas del motor Baldor CM3538

Especificación	Descripción
Potencia	0.5 hp
Voltaje de alimentación	230/460 V AC, 60 Hz
Corriente nominal	2 A a 230 V
	1 A a 460 V
Velocidad nominal	1725 rpm
Eficiencia	74%

Figura 11

Motor Baldor CM3538 1/2 hp

Nota. Tomado de *Baldor Electric General Purpose Motor*, por W.W. Grainger, Inc., 2021, https://www.grainger.com/product/BALDOR-ELECTRIC-General-Purpose-Motor-29UX64.

Fuente de alimentación Schneider ABL8REM24030

Es de mucha importancia, ya que este equipo realiza la conversión de voltaje de fase de 110V AC a 24V DC para alimentar al panel de operador HMI y al switch industrial utilizados en la maqueta.

Las especificaciones técnicas de este equipo se detallan en la Tabla 6 y en la Figura 12 se muestra la fuente de alimentación descrita anteriormente.

Tabla 6

Especificaciones técnicas de la fuente de alimentación Schneider ABL8REM24030

Especificación	Descripción
Voltaje nominal de entrada	100…240V AC fase a fase, terminales: L1-L2 100…240V AC una fase, terminales: N-L1 110…220V DC
Potencia nominal	72W
Voltaje de salida	24V DC
Corriente de salida	3A

Figura 12

Fuente de alimentación Schneider ABL8REM24030

Switch Schneider TCSESU053FN0

Se utiliza para establecer todos los enlaces que utilizan el puerto RJ45 con protocolo de comunicación Ethernet TCP/IP. En la Figura 13 se muestra el switch Schneider TCSESU053FN0 donde se puede notar que cuenta con 5 puertos disponibles para establecer la red. Las características relevantes del switch se indican en la Tabla 7.

Tabla 7

Especificaciones técnicas del switch Schneider TCSESU053FN0

Especificación	Descripción
Protocolo de puerto de comunicación	Ethernet TCP/IP
Puerto Ethernet	10/100BASE-TX - 5 cable cobre
Tipo de conexión integrada	RJ45 blindado cable cobre
Medio de soporte de transmisión	Cable pare trenzado blindado CAT 5E para cable cobre
Tensión de alimentación	12 – 24 V DC

Nota. Adaptado de *Ficha técnica Switch Ethernet TCSESU053FN0*, por Schneider Electric, 2021, https://www.se.com/cl/es/product/download-pdf/TCSESU053FN0.

Figura 13

Switch Schneider TCSESU053FN0

Nota. Tomado de *Ficha técnica Switch Ethernet TCSESU053FN0*, por Schneider Electric, 2021, https://www.se.com/cl/es/product/download-pdf/TCSESU053FN0.

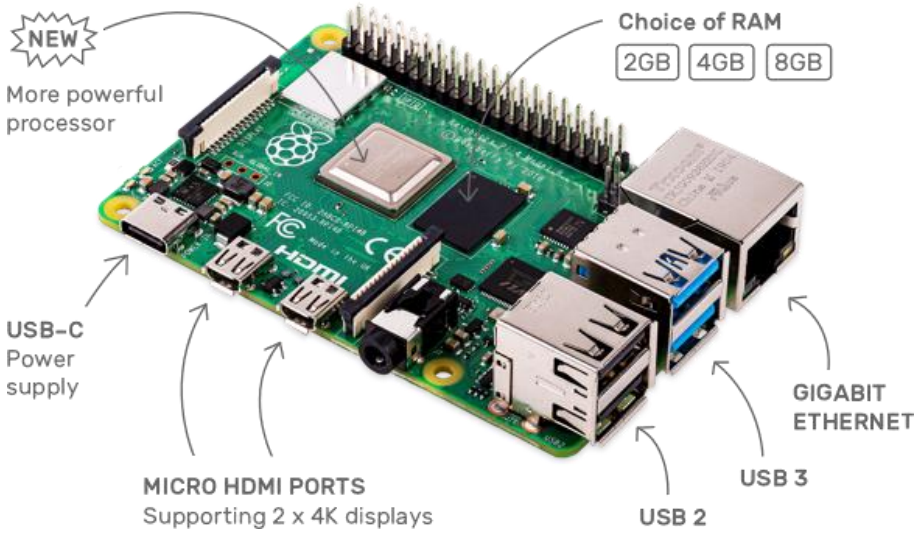
Raspberry Pi 4

En el presente trabajo, este computador va a permitir el diseño de la aplicación web, la adquisición y envío de datos al controlador PLC Siemens S7-1200, y en consecuencia el enlace remoto para que un cliente pueda acceder desde la web hacia la maqueta implementada de forma física.

Aparte de ser un computador muy barato que corre a partir de una distribución de Linux, también proporciona un conjunto de entrada/salida de propósito general (GPIO), lo que permite interactuar componentes electrónicos como sensores o actuadores y explorar en el Internet de las cosas (IoT). En la Figura 14 se observa el modelo 4 de la Raspberry Pi.

Figura 14

Tarjeta Raspberry Pi 4



Nota. Tomado de *Raspberry Pi 4*, por Raspberry Pi Foundation, 2011, https://www.raspberrypi.com/products/raspberry-pi-4-model-b.

La velocidad y el rendimiento de la Raspberry Pi 4, la última tarjeta de desarrollo lanzada, es capaz de brindar experiencia de escritorio similar a un ordenador convencional, es decir, permite editar documentos, navegar por la web con un montón de pestañas abiertas, realizar cálculos estadísticos, entre otros, de forma fluida, pero en una máquina más pequeña, lo que permite un gasto energético más eficiente (Raspberry Pi Foundation, 2021).

La Tabla 8 resume las especificaciones técnicas más importantes de este componente.

Tabla 8

Especificaciones técnicas de la tarjeta Raspberry Pi 4

Especificaciones	
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memoria RAM	8GB LPDDR4-3200 SDRAM
Conectividad	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet
Puertos USB	2 puertos USB 3.0 2 puertos USB 2.0
GPIO	40 pines estándar de Raspberry Pi (totalmente compatible con placas anteriores)
Puertos HDMI	2 puertos micro-HDMI (hasta 4kp60 soportados)
Puertos multimedia	• Puerto de pantalla MIPI DSI de 2 carriles • Puerto de cámara MIPI CSI de 2 carriles • Puerto de audio estéreo de 4 polos y de vídeo compuesto
Alimentación	5V DC a través del conector USB-C o cabezal GPIO (mínimo 3A)
Temperatura de funcionamiento	0 - 50 °C ambiente

Nota. Adaptado de *"Raspberry Pi 4 Tech Specs"* por Raspberry Pi Foundation, 2021, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Cámara

Para capturar en video el funcionamiento de la maqueta y mostrarlo en la interfaz del estudiante durante la práctica, se utilizó la webcam AGILER AGI-4188. Esta webcam es capaz de grabar con resolución de 1080P a una velocidad de 30 fps, tiene micrófono incorporado, se conecta a través de USB 2.0 y es compatible con Windows XP y superior, Linux 2.6.24 y superior y Mac OS 10.6 y superior (Bocancho Sanchez, 2021). En la Figura 15 se observa la cámara utilizada.

Figura 15

Webcam AGILER AGI-4188

Nota. Tomado de *Camara Web Agiler AGI 4188*, por Bocancho Sanchez, I. E., 2021, https://www.ecuatech.com/shop/product/cwagiler4188-agiler-agi4188-camara-web-agiler-agi-4188-1080p-full-hd-microfono-incluido-1920x1080-usb-2-0-ideal-clases-en-linea-y-zoom-8045?category=31.

Pulsadores y luces

Se elabora un pequeño tablero de control con luces piloto y pulsadores para habilitar la maqueta, observar el estado de funcionamiento y como medida de seguridad ante posibles fallos. Los pulsadores funcionan con 24 V DC para que la señal pueda ir a las entradas del PLC, mientras que las luces piloto se energizan con 110 V AC. En las

Figuras Figura 16 y Figura 17 se muestran los componentes utilizados en construcción de dicho tablero.

Figura 16

Luces piloto AD1622DS-110V

Nota. Tomado de *110v Ac Indicator Light AD1622DS White Colour*, por Circuit.pk., 2021, http://circuit.pk/product/3018.

Figura 17

Pulsador Talmeminque ZB2-BE102C

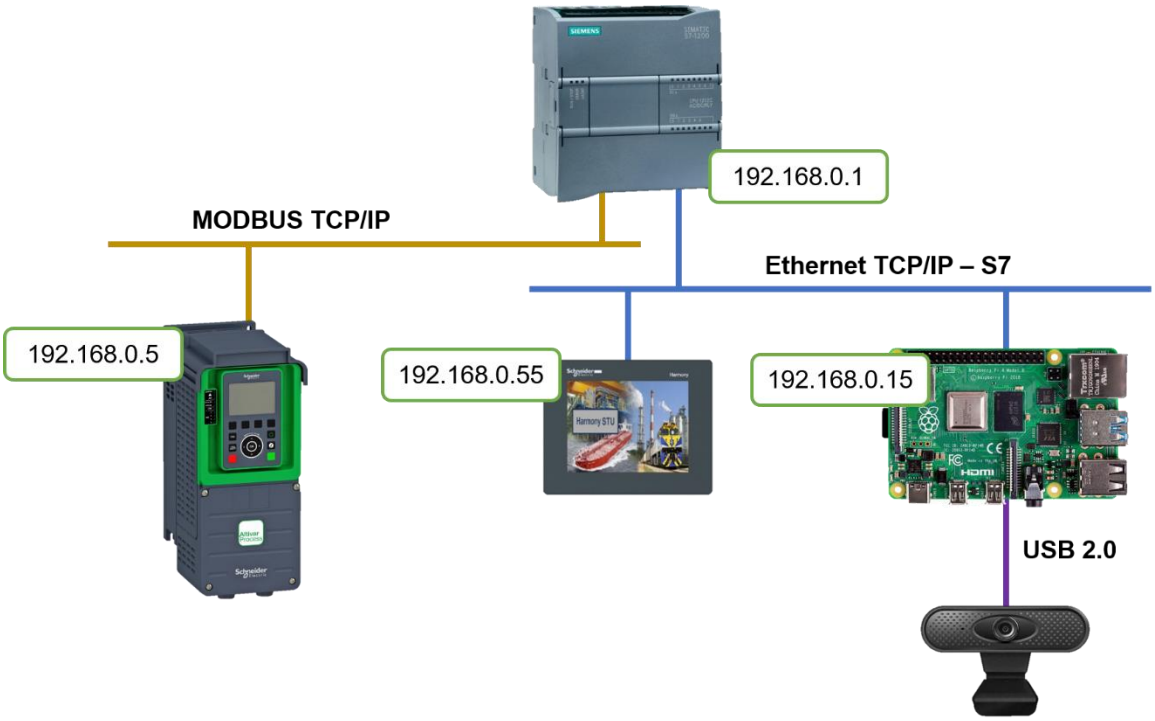
Nota. Tomado de *Suiche pulsador NC 22mm Rojo*, por Electrónica I+D, 2021, https://www.didacticaselectronicas.com/index.php/suiches-y-conectores/suiches/pulsadores/suiche-pulsador-nc-22mm-rojo-suiches-switches-botones-pulsadores-moment%C3%A1neos-pulsadores-industriales-normalmente-cerrados-de-22mm-rojo-detail.

Red de integración de los componentes de la maqueta y visualización

Al igual que en un proceso industrial, es muy importante establecer una red de comunicación que garantice el intercambio de información entre los componentes de la maqueta, y que el estudiante pueda experimentar el funcionamiento del sistema de la manera más real posible. En la Figura 18 se muestra el diagrama de red con los componentes principales de la maqueta, las direcciones IP de cada uno y los protocolos o medios definidos para la comunicación entre ellos.

Figura 18

Diagrama de red de los componentes de la maqueta y visualización



Conexión entre PLC y HMI

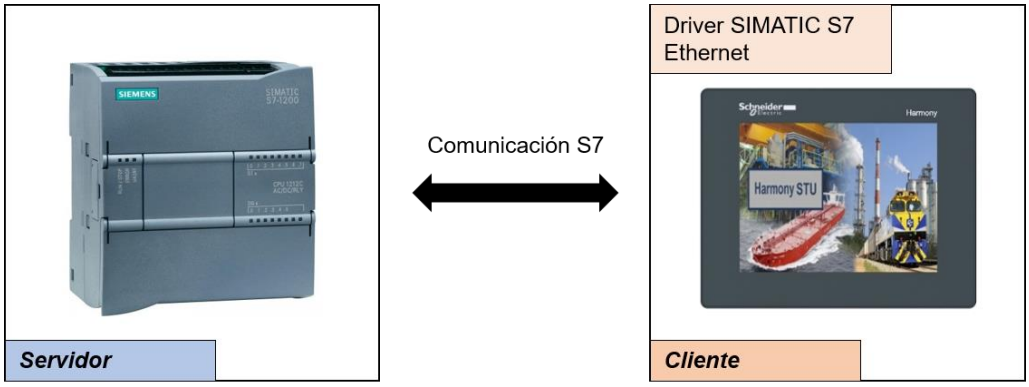
El enlace entre el PLC Siemens S7-1200 1212C AC/DC/RLY y el panel de operador Schneider Magelis HMI STU855 se da mediante el protocolo de comunicación S7 a través de Ethernet Industrial.

La CPU del autómata programable es la SIMATIC S7 el cual incluye servicios de comunicación integrados. El HMI de la marca Schneider es compatible con la comunicación S7 Ethernet y dentro de su software Vijeo Designer consta el controlador lógico S7-1200 CPU 1212C; por lo tanto, es suficiente con que los dos componentes mencionados se configuren dentro de una misma red para poder conectarse. La conexión física se realiza a través de un cable de red Ethernet con conectores RJ45.

Para el presente trabajo, el HMI actúa como cliente debido a que solicita servicios al PLC, principalmente referentes a lectura y escritura de datos en los registros del autómata. Además, es en Vijeo Designer, software de diseño para el HMI, en donde se configura la dirección IP del PLC para acceder y enviar información. La Figura 19 describe lo mencionado recientemente.

Figura 19

Comunicación entre el PLC Siemens S7-1200 1212C AC/DC/RLY y el HMI Schneider Magelis STU855



Conexión entre PLC y variador

Como se indica en la Figura 20, para la comunicación entre el PLC Siemens S7-1200 y el variador Schneider ATV630 se utilizó el protocolo MODBUS TCP/IP dado que es uno de los protocolos que admite el puerto Ethernet integrado en el variador, y que,

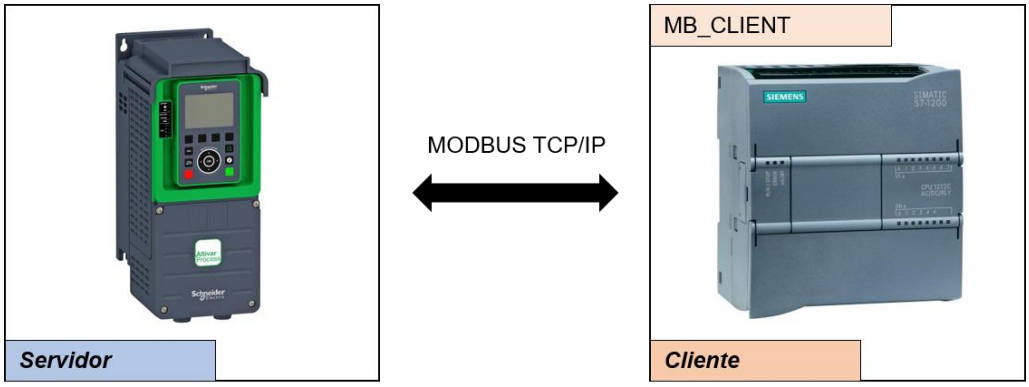
el puerto del PLC también es compatible; con lo cual, no es necesario agregar un puerto adicional en el variador, ni algún módulo de comunicación para el PLC.

En este caso, el PLC se comporta como cliente dado que este es el que controla la lectura y escritura de datos en el variador. En el manual de comunicación por MODBUS del ATV630 se enlista todos los registros que existen, de los cuales, se escoge los necesarios para leer los parámetros del motor y las direcciones donde se deberán ingresar las configuraciones y parámetros de funcionamiento.

Para establecer la comunicación MODBUS TCP/IP con el variador, en la programación del PLC se utiliza la instrucción MB_CLIENT. Este se comunica como cliente Modbus TCP a través de la interfaz PROFINET de la CPU. Aparte de establecer la conexión con el servidor, envía solicitudes, recibe respuestas y controla la desconexión.

Figura 20

Comunicación entre el variador Altivar ATV630U15M3 y el PLC Siemens S7-1200 1212C AC/DC/RLY



Conexión entre PLC y Raspberry Pi

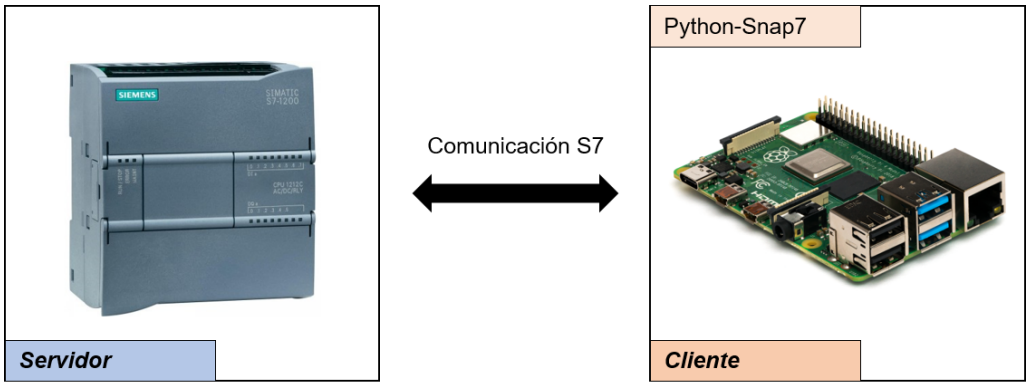
La conexión entre el PLC S7-1200 y la tarjeta Raspberry Pi 4 se realiza a través del protocolo S7, y específicamente con SNAP7, que es una multiplataforma de código abierto que procesa protocolos Ethernet de S7.

El principal lenguaje de programación de la tarjeta Raspberry Pi 4 es Python, y existe una biblioteca de código abierto llamada Python-Snap7 que permite la interconexión entre la tarjeta y el autómata programable a través de cable de red Ethernet.

Snap 7 puede funcionar como cliente o servidor; en consecuencia, en el presente proyecto, la Raspberry se comporta como cliente ya que es quien accede al PLC a través de su dirección IP y solicita lectura o escritura de datos tal y como muestra la Figura 21.

Figura 21

Comunicación entre el PLC Siemens S7-1200 1212C AC/DC/RLY y la tarjeta Raspberry Pi 4



Conexión entre Raspberry Pi y cámara

La imagen de la webcam se transmite hacia la Raspberry Pi 4 a través del conector USB 2.0. Al tratarse de un enlace punto a punto con un estándar de comunicación habitual para periféricos de ordenadores, no es necesarios definir direcciones IP, ni identificadores de comunicación, entre otros.

Programación de PLC

Una vez definidas las funcionalidades del laboratorio remoto, las prácticas que se pueden realizar en el mismo, la descripción del sistema general, los componentes de

la maqueta y su integración; se procede a definir los diagramas que permiten establecer la estructura y la utilidad del PLC en el laboratorio remoto incluyendo el funcionamiento del sistema enfocado en el manejo de bombas. La técnica que se utiliza es el diseño mediante diagramas GRAFCET, para lo cual se sigue un patrón de jerarquías que van a delimitar el funcionamiento de las prácticas a manejar en el laboratorio remoto: configuración del variador de frecuencia, control y sintonización del sistema enfocado en el manejo de bombas eléctricas.

GRAFCET Nivel II

En las Figuras Figura 23 a Figura 26 se muestran los diagramas GRAFCET diseñados para la programación en el PLC. De acuerdo a la función que cumple cada uno, existe una jerarquía la cual se indica en la Figura 22, que permite priorizar el accionar de determinados estados.

- El GRAFCET “Seguridad (G0)” tiene el propósito de inhabilitar los equipos en caso de que ocurra un fallo durante el desarrollo de las prácticas. Este paro de emergencia puede ser accionado desde la HMI o a través de uno de los botones ubicados en el panel.
- “Paro del sistema (G1)” controla la ejecución frente al accionamiento de uno de los botones de “Paro” cuando la maqueta está emulando al sistema de bombas.
- En “Principal (G2)” se realiza los cálculos para simular las variables del sistema y realiza el control PID para el manejo del motor.
- “Parámetros PID (G3)” asigna la ganancia y los tiempos de integración y derivación en el bloque PID que realiza el control del motor.

- “Habilitación (G4)” controla la activación de los diagramas GRAFCET de acuerdo a la práctica que se va a realizar (manejo del variador o las prácticas para el manejo de bombas).
- “Arranque variador (G5)” realiza la configuración de la velocidad referencia y los tiempos de aceleración y desaceleración en el variador de velocidad, además del arranque y paro del motor según el accionar del estudiante, cuando se está realizando la práctica 1.
- Por último, se diseña un GRAFCET “Lectura demanda” que se trata de una interrupción cíclica para registrar constantemente la demanda deseada para el sistema de bombas, valida que dicha demanda sea menor a la capacidad máxima y establece que bombas deberían funcionar a su máxima capacidad y cuál será la bomba controlada para mantener la presión estable.

Figura 22

Jerarquía de los diagramas GRAFCET

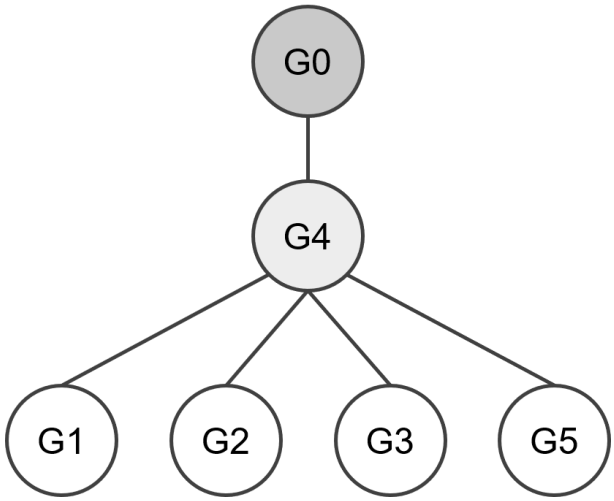


Figura 23

GRAFCET de la programación en el PLC 1/4

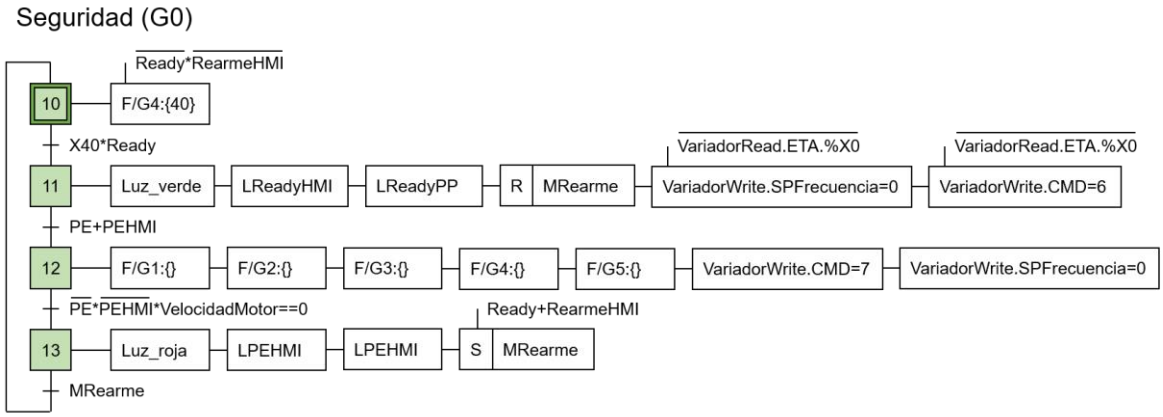
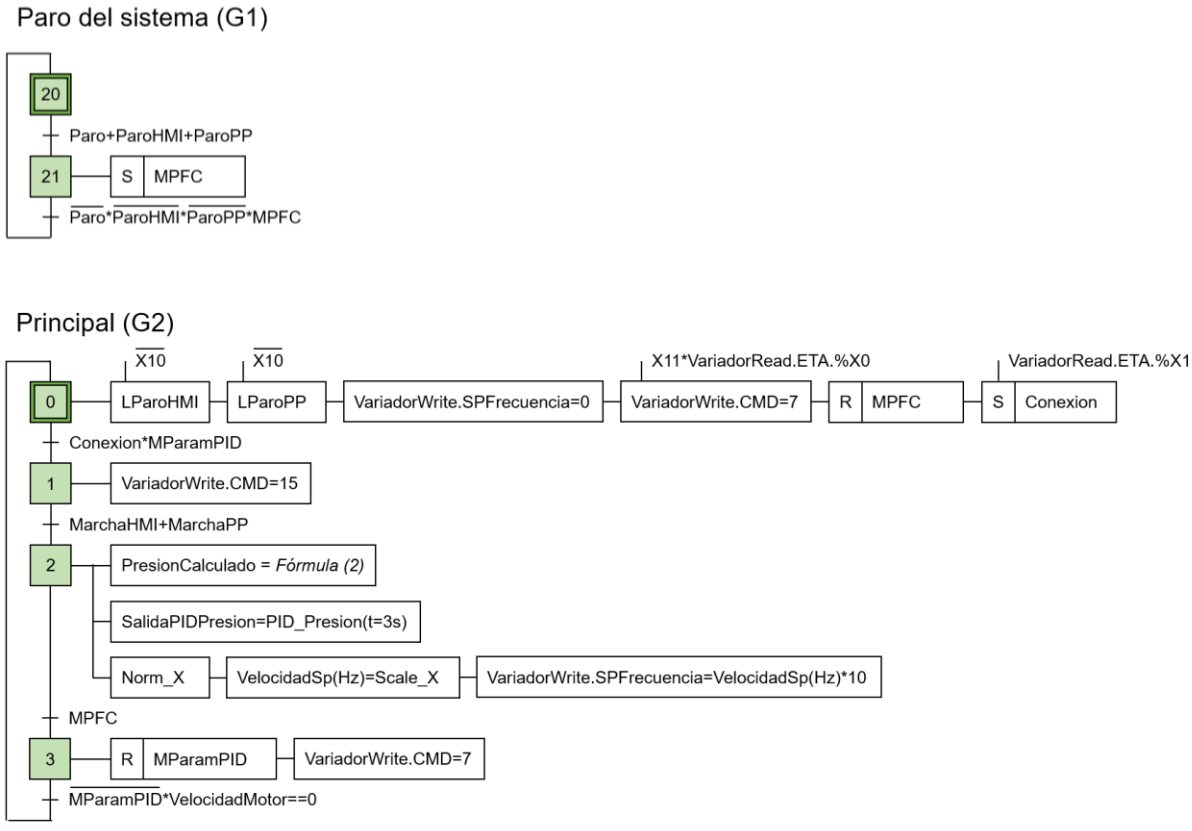


Figura 24

GRAFCET de la programación en el PLC 2/4



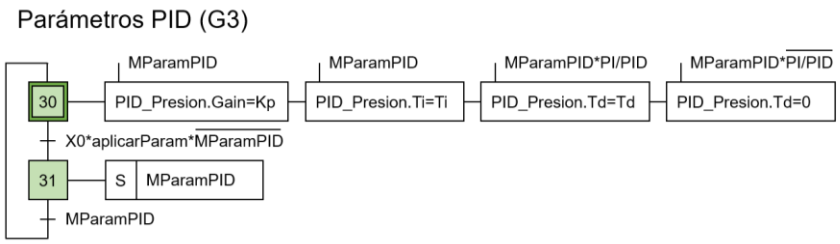
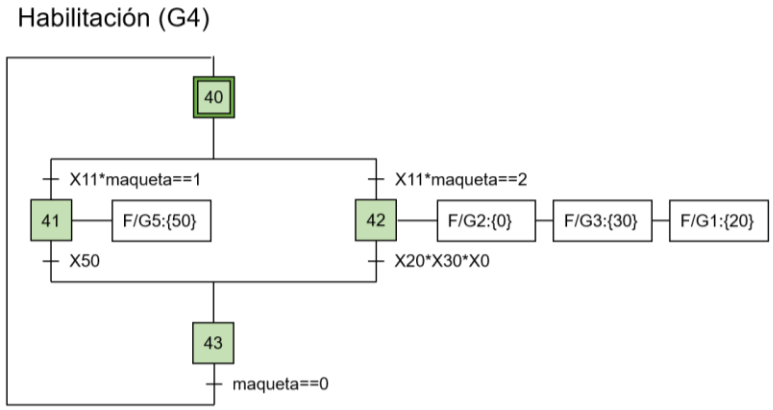


Figura 25

GRAFCET de la programación en el PLC 3/4



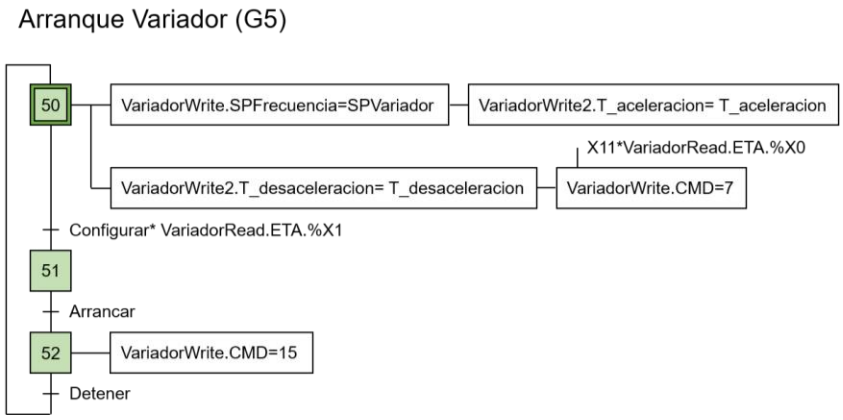
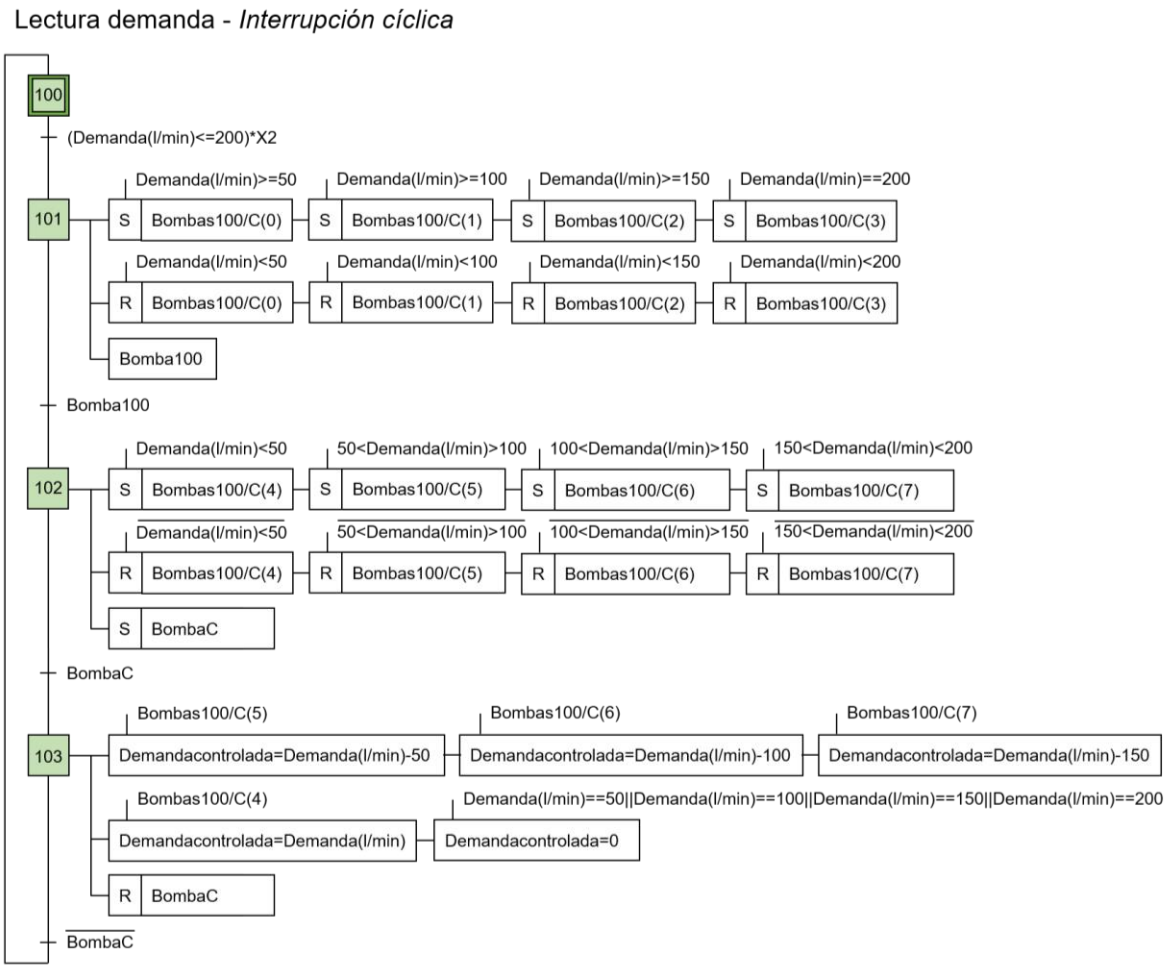


Figura 26

GRAFCET de la programación en el PLC 4/4



Tablas de recursos

Después de diseñar el diagrama GRAFCET que determina cómo va a funcionar el sistema, la Tabla 9 define las entradas, salidas y variables de memoria que van a permitir posteriormente programar el PLC, las Tablas Tabla 10 y Tabla 11 reflejan las variables de los bloques de datos para enviar al variador de frecuencia y la Tabla 12 el bloque de datos para recibir información desde el variador de frecuencia.

Tabla 9

Entradas, salidas y variables de memorias para el PLC Siemens S7-1200

Número	Descripción	Símbolo	Dirección	Tipo
1	Pulsador NA para indicar que el sistema está listo para iniciar	Ready	I0.0	Entradas digitales
2	Pulsador NA para realizar paro del sistema en fin de ciclo	Paro	I0.1	
3	Pulsador NA para realizar paro de emergencia	PE	I0.2	
4	Luz indicadora, muestra que el sistema está en marcha	Luz_verde	Q0.0	Salidas digitales
5	Luz indicadora, muestra que el sistema está detenido por paro de emergencia	Luz_roja	Q0.1	
6	Luz indicadora, muestra que la primera bomba funciona al 100% de su capacidad	BombaF1	Q0.2	
7	Luz indicadora, muestra que la segunda bomba	BombaF2	Q0.3	

Número	Descripción	Símbolo	Dirección	Tipo
	funciona al 100% de su capacidad			
8	Luz indicadora, muestra que la tercera bomba funciona al 100% de su capacidad	BombaF3	Q0.4	
9	Luz indicadora, muestra que la cuarta bomba funciona al 100% de su capacidad	BombaF4	Q0.5	
10	Bit que se activa cuando el usuario oprima un botón desde la interfaz web para indicar que ha configurado todos los parámetros del variador de frecuencia	Configurar	M2.0	Bits de memoria
11	Bit que se activa cuando el usuario oprima un botón desde la interfaz web para arrancar el motor	Arrancar	M2.1	
12	Bit que se activa cuando el usuario oprima un botón desde la interfaz web para detener el motor	Detener	M2.2	

Número	Descripción	Símbolo	Dirección	Tipo
13	Bit que permite escoger entre controlador PI o PID	PI/PID	M8.0	
14	Estado del proceso 0	X0	M40.0	
15	Estado del proceso 1	X1	M40.1	
16	Estado del proceso 2	X2	M40.2	
17	Estado del proceso 3	X3	M40.5	
18	Estado del proceso 10	X10	M40.6	
19	Estado del proceso 11	X11	M40.7	
20	Estado del proceso 12	X12	M41.0	
21	Estado del proceso 13	X13	M41.1	
22	Estado del proceso 20	X20	M41.2	
23	Estado del proceso 21	X21	M41.3	
24	Estado del proceso 100	X100	M41.4	
25	Estado del proceso 101	X101	M41.5	
26	Estado del proceso 102	X102	M41.6	
27	Estado del proceso 103	X103	M41.7	
28	Estado del proceso 30	X30	M50.2	
29	Estado del proceso 31	X31	M50.3	
30	Estado del proceso 40	X40	M50.4	
31	Estado del proceso 41	X41	M50.5	
32	Estado del proceso 42	X42	M50.6	
33	Estado del proceso 43	X43	M50.7	
34	Estado del proceso 50	X50	M51.0	
35	Estado del proceso 51	X51	M51.1	

Número	Descripción	Símbolo	Dirección	Tipo
36	Estado del proceso 52	X52	M51.2	
37	Bit para poner en marcha al sistema desde el HMI	MarchaHMI	M54.0	
38	Bit para dar paro de fin de ciclo del sistema desde el HMI	ParoHMI	M54.1	
39	Bit para dar paro de emergencia del sistema desde el HMI	PEHMI	M54.2	
40	Bit para reestablecer el sistema luego de realizar un paro de emergencia desde el HMI	RearmeHMI	M54.3	
41	Bit indicador del sistema en marcha en HMI	LMarchaHMI	M56.0	
42	Bit indicador de sistema en paro en fin de ciclo en HMI	LParoHMI	M56.1	
43	Bit indicador de sistema en paro de emergencia en HMI	LPEHMI	M56.2	
44	Marca para determinar que el variador de frecuencia está conectado al PLC y fue inicializado	Conexión	M90.3	

Número	Descripción	Símbolo	Dirección	Tipo
45	Marca de paro fin de ciclo	MPFC	M100.0	
46	Marca que finaliza el estado 101 e inicia el estado 102	Bomba100	M100.2	
47	Marca que finaliza el estado 102 e inicia el estado 103	BombaC	M100.3	
48	Marca que indica que el sistema ha sido rearmado cuando se oprime el pulsador RearmeHMI	MRearme	M100.4	
49	Bit que se activa cuando el usuario oprima un botón desde la interfaz web para poner en marcha al sistema de manejo de bombas	MarchaPP	M140.0	
50	Bit que se activa cuando el usuario oprima un botón desde la interfaz web para parar al sistema de manejo de bombas	ParoPP	M140.1	
51	Bit que se activa cuando los parámetros del PID han	aplicarParam	M140.2	

Número	Descripción	Símbolo	Dirección	Tipo
	sido asignados desde la HMI			
52	Marca que se activa cuando los parámetros del PID han sido asignados	MParamPID	M160.0	
53	Entero utilizado para enviar la referencia de velocidad al variador de frecuencia en RPM	VelocidadSP(rpm)	MW4	Enteros de memoria
54	Entero utilizado para asignar el setpoint al controlador de presión	PresionSP(bar)	MW6	
55	Entero utilizado para almacenar la velocidad actual del motor proveniente del variador de frecuencia en RPM	VelocidadMotor(rpm)	MW10	
56	Entero de la demanda requerida por el usuario proveniente desde la HMI en l/min	Demanda(l/min)	MW12	
57	Entero que indica el número de práctica que el usuario va a realizar en el	Maqueta	MW26	

Número	Descripción	Símbolo	Dirección	Tipo
	laboratorio remoto. Ingresado desde la interfaz web.			
58	Entero que va almacenar el setpoint de frecuencia ingresado por el usuario desde la interfaz web.	SPvariador	MW28	
59	Entero que va almacenar el tiempo de aceleración ingresado por el usuario desde la interfaz web.	T_aceleracion	MW30	
60	Entero que va almacenar el tiempo de desaceleración ingresado por el usuario desde la interfaz web.	T_desaceleracion	MW32	
61	Entero utilizado para enviar la referencia de velocidad al variador de frecuencia en Hz	VelocidadSP(Hz)	MW38	
62	Entero utilizado para almacenar el valor de demanda que requiere la bomba controlada en lit/min	DemandaControlada(l/min)	MW68	

Número	Descripción	Símbolo	Dirección	Tipo
63	Entero donde se calcula la velocidad de la bomba 1	velocidadB1	MW92	
64	Entero donde se calcula la velocidad de la bomba 2	velocidadB2	MW94	
65	Entero donde se calcula la velocidad de la bomba 3	velocidadB3	MW96	
66	Entero donde se calcula la velocidad de la bomba 4	velocidadB4	MW98	
67	Byte utilizado en sus primeros 4 bits para activar las bombas que funcionan en su máxima capacidad. Los siguientes 4 bits indican la bomba que será controlada.	Bombas100/C	MB66	Bytes de memoria
68	Valor real que almacena la constante proporcional para el controlador de presión	Kp	MD14	Reales de memoria
69	Valor real que almacena el tiempo de integración para el controlador de presión	Ti	MD18	

Número	Descripción	Símbolo	Dirección	Tipo
70	Valor real que almacena el tiempo derivativo para el controlador de presión	Td	MD22	
71	Real para asignar la referencia de frecuencia al variador desde la HMI en la práctica VFD	Spfrecuencia_HMI	MD42	
72	Real para asignar el tiempo de aceleración al variador desde la HMI en la práctica VFD	t_aceleracion_HMI	MD46	
73	Valor real que almacena la potencia real del motor en unidades hp.	PotenciaMotor(hp)	MD58	
74	Valor real que representa el caudal total calculado en el sistema de manejo de bombas.	CaudalTotal(l/min)	MD62	
75	Valor real que almacena la salida resultante del controlador de presión	SalidaPIDPresion(%)	MD70	
76	Real para asignar el tiempo de desaceleración al	t_desaceleracion_HMI	MD74	

Número	Descripción	Símbolo	Dirección	Tipo
	variador desde la HMI en la práctica VFD			
77	Valor real que almacena la presión real de la bomba controlada. Es calculada a partir de la ecuación (2).	PresionBomba(bar)	MD114	
78	Real donde se convierte el valor de intensidad de corriente que el variador almacena como UInt	CorrienteMotor	MD150	
79	Real donde se convierte el valor de potencia que el variador almacena como Int	PotenciaMotor(W)	MD154	
80	Marca de reloj que se activa cada 2 segundos	Clock_0.5Hz	M0.7	Bits de status
81	Marca de arranque de sistema	FirstScan	M1.0	
82	Marca que se activa siempre	AlwaysTrue	M1.2	

Tabla 10

Bloque de datos "VariadorWrite"

Número	Descripción	Símbolo	Offset	Tipo
1	Entero que sirve para enviar comandos de inicialización, paro o marcha al motor a través del variador de frecuencia	CMD	0.0	Int
2	Entero que sirve para enviar la referencia de velocidad al variador de frecuencia	SPFrecuencia	2.0	Int

Tabla 11

Bloque de datos "VariadorWrite2"

Número	Descripción	Símbolo	Offset	Tipo
1	Entero que sirve para enviar el tiempo de aceleración en segundos al variador de frecuencia	T_aceleracion	0.0	Uint
2	Entero que sirve para enviar el tiempo de desaceleración en segundos al variador de frecuencia	T_desaceleracion	2.0	Uint

Tabla 12

Bloque de datos "VariadorRead"

Número	Descripción	Símbolo	Offset	Tipo
1	Word que indica el estado del motor	ETA	0.0	Word
2	Entero que recibe el valor de corriente del motor	CorrienteMotor	6.0	Uint
3	Entero que recibe el valor de voltaje del motor	VoltajeMotor	14.0	UInt
4	Entero que recibe el valor de potencia del motor	PotenciaMotor	20.0	Int
5	Entero que recibe el valor de velocidad del motor	VelocidadActualRPM	22.0	Int

Diseño de la interfaz de aprendizaje

La interfaz de aprendizaje se realiza mediante un *framework* enfocado al desarrollo de sitios web conocido como Django, el cual está escrito en Python. Debido a esta razón, se puede ejecutar en cualquier plataforma y sistema operativo como Linux, Windows y Mac OS X. Además, incluye formatos HTML, útil para el desarrollo de ventanas web (MDN Web Docs, s.f.).

Ventanas de la aplicación web

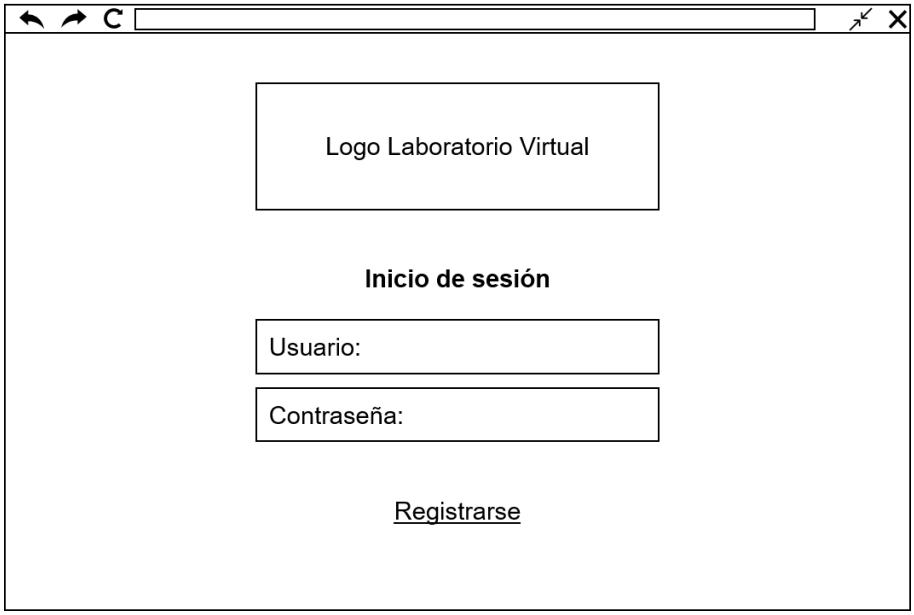
La aplicación web a diseñar debe ser organizada, estructurada y entendible para el usuario; por tanto, se diseñan las ventanas en bloques de tal forma que permitan una guía en el desarrollo de la aplicación.

La Figura 27 muestra el diseño de la ventana de ingreso que se despliega inicialmente. Esta ventana permite:

- Iniciar sesión.
- Redirigirse a la ventana de registro.

Figura 27

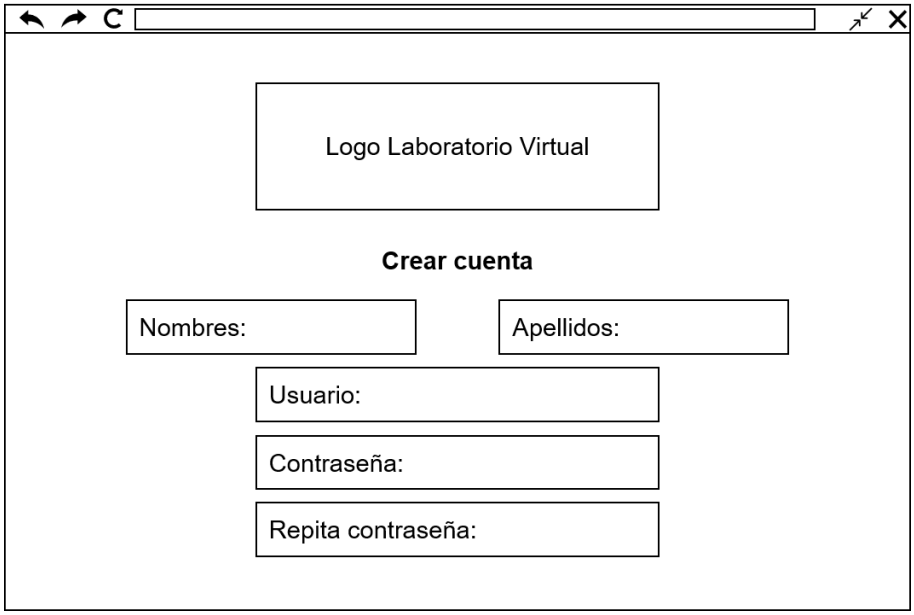
Ventana principal de inicio de sesión



La Figura 28 muestra la ventana que se desplegaría desde la ventana principal y cuya principal función es el registro de un nuevo usuario para acceder a la aplicación web del laboratorio remoto.

Figura 28

Ventana de registro de nuevo usuario

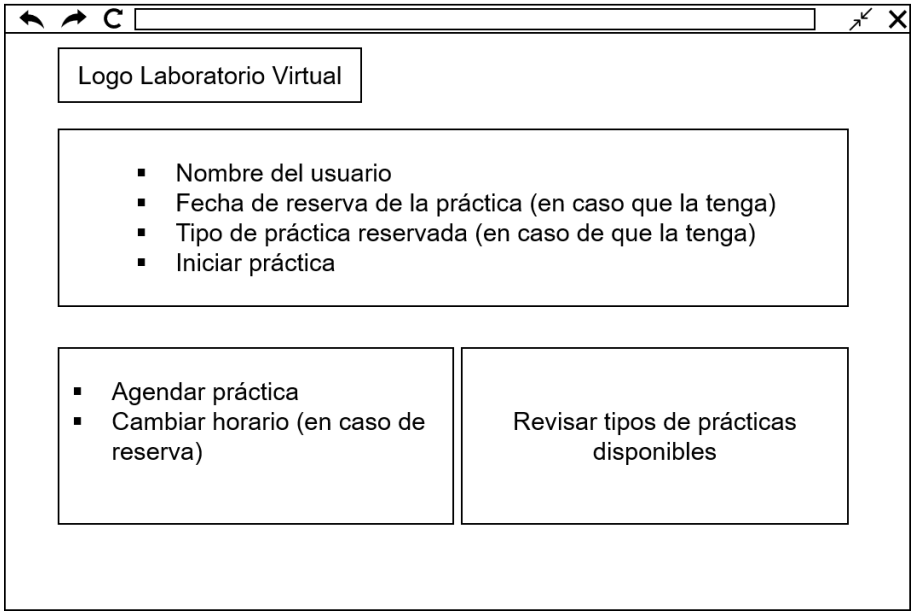


La Figura 29 muestra la ventana que debe aparecer luego del inicio de sesión. Las funcionalidades de esta ventana son:

- Visualizar nombre del usuario, fecha y hora de una práctica reservada, tipo de práctica reservada.
- Redirigirse a la ventana para agendar una práctica.
- Redirigirse a la ventana para cambiar el horario de una práctica.
- Redirigirse a la ventana para revisar información de los tipos de prácticas disponibles.

Figura 29

Ventana de inicio

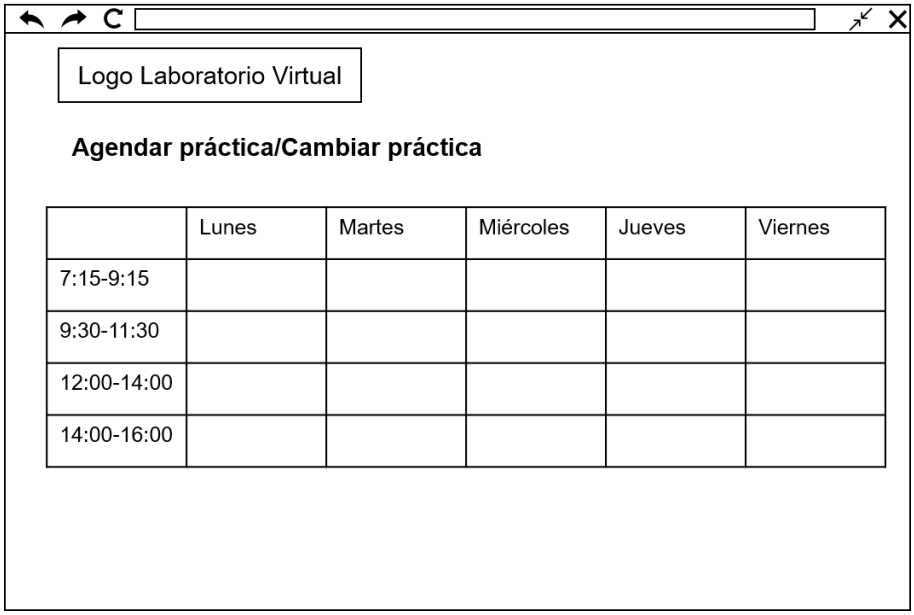


La Figura 30 muestra la ventana de agendamiento/cambio de práctica que proviene de la ventana de inicio. Cabe recalcar que para cualquiera de las dos selecciones se redirige a la misma ventana. Las funcionalidades de la misma son:

- Escoger una fecha y hora para hacer una práctica en caso de que el usuario aún no la tenga reservada.
- Cambiar la fecha y la hora de la práctica en caso de que el usuario ya tenga reservada una.

Figura 30

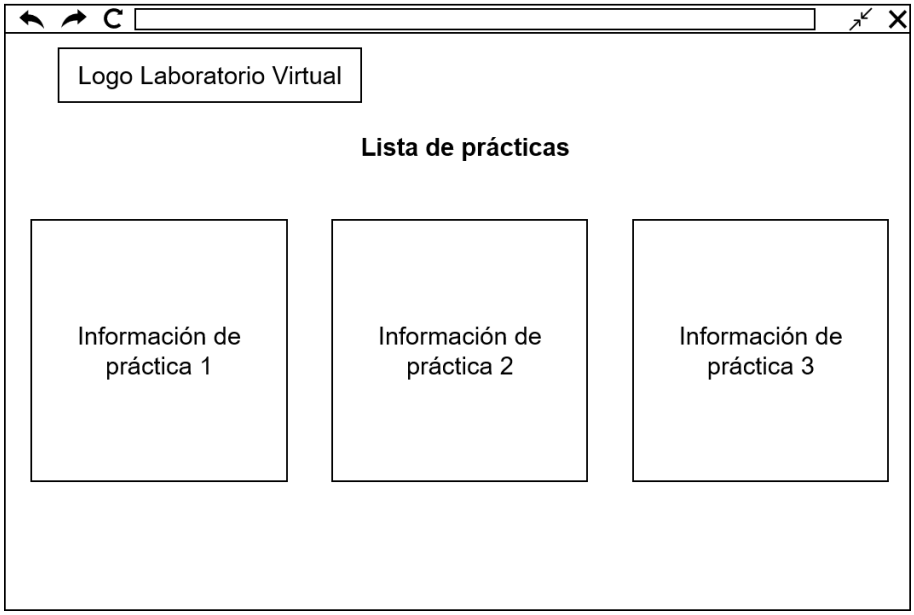
Ventana de agendamiento o cambio de la fecha y hora de una práctica



La Figura 31 presenta el diseño de la ventana en donde se muestra las prácticas disponibles y en donde se muestra la información más relevante de cada una de ellas. Esta ventana aparece cuando el usuario, en la ventana de inicio, selecciona la opción de revisar las prácticas disponibles.

Figura 31

Ventana donde se visualiza la lista de prácticas disponibles del laboratorio remoto



Cuando el usuario inicie la práctica acorde a agendamiento se despliega el laboratorio remoto según el tipo de práctica que haya escogido. En las Figuras Figura 32Figura 34 se muestran los bocetos de las ventanas asociadas a la primera, segunda y tercera práctica respectivamente.

Figura 32

Ventana del laboratorio remoto de la primera práctica

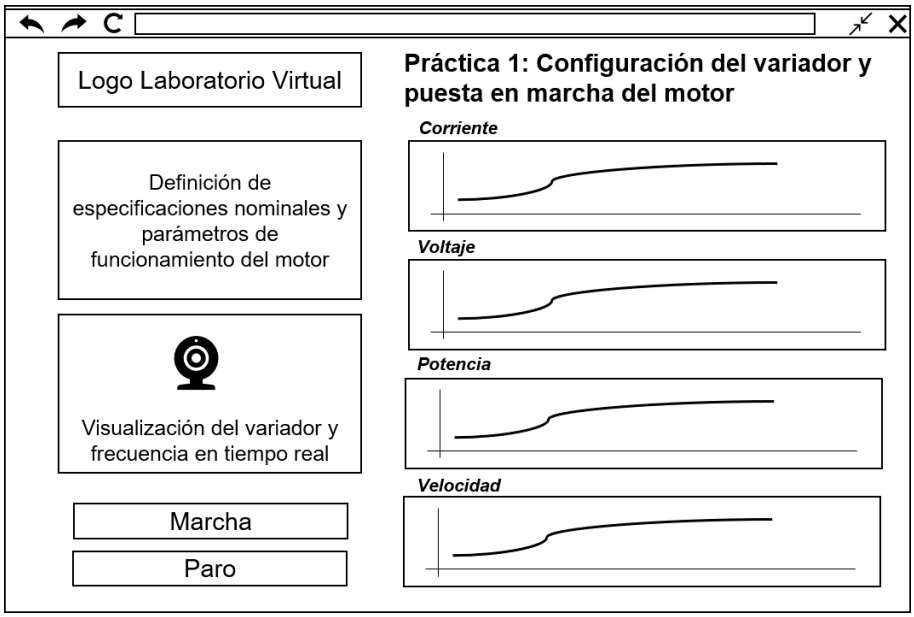


Figura 33

Ventana del laboratorio remoto de la segunda práctica

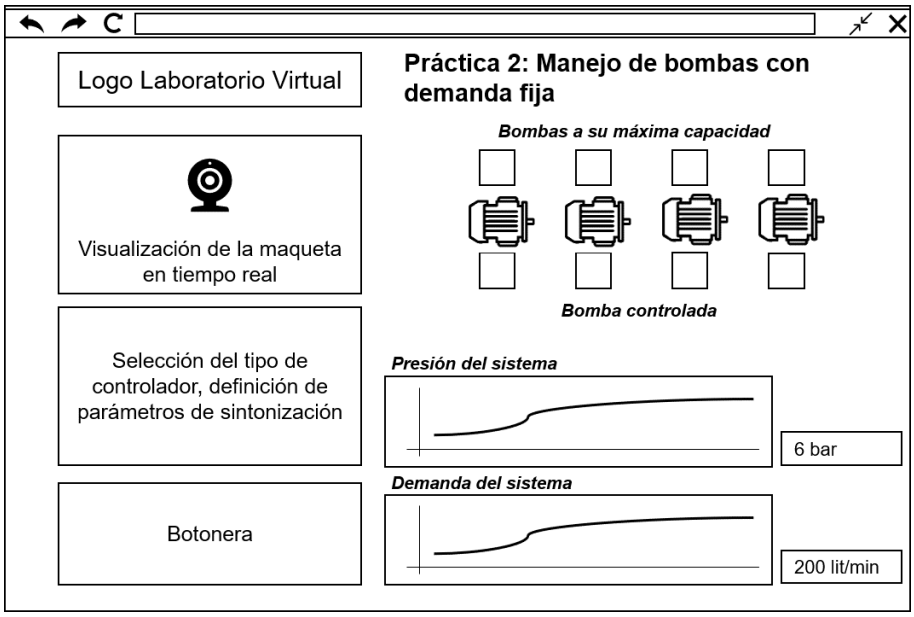
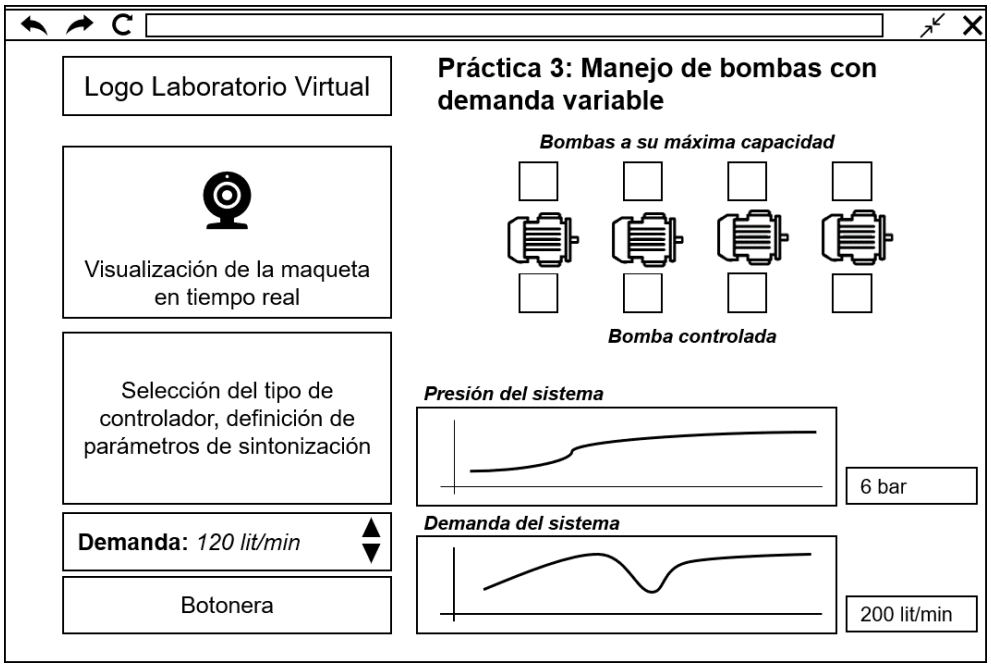


Figura 34

Ventana del laboratorio remoto de la tercera práctica

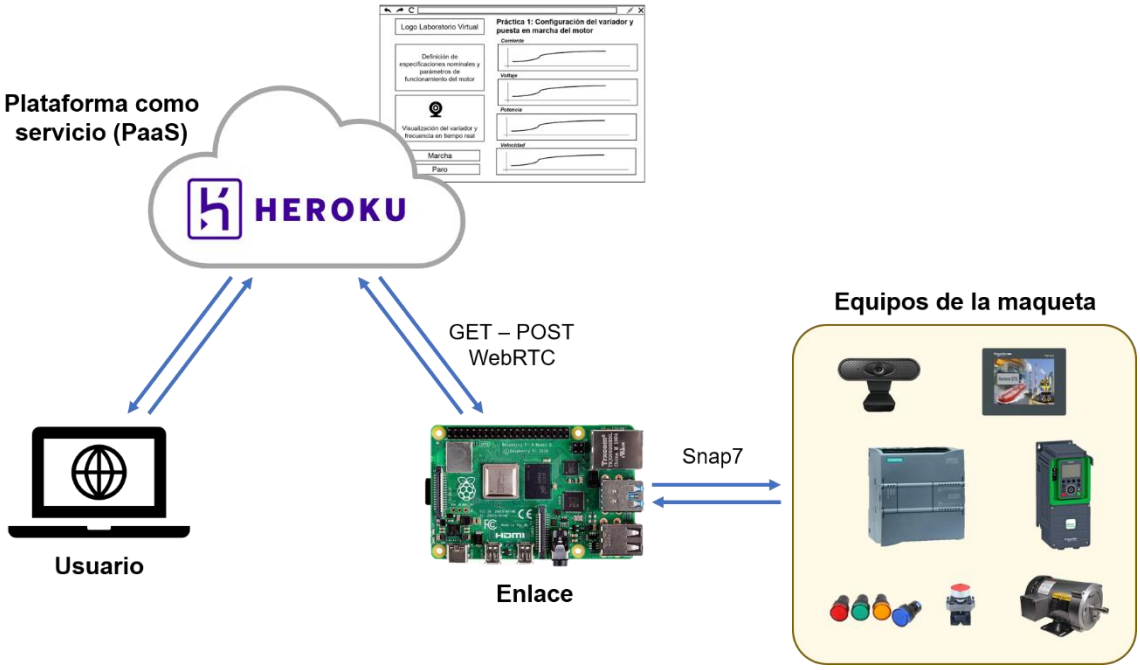


Diseño del enlace para el acceso remoto

Una vez que la maqueta y la aplicación web funcionen de forma local, es necesario diseñar el enlace que permita al usuario acceder a la aplicación de manera remota. Se detalla a continuación los componentes en software que solucionan este requerimiento. En la Figura 35 se muestra el esquema del enlace para el acceso remoto.

Figura 35

Esquema completo del laboratorio remoto



Heroku

El laboratorio remoto consiste en un conjunto de interfaces integradas en una aplicación cuyo objetivo es ser accedida por el usuario desde cualquier lugar para realizar prácticas de laboratorio. Una de las soluciones más utilizadas especialmente por el avance de la tecnología, es la creación y alojamiento de la aplicación en la nube.

Las plataformas de servicio (PaaS) son herramientas que permiten lo mencionado recientemente, para lo cual se ha optado por utilizar Heroku como la plataforma que almacene la aplicación del laboratorio remoto diseñada en el presente trabajo.

Heroku es un servicio que soporta varios lenguajes de programación como Python, PHP, Ruby, entre otros. Tomando en cuenta que la aplicación se desarrolla en Django que es escrito en lenguaje Python, Heroku es una solución óptima ya que soporta dicho lenguaje de programación y requiere únicamente de la creación de una

cuenta para subir la aplicación. Además, dispone de dos versiones: una gratuita y otra pagada, no obstante, la que no tiene costo otorga prestaciones básicas con capacidades suficientes para almacenar la aplicación del laboratorio remoto en la nube. De esta forma, el usuario puede acceder a la aplicación desde cualquier dispositivo a través del Internet (Romero, s.f.).

Solicitudes GET y POST

GET y POST son dos tipos diferentes de solicitudes HTTP. Con el método GET se recupera una representación del recurso especificado e incluye todos los datos necesarios en la URL. Por otro lado, POST envía datos para ser procesados (por ejemplo, desde un formulario HTML) al recurso identificado. Esto puede resultar en la creación de un nuevo recurso o la actualización de recursos existentes (Ibanez, 2018).

Para Python existe una biblioteca denominada *requests*, el cual es el estándar para realizar solicitudes HTTP. Evita las complejidades de realizar solicitudes detrás de una API para enfocarse en interactuar con los servicios y consumir datos de la aplicación (Bailey, 2022). Dispone de todos los métodos de petición HTTP a excepción de “CONNECT” y “TRACE”.

WebRTC

Para la transmisión del video utilizando la cámara web se utiliza Web Real-Time Communications (WebRTC). WebRTC es un proyecto de código abierto que brinda capacidades de comunicación de voz, texto y video en tiempo real entre navegadores web y dispositivos. Proporciona interfaces de programación de aplicaciones (API) escritas en JavaScript para crear comunicaciones peer-to-peer (P2P) entre navegadores web de Internet y aplicaciones móviles sin preocuparse por la compatibilidad y el soporte para contenido basado en audio, video o texto (Shacklett, 2021).

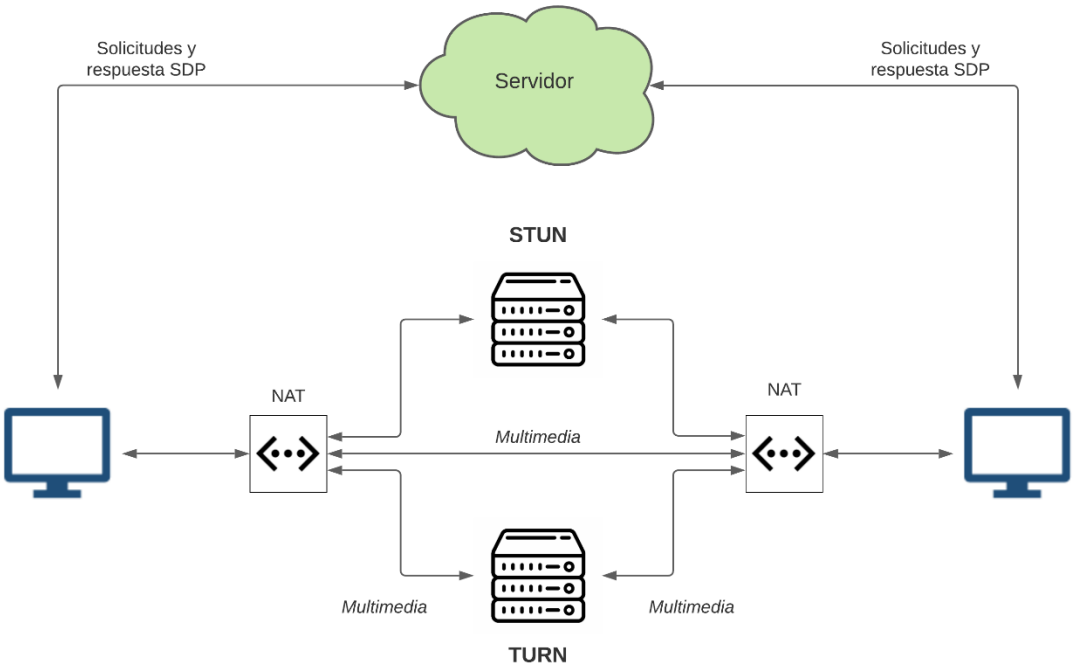
WebRTC no requiere componentes o complementos de terceros, puede ajustar la calidad de la comunicación, el ancho de banda y el flujo de tráfico cuando cambian las condiciones de la red, además, actualmente es soportado por Google Chrome, Mozilla Firefox, Opera, Safari y demás navegadores basados en Chromium, tanto en las versiones de escritorio como de aplicación móvil (3CX, 2021).

En la Figura 36 se observa un diagrama que describe la arquitectura punto a punto en el navegador utilizando WebRTC. El servidor permite transmitir las ofertas y respuestas entre los clientes. La oferta incluye toda la información sobre la configuración propuesta por la persona que llama. Luego, el destinatario responde con una respuesta, que es una descripción de su terminal para la llamada. De esta forma, ambos dispositivos comparten entre sí la información necesaria para intercambiar datos multimedia (Mozilla Corporation, 2022).

NAT (*Network Address Translation*) se utiliza para dar al dispositivo una dirección IP pública. STUN (*Session Traversal Utilities for NAT*) es un protocolo que se usa para descubrir direcciones públicas y determina cualquier restricción en su enrutador que impida una conexión directa con un par. Por otro lado, TURN (*Traversal Using Relays around NAT*) está destinado a eludir la restricción NAT simétrica al abrir una conexión con un servidor TURN y transmitir toda la información a través de ese servidor. La mayoría de las llamadas de WebRTC se conectan con éxito mediante servidores STUN, con lo cual, un servidor TURN se usa para retransmitir el tráfico si falla la conexión directa (Rossouw, 2020).

Figura 36

Descripción general de la arquitectura WebRTC



Capítulo IV. Implementación

En este capítulo se explica todo el procedimiento realizado para llevar a la práctica lo expuesto en el capítulo anterior. En la sección de implementación de la maqueta, se comienza por describir el armado y conexión de los equipos de la maqueta mostrando el diagrama esquemático. Luego se explica la configuración y programación del PLC basado en diagrama GRAFCET. De igual forma, se indica la configuración realizada para el panel de operador y se expone las interfaces diseñadas resaltando el propósito de cada una. Por último, se señalan todos los pasos realizados para establecer la comunicación de los equipos: PLC – Variador de velocidad, PLC – HMI y PLC – Raspberry Pi 4.

En la segunda sección se indica el desarrollo de la parte estática de la aplicación web, comenzado por mostrar el procedimiento para crear un proyecto y una aplicación en Django. Después se describe la estructura del proyecto y los componentes más importantes para desarrollo la aplicación. Finalmente, se muestra la programación realizada en relación al diseño de las interfaces (*templates*), la parte lógica o acciones del servidor (*views*) y la creación de la base de datos (*models*).

En las siguientes secciones se señala la forma para alojar la aplicación web en Heroku y crear el enlace con la Raspberry Pi 4 y, por último, se expone todas las herramientas y desarrollo realizado para generar gráficas dinámicas y transmitir el video desde la maqueta en tiempo real.

Implementación de la maqueta

El armado y conexión de los equipos se realizó partiendo de uno de los sistemas de entrenamiento en control de movimiento que dispone la universidad (Gavilema Casa & Yánez Garzón, 2020). En ella se encuentran ya instaladas el variador de velocidad, el panel de operador, el motor trifásico, la fuente de alimentación y el switch industrial; con cual, se añade a la estación el PLC, los botones y luces.

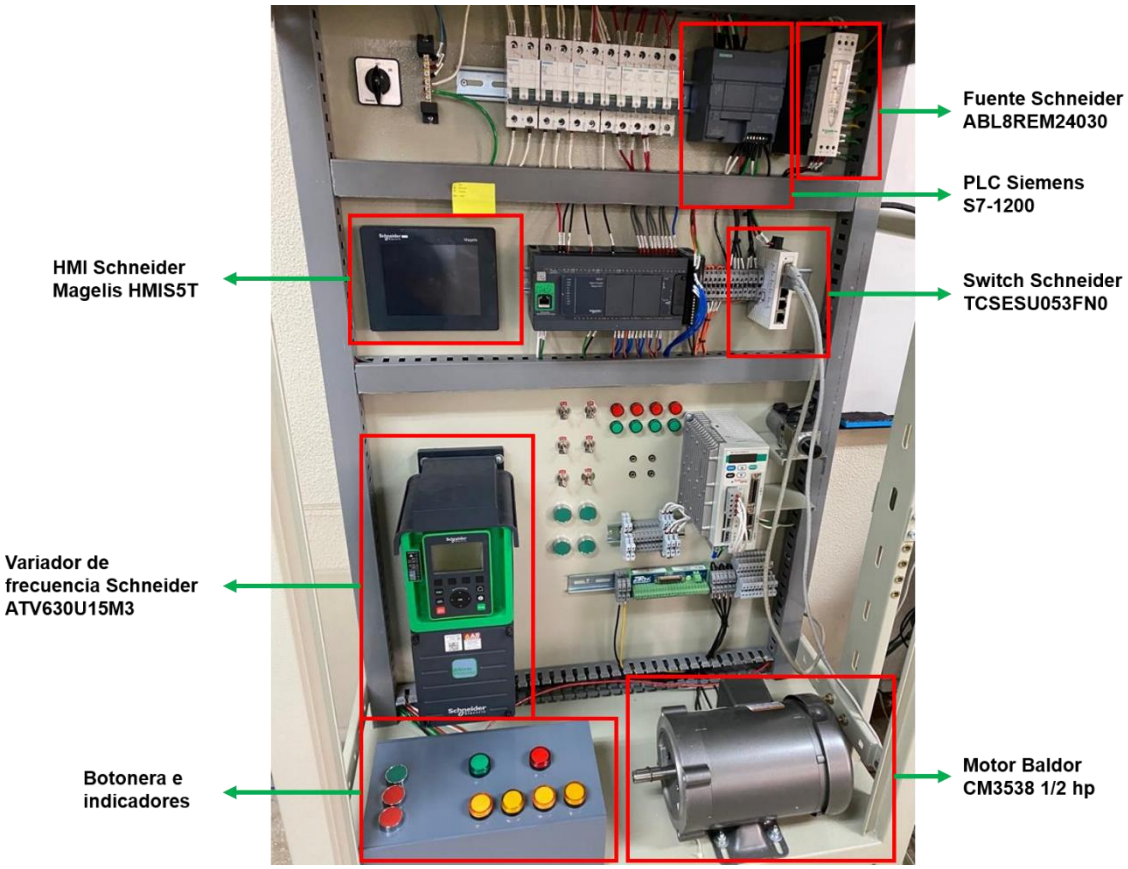
Diagrama esquemático

Para implementar la maqueta, es necesario realizar un diagrama esquemático que permita establecer las conexiones eléctricas que se deben implementar entre todos los componentes. Esto, dado que cada elemento de la maqueta requiere de una alimentación distinta, es decir, trifásica, bifásica o monofásica. El plano que resume todas las conexiones eléctricas de la maqueta se encuentra en el Anexo A.

A partir del esquema mencionado, se implementa la maqueta físicamente en el laboratorio de Robótica y PLCs de la Universidad de las Fuerzas Armadas ESPE matriz Sangolquí. El resultado se muestra en la Figura 37.

Figura 37

Maqueta implementada en el laboratorio de PLC y Robótica de la Universidad de las Fuerzas Armadas ESPE matriz Sangolquí

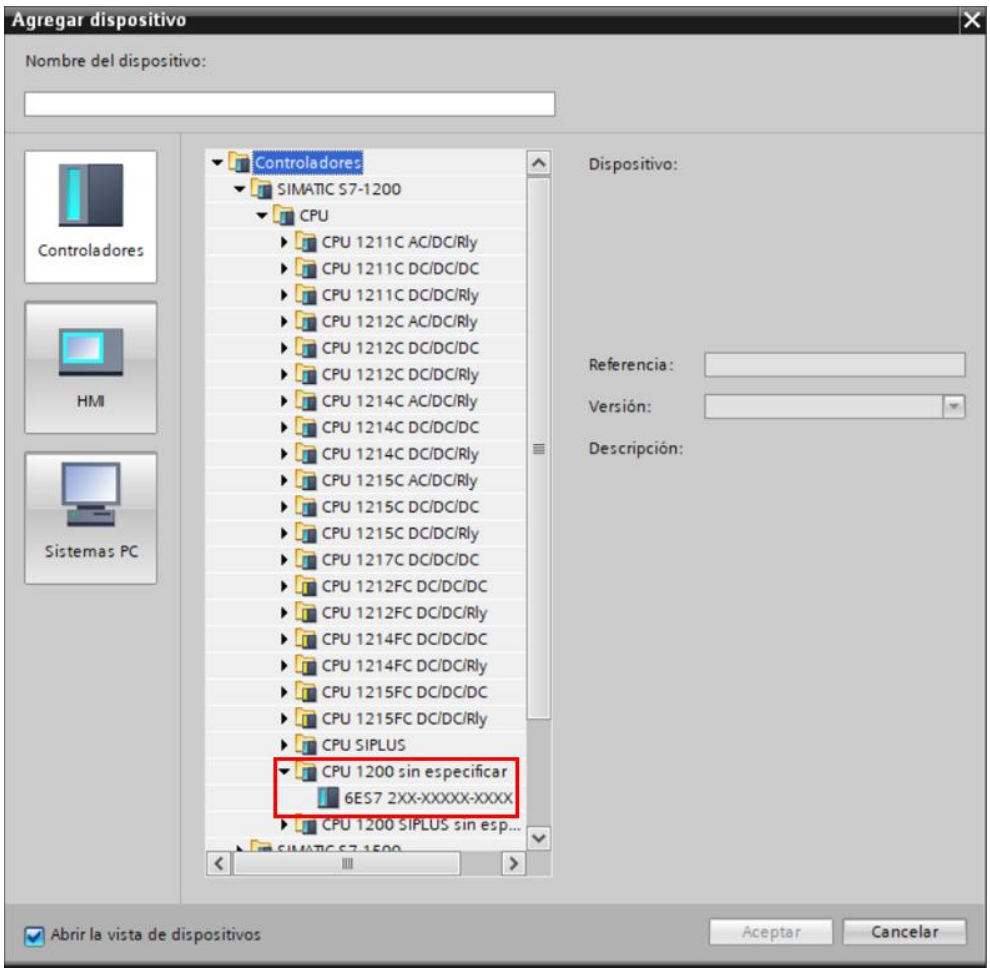


Configuración y programación del PLC

El primer componente que se configura es el PLC Siemens 1200 que es programado a través del software Tia Portal V15.1. Inicialmente, se crea un proyecto y se agrega un dispositivo SIMATIC S7-1200 con CPU sin especificar como muestra la Figura 38.

Figura 38

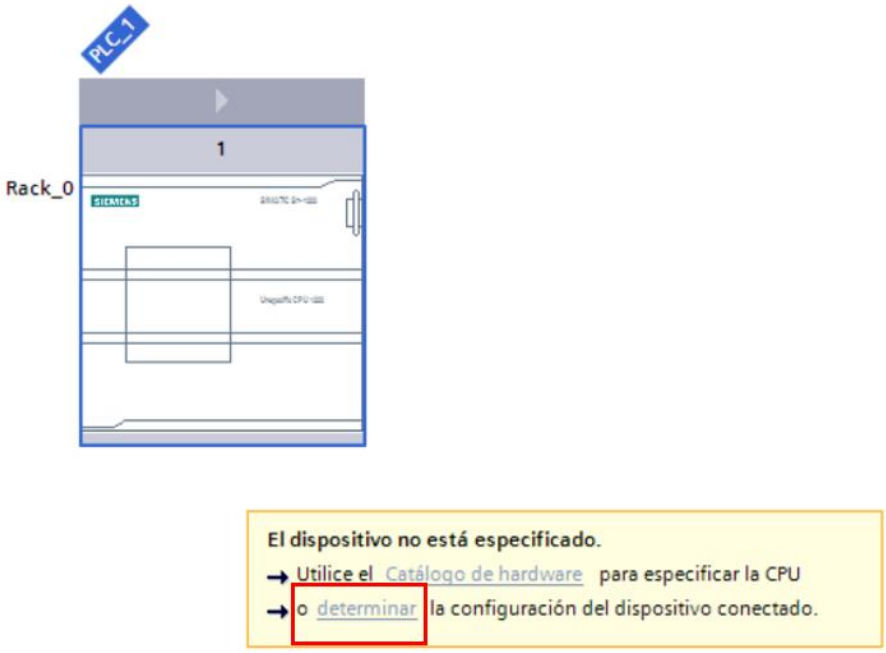
Selección del controlador SIMATIC S7-1200 con CPU sin especificar



A continuación, la Figura 39 presenta cómo el software permite determinar la configuración del dispositivo que está conectado a la computadora por medio de cable de red, estableciendo el modelo exacto del CPU y todas sus características internas.

Figura 39

Opción para determinar la configuración del PLC físico



Una vez que se reconozca el PLC físico tal y como se aprecia en la Figura 40, el dispositivo queda configurado dentro del proyecto, incluyendo la asignación de la dirección IP (para este caso es la 192.168.0.1 con máscara de subred 255.255.255.0 tal y como se evidencia en la Figura 41).

Figura 40

Detección del PLC Siemens S7-1200 físico

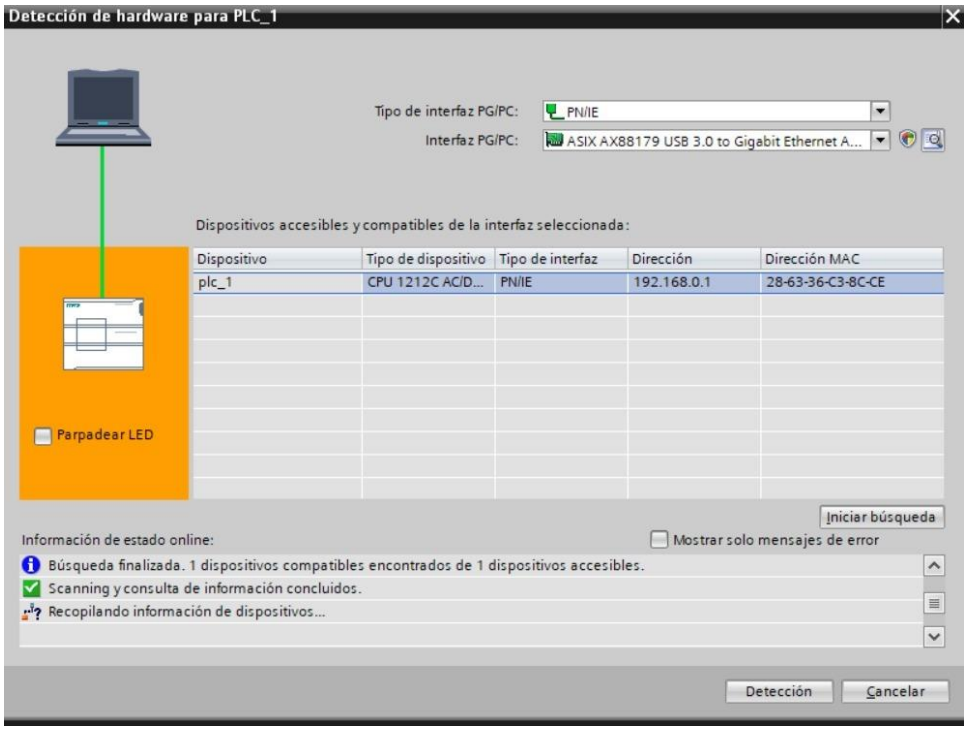
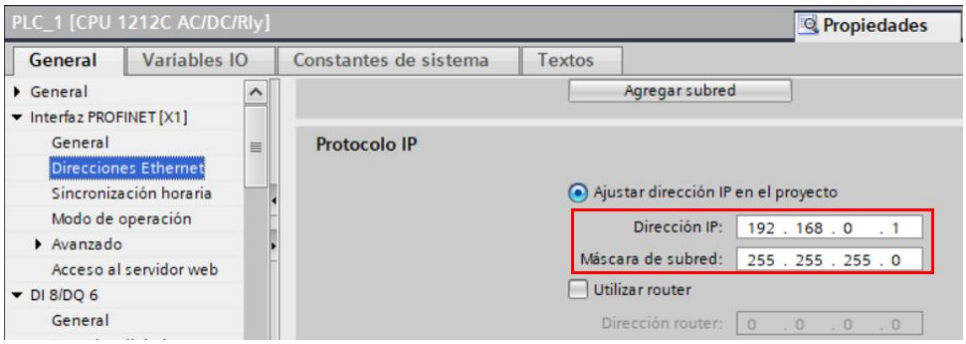


Figura 41

Asignación de la dirección IP y máscara de subred del PLC Siemens S7-1200

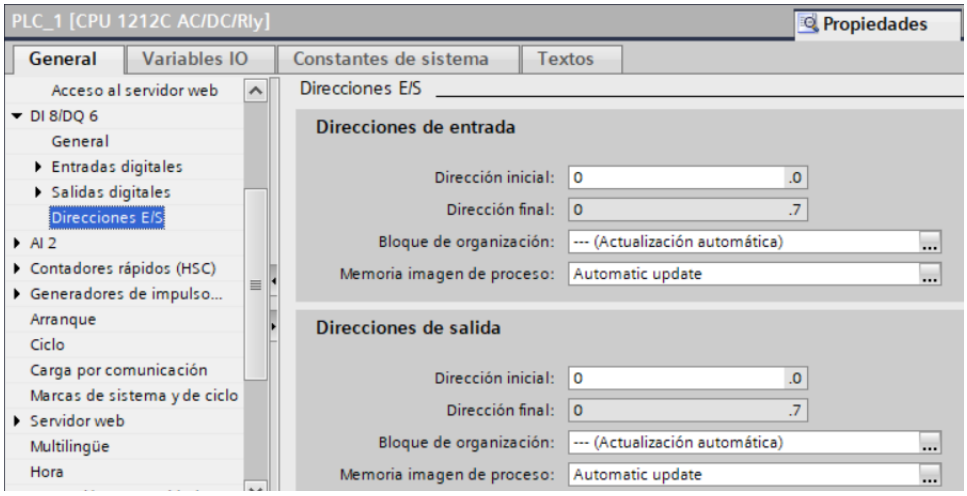


Otro aspecto relevante a tomar en cuenta, es la definición de las direcciones de entrada y salida del controlador, que luego son utilizadas en la lógica de programación. Como el PLC utilizado dispone de 8 entradas y 6 salidas, se define para las entradas la dirección inicial I0.0 y la final I0.7; y para las salidas la Q0.0 y Q0.5 (tomar en cuenta que por defecto Tia Portal asigna las direcciones de salida hasta la dirección 7, sin

embargo, debido a las especificaciones del PLC solo se debe utilizar hasta la dirección Q.0.5) como se muestra en la Figura 42.

Figura 42

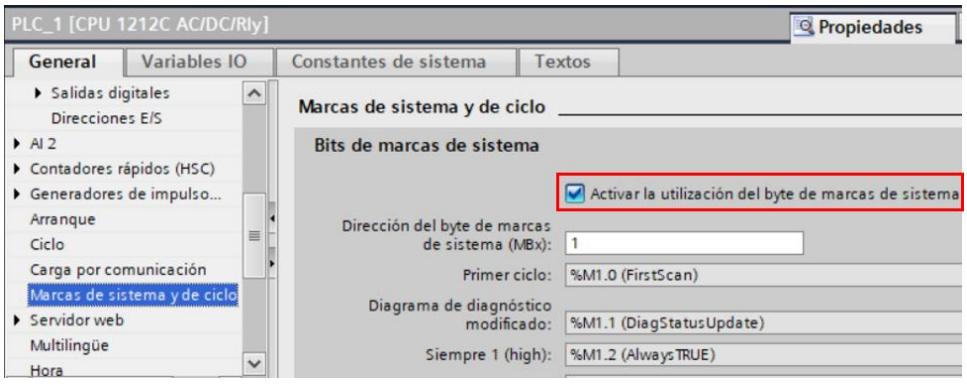
Direcciones de entrada y salida del PLC



Luego, se procede a activar dos parámetros que son útiles dentro de la programación y seguridad del PLC. La primera configuración son las marcas de sistema y de ciclo, que son bits internos del PLC con múltiples funcionalidades. Para activar este recurso se debe acceder a las propiedades del CPU del autómata y dirigirse a la sección que indica la Figura 43.

Figura 43

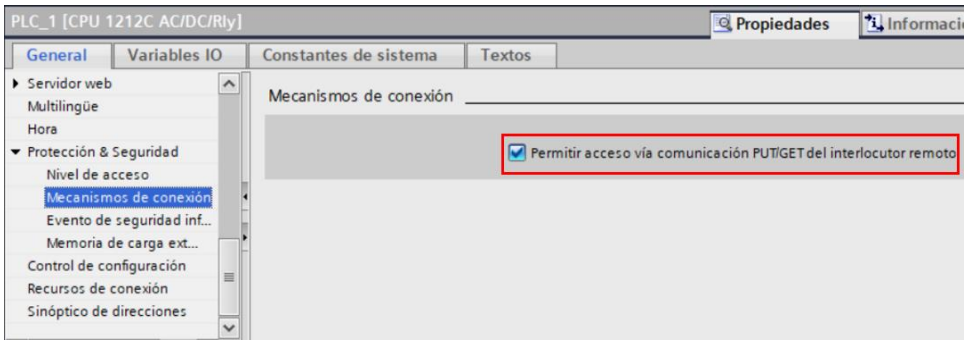
Activación de marcas de sistema y de ciclo



El segundo parámetro es la activación de los mecanismos de conexión de la CPU del PLC. Esta configuración permite la adquisición y envío de datos en el autómata y para habilitar esta herramienta hay que dirigirse a la sección de propiedades que indica la Figura 44.

Figura 44

Activación del acceso vía comunicación PUT/GET



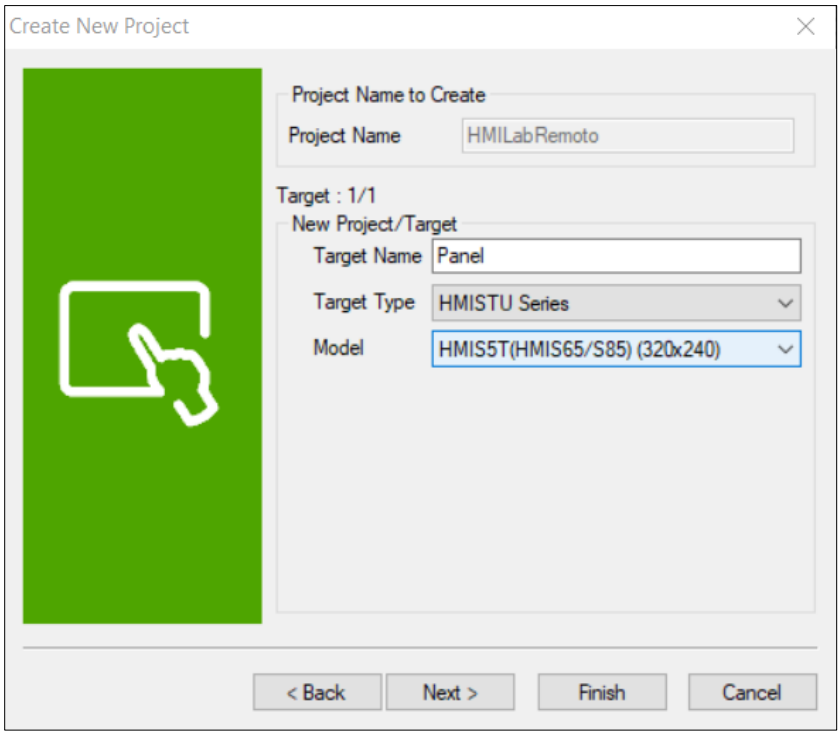
Finalmente, se procede a programar lo diseñado a través de diagramas GRAFCET anteriormente. Se utilizó el lenguaje de programación de escalera o también conocido como *Ladder*.

Interfaces HMI

Para configurar y diseñar las interfaces del panel de operador se utiliza Vijeo Designer versión 6.2.8, software propietario de Schneider Electric. Se comienza creando un proyecto y escogiendo el modelo del panel. En este caso, como se indica en la Figura 45, no se selecciona el panel Magelis HMI STU855, sino el panel posterior HMIS5T donde se encuentran los puertos de comunicación y comandos.

Figura 45

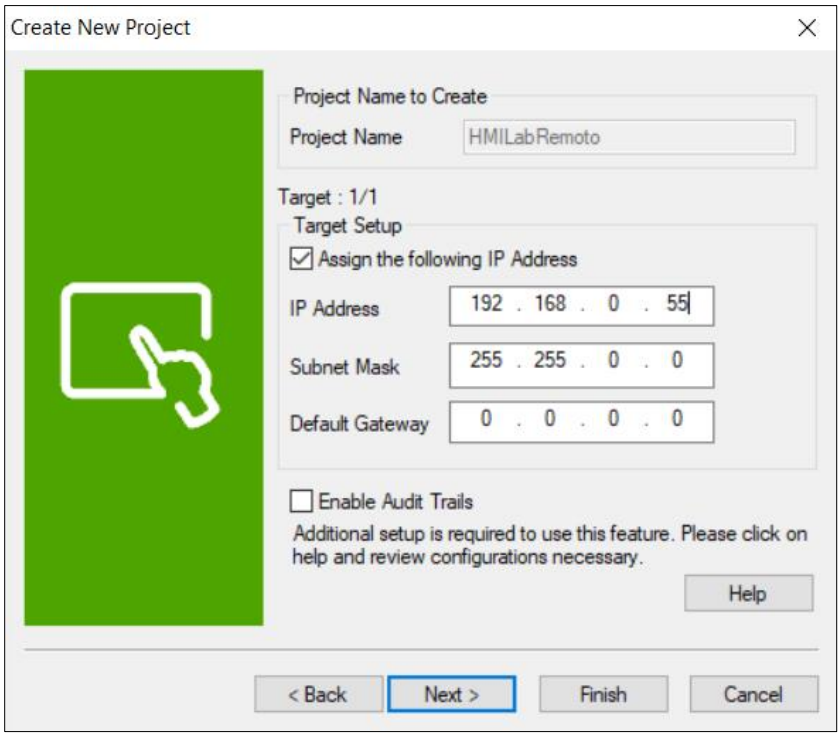
Selección del modelo HMIS5T



En la siguiente ventana se fija la dirección IP del panel y la máscara de red. De acuerdo al diagrama de red de la Figura 18, la dirección IP es 192.168.0.55 con máscara 255.255.255.0, por lo tanto, se configura esta información como se muestra en la Figura 46.

Figura 46

Asignación de la dirección IP y máscara de red



Una vez creado el proyecto, se diseña todas las interfaces que servirán para monitorizar y controlar de forma local la maqueta. A continuación, se exponen todas las pantallas elaboradas, la función que cumplen y como fueron desarrolladas:

Selección de práctica. Esta es la interfaz de inicio del panel de operador. Permite seleccionar cuál práctica se va a ejecutar desde la HMI. Como se muestra en la Figura 47, en la parte superior se coloca el logo de la Universidad y de la carrera de Electrónica, Automatización y Control. En la parte inferior se añade los campos de fecha y hora, lo cuales, se configuran de forma local. Mientras que en la parte central se coloca el logo del laboratorio remoto y dos botones para escoger si se va a probar el funcionamiento de la práctica 1 o el comportamiento de la maqueta al realizar las prácticas 2 o 3.

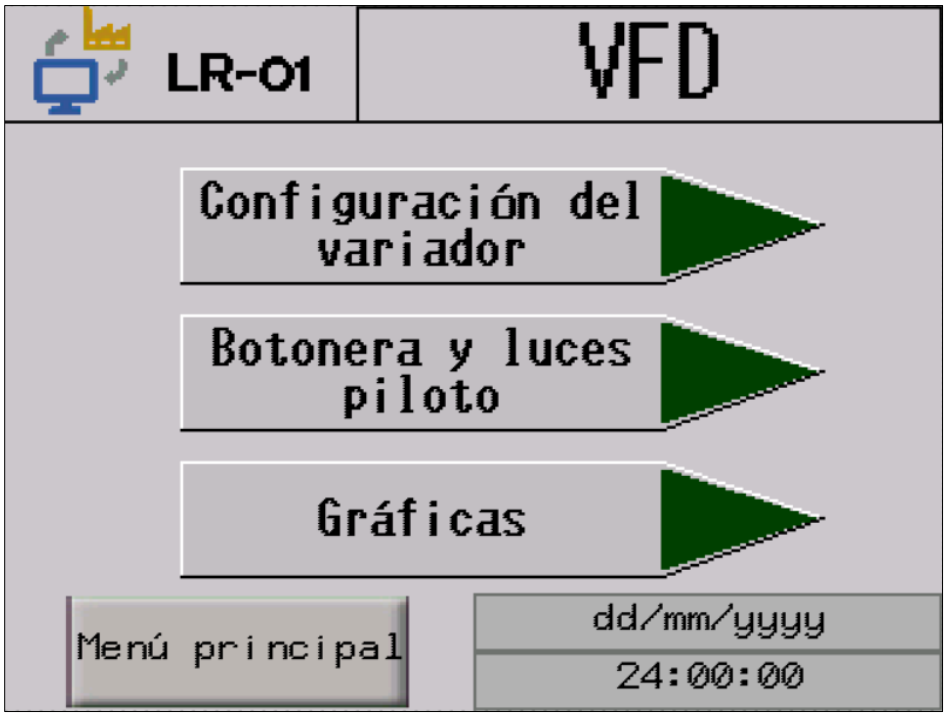
Figura 47

Interfaz – Selección práctica

Menú VFD. Esta pantalla permite navegar entre los apartados para configurar el variador de frecuencia, controlar el arranque y parada del motor, y observar gráficas de funcionamiento. Como se muestra en la Figura 48, la selección se realiza utilizando tres botones y uno adicional para regresar al menú principal.

Figura 48

Interfaz - Menú variador



Botonera e indicadores. La interfaz mostrada en la Figura 49 permite poner en marcha o detener el funcionamiento, sea únicamente del motor cuando en el menú principal se escogió probar el variador, o la simulación del sistema de manejo de bombas cuando se escoge la segunda opción. Además, cuenta con un botón en la parte inferior para realizar un paro de emergencia, en caso de que ocurra un fallo y se requiera detener inmediatamente toda la maqueta. Este botón se encontrará en todas las demás interfaces en la misma ubicación por cuestión de seguridad. Una vez solucionado el problema que ocasionó el fallo, existe un botón de “Rearme” para habilitar nuevamente la maqueta. En la parte derecha existen indicadores para mostrar el estado del sistema al momento de desarrollar cualquiera de las dos prácticas.

Figura 49

Interfaz – Botonera e indicadores

Parámetros variador. En la interfaz que se muestra en la Figura 50 se ingresan los parámetros en el variador para establecer el funcionamiento del motor, estos son: referencia de frecuencia en Hz, tiempo de aceleración en segundos y tiempo de desaceleración en segundos. En la parte inferior está el botón para registrar los parámetros en el variador, al igual que un botón para regresar al menú del variador de frecuencia y el paro de emergencia.

Figura 50

Interfaz – Parámetros variador

Gráficas de tendencia VFD. En las Figuras Figura 51 a Figura 54 se muestran las interfaces donde se crean líneas de tendencia de las variables que indican el estado de funcionamiento del motor: velocidad, corriente, voltaje y potencia. En las gráficas se dibujan hasta 21 muestras tomadas cada 1 segundo. En la parte derecha de la interfaz se encuentran flechas para moverse entre las cuatro interfaces y monitorear todos los parámetros de forma rápida. En la parte inferior se escribe el valor actual de cada magnitud junto a las unidades en la cual se muestra.

Figura 51

Interfaz – Gráfica de tendencia velocidad del motor

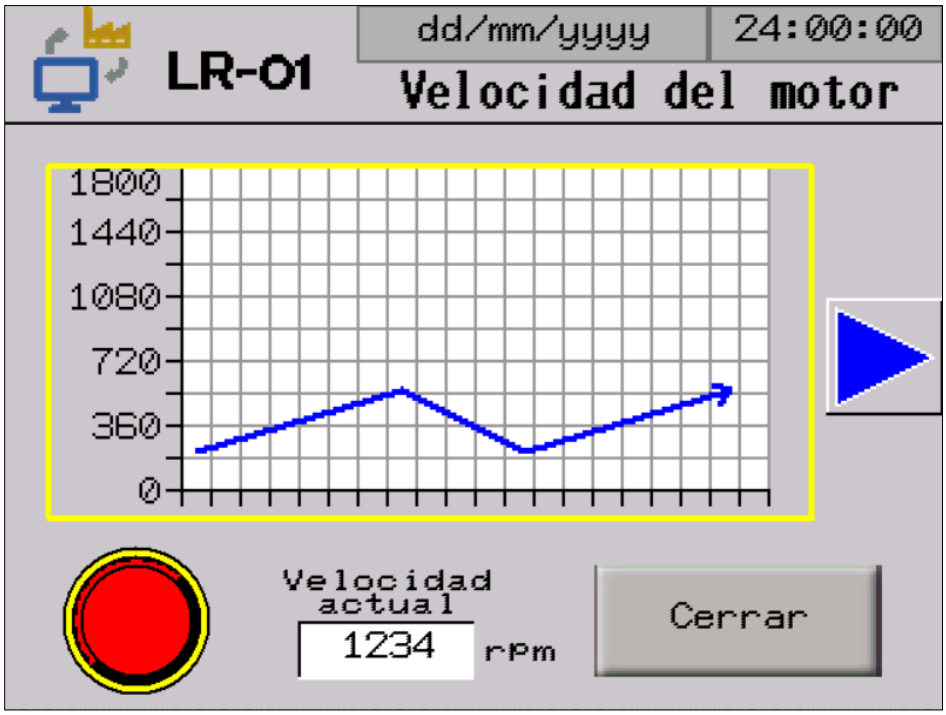


Figura 52

Interfaz – Gráfica de tendencia intensidad de corriente del motor

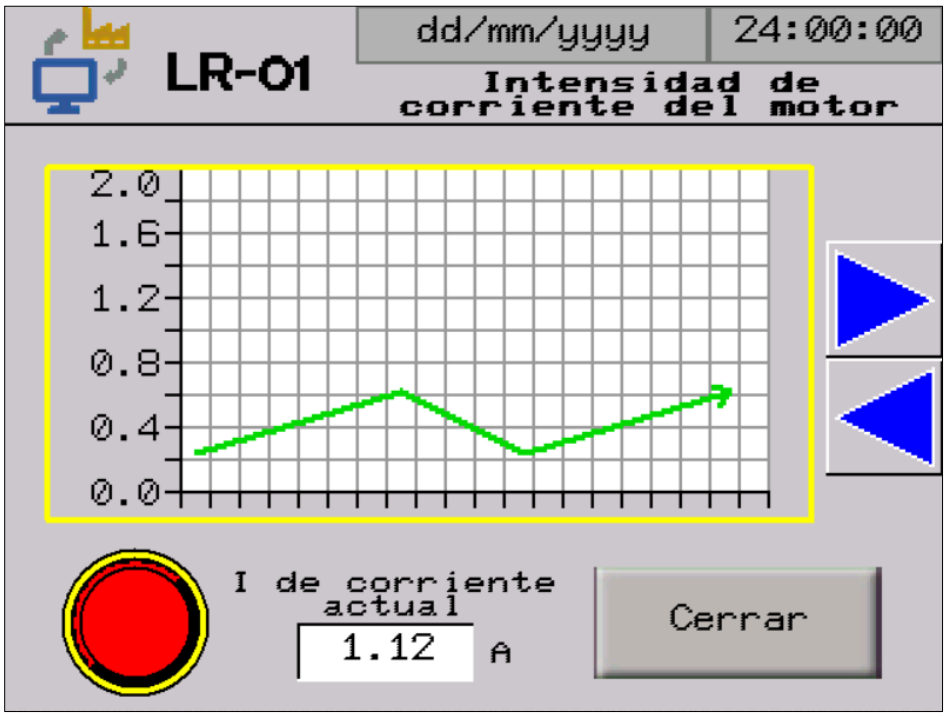


Figura 53

Interfaz – Gráfica de tendencia voltaje del motor

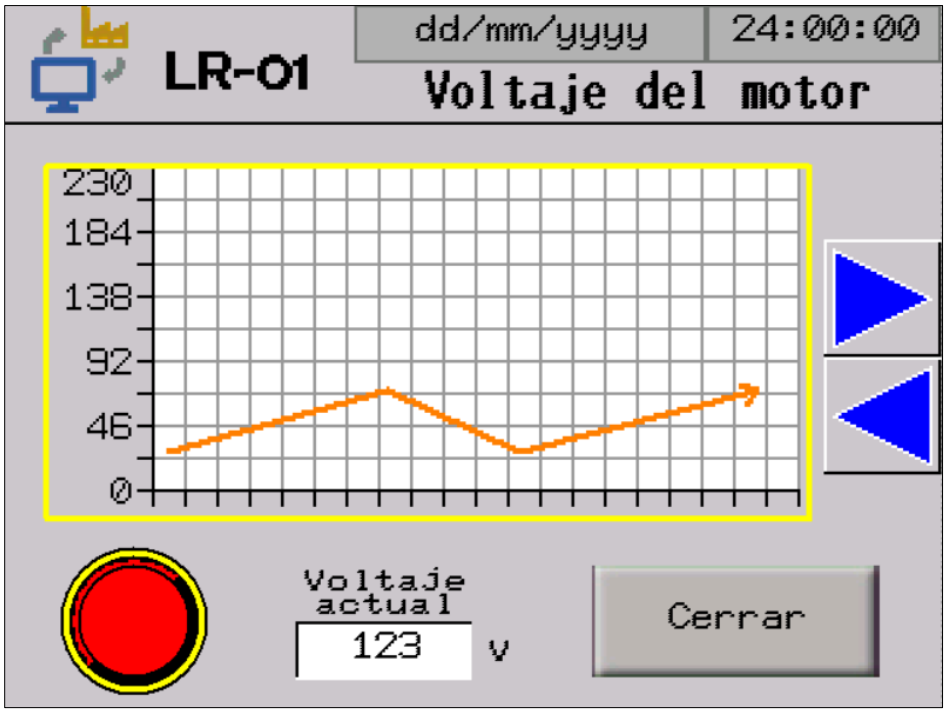


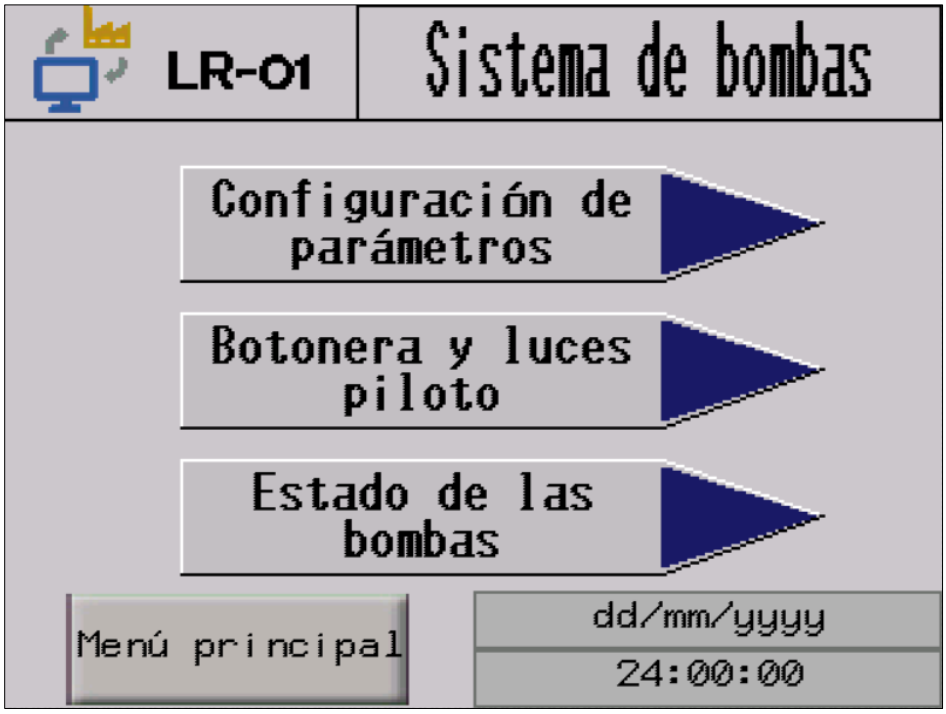
Figura 54

Interfaz – Gráfica de tendencia potencia del motor

Menú sistema de bombas. Al igual que en el menú para el variador de frecuencia, esta pantalla permite navegar entre los apartados para configurar los parámetros del sistema, controlar y monitorear el estado de la maqueta, y supervisar el funcionamiento del sistema de bombas controlado. En la Figura 55 se observa el diseño de esta pantalla.

Figura 55

Interfaz – Menú sistema de bombas



Parámetros sistema. La determinación de la demanda para el sistema y la configuración del controlador se realizar en la pantalla de la Figura 56. Primero, mediante el selector, se escoge entre un controlador PI o PID, luego se ingresan los parámetros del controlador, es decir, ganancia, tiempo integral y tiempo derivativo en caso de PID. Luego, en el lado derecho se ingresa la demanda del sistema en l/min. Finalmente, con el botón “Aplicar” de la parte inferior, se establecen estos parámetros para simular el sistema controlado.

Figura 56

Interfaz – Parámetros sistema

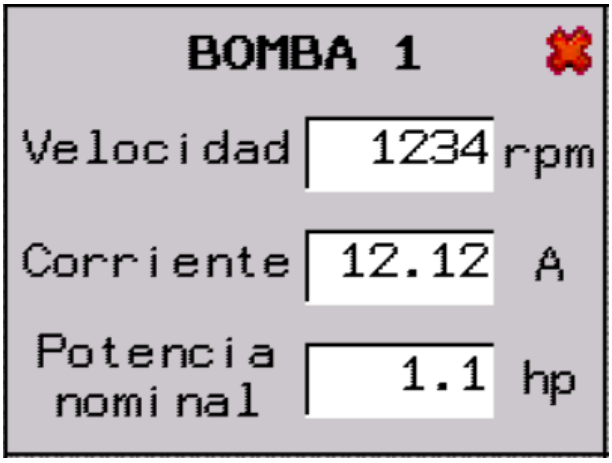
Estado de bombas. Interfaz para comprobar el estado de activación de las bombas. Como se observa en la Figura 57, existen 8 indicadores: los de la parte superior se encienden cuando la bomba en consideración está funcionando al 100%, mientras que uno de los inferiores se enciende para mostrar cual bomba es la que se está controlando para cubrir la demanda. Asociada a esta pantalla, se despliega la ventana emergente de la Figura 58 al dar clic en los íconos de las bombas; en ella se muestra la velocidad, corriente y potencia nominal de la bomba correspondiente.

Figura 57

Interfaz – Estado de bombas

Figura 58

Ventana emergente (Popup) – Estado de la bomba



Gráficas de tendencia del sistema de bombas. En las Figuras Figura 59 a Figura 61 se muestran las interfaces donde se crean líneas de tendencia de las variables del sistema de bombas controlada: velocidad bomba controlada, caudal y

presión. Todas las características expuestas para las interfaces del mismo tipo en el apartado de prueba del variador, también aplican en estas pantallas.

Figura 59

Interfaz – Gráfica de tendencia velocidad bomba

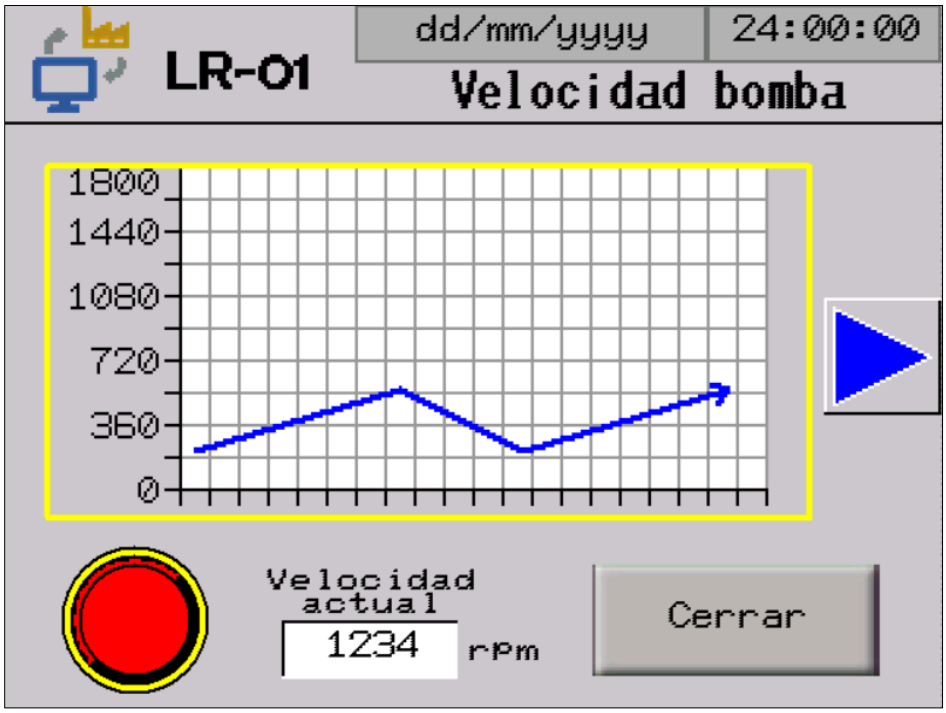


Figura 60

Interfaz – Gráfica de tendencia caudal

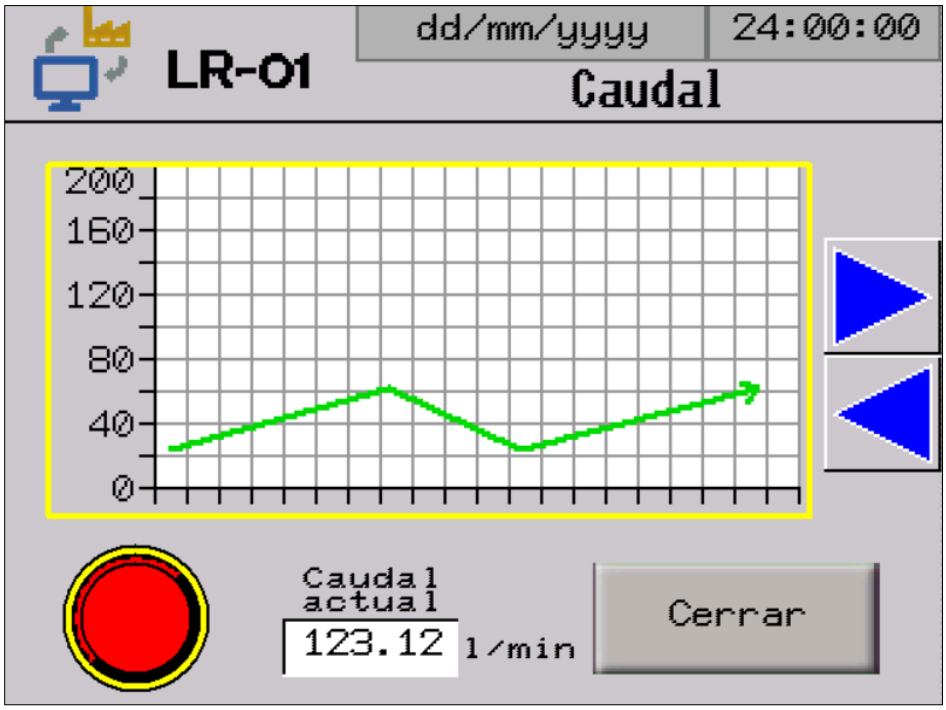
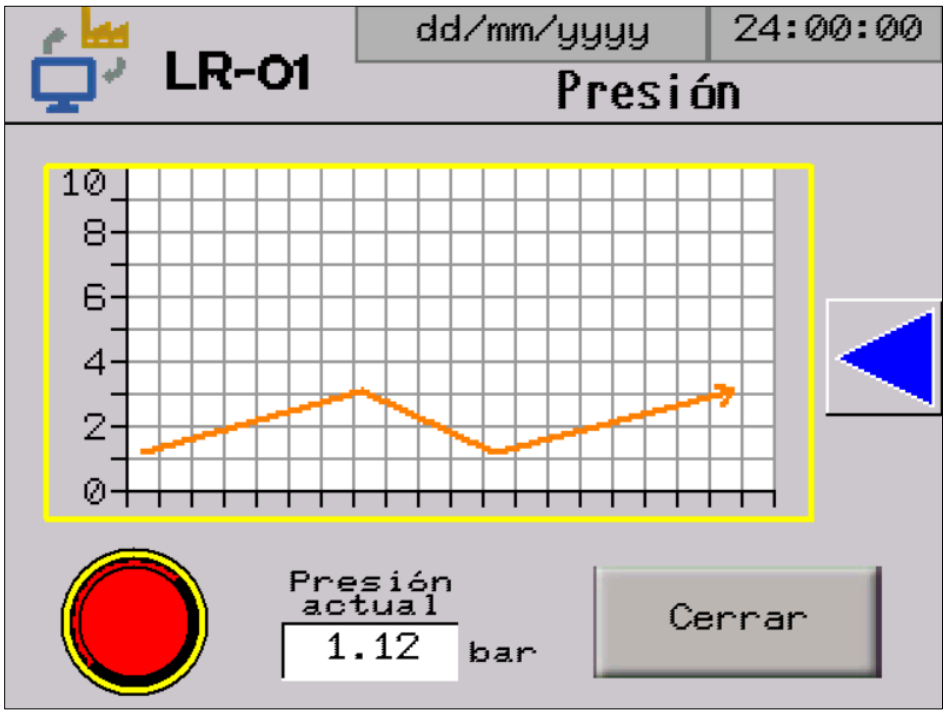


Figura 61

Interfaz – Gráfica de tendencia presión



Alarmas. Al momento de ocurrir algún evento fuera de lo normal, por ejemplo, si el paro de emergencia es accionado o si alguna de las variables monitoreadas se encuentra fuera del rango establecido, se activa una alarma la cual, se va registrando en la tabla de la interfaz que se muestra en la Figura 62. Además, al activarse una alarma, automáticamente aparece esta interfaz para mostrar al usuario de qué se trata el evento.

Figura 62

Interfaz - Alarmas

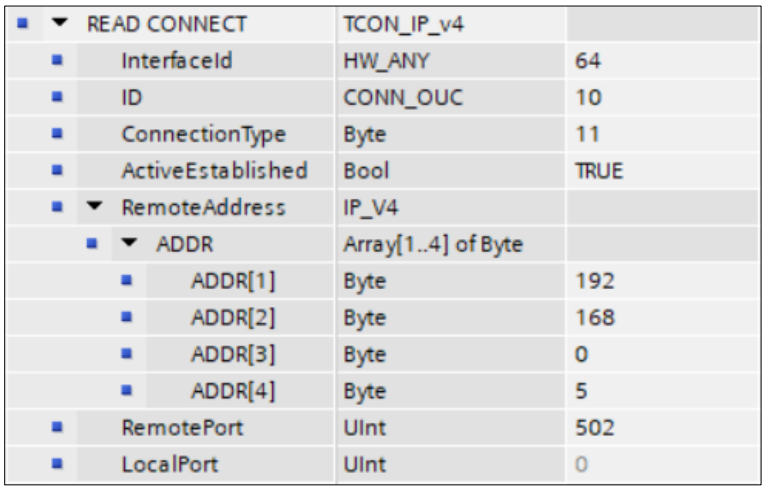
Conexión PLC y variador

Para establecer la comunicación a través de MODBUS TCP/IP, dentro de la programación en Ladder del PLC se utiliza el bloque MB_CLIENT el cual se encuentra dentro del set de instrucciones “Comunicación”, en la carpeta “MODBUS TCP” dentro de la carpeta “Otros”. Esta instrucción se comunica como un cliente MODBUS TCP a través de la conexión PROFINET de la CPU S7-1200.

La entrada “CONNECT” se emplea para configurar la conexión con el cliente, en el cual se debe referenciar una estructura TCON_IP_v4. En la Figura 63 se observa la estructura TCON_IP_v4 con la configuración realizada para crear una instancia de la instrucción MB_CLIENT. “InterfaceId” es el identificador de hardware de la interfaz local, es decir, de la interfaz PROFINET. En los PLC S7-1200 por lo general esta interfaz tiene el identificador 64. “ID” identifica la conexión dentro de la CPU. Cada instancia individual de la instrucción MB_CLIENT debe usar un ID único, por lo tanto, se debe crear una estructura TCON_IP_v4 por cada instrucción MB_CLIENT. “ConnectionType” establece el protocolo de transporte (TCP o UDP). En este bloque solo se permite TCP y se determina con el valor 11. “ActiveEstablished” indica la forma en que se establece la conexión, el cual siempre debe ser “TRUE”. En el arreglo dentro de la estructura IP_V4 “RemoteAddress” se coloca por bytes la dirección IP del servidor MODBUS, en este caso, el variador; esta dirección IP se observa en la Figura 18. En “RemotePort” se coloca el número de puerto IP del servidor al que el cliente establece la conexión y se comunica mediante el protocolo TCP/IP, por defecto el servidor utiliza el puerto 502. Por último, en “LocalPort” se asigna el número de puerto de conexión local, este puede estar entre 1 y 49151, aunque si se asigna un 0 esto quiere decir que puede utilizar cualquier puerto.

Figura 63

Estructura TCON_IP_v4 con la configuración para establecer comunicación con el variador mediante MODBUS TCP/IP



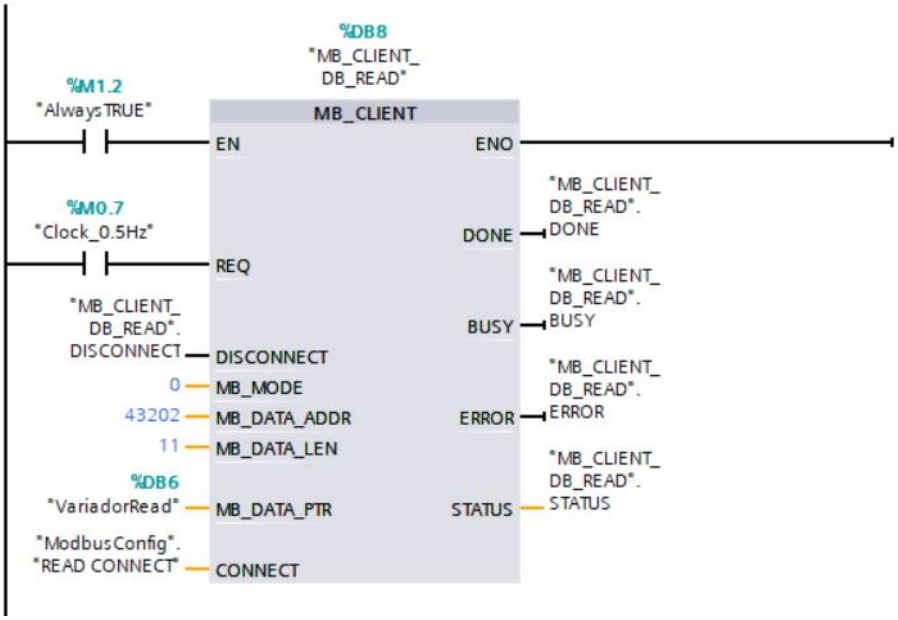
READ CONNECT	TCON_IP_v4	
InterfaceId	HW_ANY	64
ID	CONN_OUC	10
ConnectionType	Byte	11
ActiveEstablished	Bool	TRUE
RemoteAddress	IP_V4	
ADDR	Array[1..4] of Byte	
ADDR[1]	Byte	192
ADDR[2]	Byte	168
ADDR[3]	Byte	0
ADDR[4]	Byte	5
RemotePort	UInt	502
LocalPort	UInt	0

Se establecieron tres instancias de la instrucción MB_CLIENT: dos de lectura y uno de escritura; con lo cual, se crearon tres estructuras TCON_IP_V4 con la misma configuración mostrada en la Figura 63, cambiando únicamente el “ID”. En la Figura 64 se muestra una de las instancias de la instrucción MB_CLIENT. Adicional a la entrada “CONNECT” tenemos el habilitador de bloque “EN”, la entrada “REQ” que, cuando se encuentra en alto, realiza la lectura o escritura de datos. Con el parámetro “DISCONNECT” se controla el establecimiento y la terminación de la conexión al servidor MODBUS. En “MB_MODE” se define si se va a realizar una lectura “0” o escritura “1” de datos en el servidor. “MB_DATA_ADDR” contiene la dirección a partir de la cual la instrucción "MB_CLIENT" calcula la dirección remota; en MODBUS, al leer o escribir en registros de tenencia o *holding registers*, se direcciona a partir del 40001, es decir, si se desea leer a partir del registro 100, “MB_DATA_ADDR” tendría la dirección 40101. “MB_DATA_LEN” contiene el número de valores a leer o escribir donde cada número representa 2 bytes. Finalmente, en “MB_DATA_PTR” se referencia el bloque de datos del PLC de donde se va a enviar o donde se almacenará los datos del servidor.

A la salida del bloque se tiene bits de estado y un registro “STATUS” para comprobar si el envío o lectura de datos se realizó correctamente o qué tipo de error ocurrió.

Figura 64

Instancia de la instrucción MB_CLIENT para lectura de datos del variador



En las Figuras Figura 65 a Figura 67 se observa los datos declarados dentro del bloques “VariadorWrite”, “VariadorWrite2” y “VariadorRead”. Los dos primeros son utilizados para enviar datos al variador, mientras que el último se utiliza para leer determinados registros.

Figura 65

Bloque de datos “VariadorWrite”

VariadorWrite

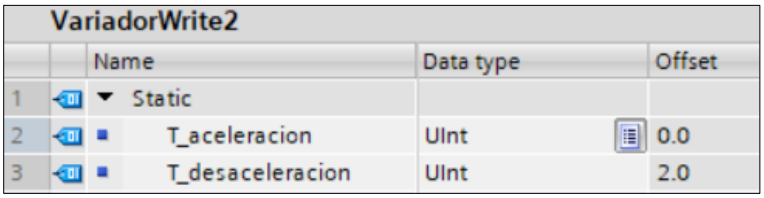
		Name	Data type	Offset
1		Static		
2		CMD	Int	0.0
3		SPFrecuencia	Int	2.0

Dentro del bloque “VariadorWrite” se define dos datos: “CMD” y “SPFrecuencia”. El primero corresponde al registro de comando. Es donde se configura el estado del

variador, por ejemplo, si se escribe “6” el variador pasa a estado READY, ingresando “15” arranca el motor, y con “7” el motor se detiene. El segundo registro sirve para ingresar la frecuencia de referencia para controlar la velocidad de giro del motor. Ambos registros son de tipo “Int” y las direcciones lógicas dentro del variador son 8501 y 8502, respectivamente, por lo tanto, en la instancia "MB_CLIENT" se ingresa en “MB_DATA_ADDR” el valor 48502.

Figura 66

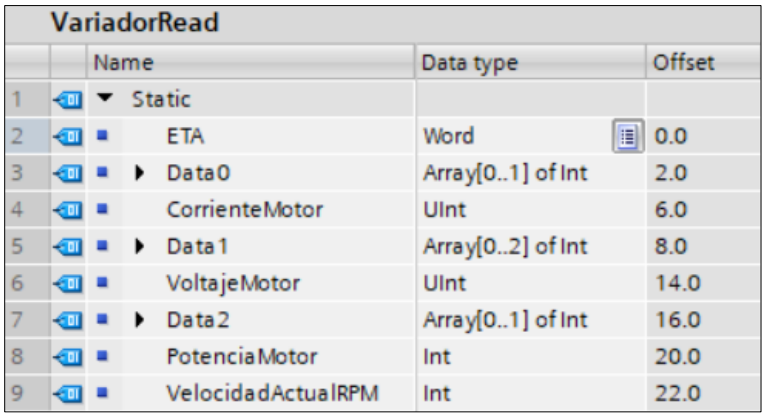
Bloque de datos “VariadorWrite2”



En el bloque “VariadorWrite2” se declara dos datos de tipo “UInt”: “T_aceleración” y “T_desaceleracion”. En estos registros se asigna los tiempos de aceleración y desaceleración al arrancar y detener el motor. El tiempo ingresado se configura al variador con un número decimal, es decir, si se asigna 10 en alguno de estos registros, el tiempo que tardará en hacer el cambio será de 1 segundo. Las direcciones lógicas de estos registros en el variador son 9001 y 9002, respectivamente, por lo tanto, en la instancia "MB_CLIENT" se ingresa en “MB_DATA_ADDR” el valor 49002.

Figura 67

Bloque de datos "VariadorRead"



VariadorRead

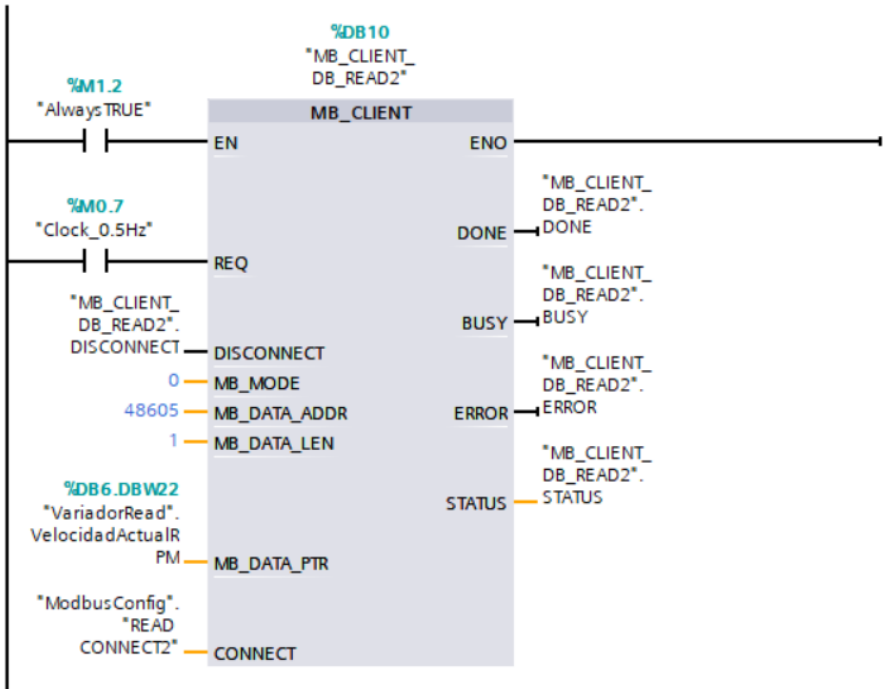
		Name	Data type	Offset
1		Static		
2		ETA	Word	0.0
3		Data0	Array[0..1] of Int	2.0
4		CorrienteMotor	UInt	6.0
5		Data1	Array[0..2] of Int	8.0
6		VoltajeMotor	UInt	14.0
7		Data2	Array[0..1] of Int	16.0
8		PotenciaMotor	Int	20.0
9		VelocidadActualRPM	Int	22.0

En el bloque "VariadorRead" se declaran todas las variables donde se recibirán datos provenientes del variador. "ETA" es el registro para leer el estado del variador, el tipo de dato es "Word" y su dirección lógica es 3201. "CorrienteMotor", "VoltajeMotor" y "PotenciaMotor" se asocia a los registros que reflejan el estado de funcionamiento del motor: corriente, voltaje y potencia. Las direcciones lógicas de estos registros en el mismo orden son: 3204, 3208 y 3211. Tanto la corriente como el voltaje son datos de tipo "UInt", mientras que potencia es tipo "Int". Finalmente, en "VelocidadActualRPM" se obtiene la velocidad de giro del motor en revoluciones por minuto. El tipo de dato es entero y la dirección lógica es 8604.

A excepción de "VelocidadActualRPM", todos los demás registros son leídos en la misma instancia "MB_CLIENT". Las direcciones de memoria intermedias se completan con arreglos tipo entero para coincidir con la dirección lógica necesaria. En la Figura 64 se observa la instrucción usada para esta lectura, y en la Figura 68 la instancia para obtener la velocidad del motor.

Figura 68

Instancia de la instrucción MB_CLIENT para leer la velocidad del motor



Conexión PLC y HMI

Como ya se mencionó en el capítulo anterior, la conexión entre el PLC y el panel de operador se da por medio del protocolo de comunicación S7 Ethernet. El primer paso es configurar al autómata y a la HMI en la misma red. Por esta razón, el PLC se asigna con la dirección 192.168.0.1 y el panel con la 192.168.0.155.

A continuación, es indispensable la configuración del software de diseño de la marca Schneider, Vijeo Designer. Para empezar, se añade un nuevo controlador en la sección de Administrador de E/S ubicado en el árbol del proyecto, luego se selecciona el protocolo SIAMTIC S7 Ethernet, el PLC S7-1200 CPU 1212C y se asigna su dirección IP. En las Figuras Figura 69 a Figura 71 se describen los pasos explicados.

Figura 69

Agregar un nuevo controlador en Administrador de E/S

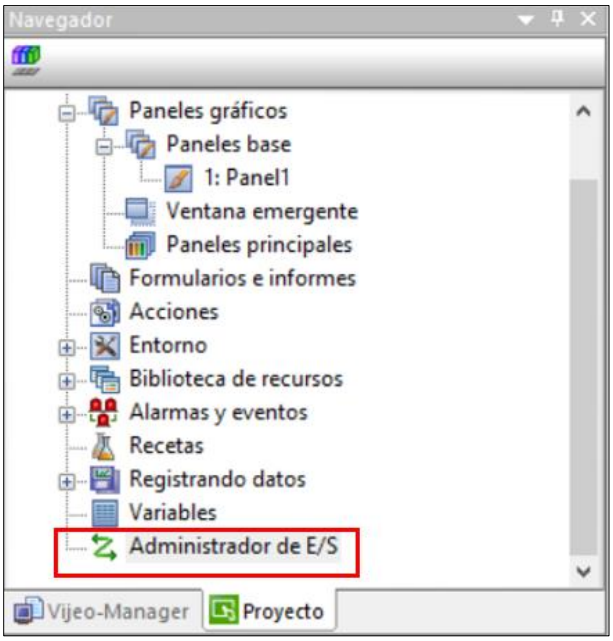


Figura 70

Selección del protocolo de comunicación y el PLC a conectar

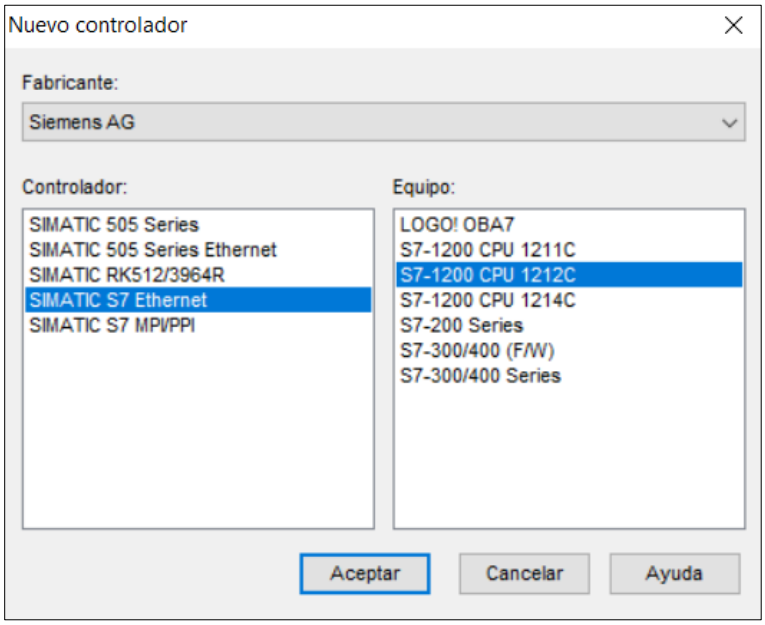
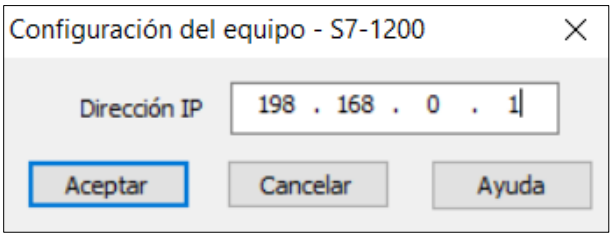


Figura 71

Asignación de dirección IP del PLC Siemens S7-1200

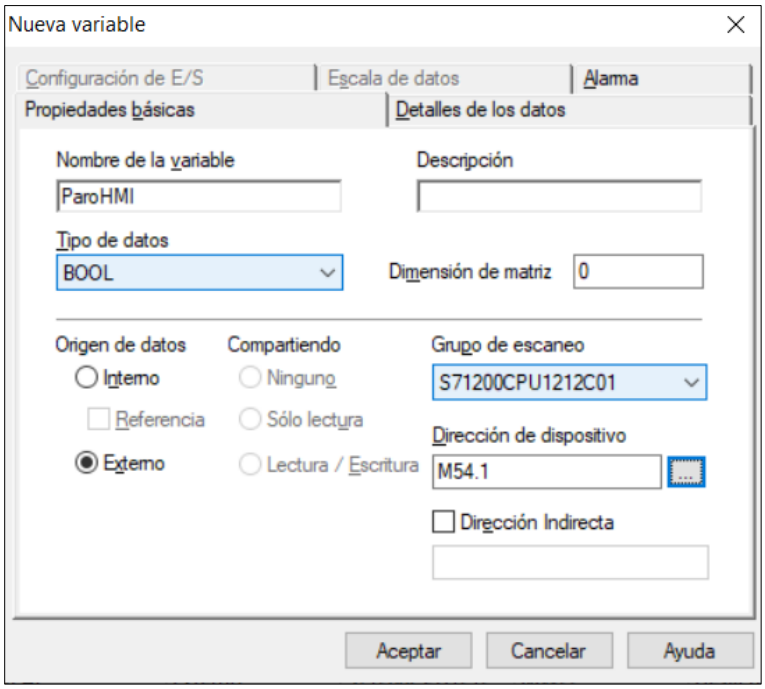


Una vez añadido y configurado el protocolo de comunicación, el controlador y su dirección IP, se ingresan todas las variables que se necesitan y están definidas en la programación del PLC. Cabe indicar, que el nombre de las variables en Vijeo Designer no necesariamente es el mismo que en el autómata, sin embargo, para el presente proyecto son similares. Cada variable ingresada en Vijeo se configura con su nombre, tipo de dato (booleano, int, uint, dint, udint, real, string y bloques), origen de datos, controlador de donde proviene y dirección en el dispositivo (puede ser de entrada, salida, memoria o bloque de datos).

Por ejemplo, y como se puede ver en la Figura 72, la variable cuya función es dar un paro al sistema desde el panel de operador se la nombra como ParoHMI, es de tipo BOOL, el origen de datos es externo, el controlador de donde proviene es el S71200CPU1212C y la dirección en el PLC es M54.1.

Figura 72

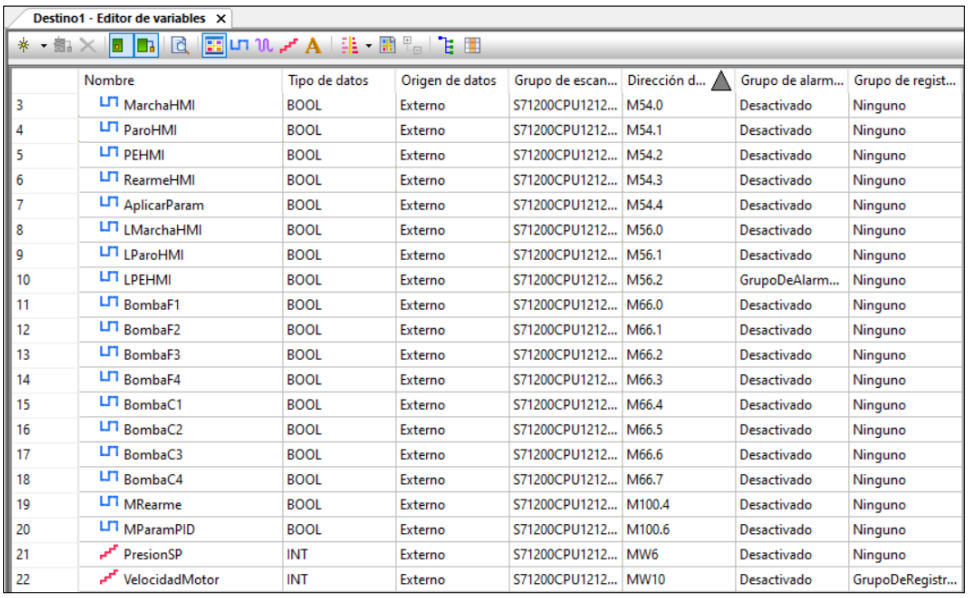
Creación de la variable ParoHMI en Vijeo Designer



La Figura 73 muestra el resultado de la declaración de las variables en Vijeo Designer que están asociadas con las del PLC a través de su dirección y el grupo de escaneo o controlador de donde proviene. Además, las variables pueden ser activadas para ser utilizadas como alarmas o registros. La primera sirve para asociar una variable a una alarma que es activada cuando se dé un evento, mientras que la segunda es necesaria para asociarla a un gráfico histórico.

Figura 73

Declaración de las variables a utilizar en la HMI asociadas con la dirección definida en el PLC



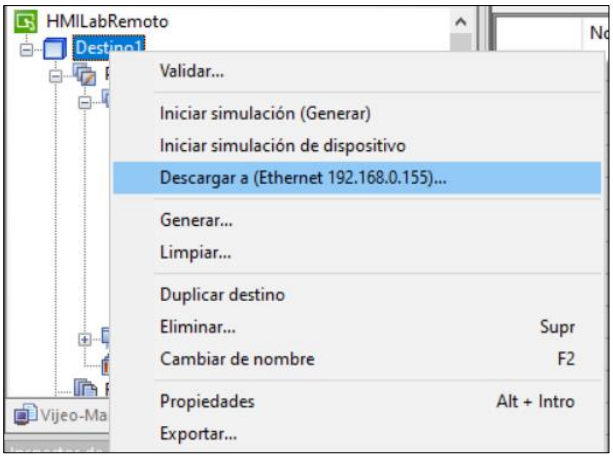
Destino1 - Editor de variables

	Nombre	Tipo de datos	Origen de datos	Grupo de escan...	Dirección d...	Grupo de alarm...	Grupo de regist...
3	MarchaHMI	BOOL	Externo	S71200CPU1212...	M54.0	Desactivado	Ninguno
4	ParoHMI	BOOL	Externo	S71200CPU1212...	M54.1	Desactivado	Ninguno
5	PEHMI	BOOL	Externo	S71200CPU1212...	M54.2	Desactivado	Ninguno
6	RearmeHMI	BOOL	Externo	S71200CPU1212...	M54.3	Desactivado	Ninguno
7	AplicarParam	BOOL	Externo	S71200CPU1212...	M54.4	Desactivado	Ninguno
8	LMarchaHMI	BOOL	Externo	S71200CPU1212...	M56.0	Desactivado	Ninguno
9	LParoHMI	BOOL	Externo	S71200CPU1212...	M56.1	Desactivado	Ninguno
10	LPEHMI	BOOL	Externo	S71200CPU1212...	M56.2	GrupoDeAlarm...	Ninguno
11	BombaF1	BOOL	Externo	S71200CPU1212...	M66.0	Desactivado	Ninguno
12	BombaF2	BOOL	Externo	S71200CPU1212...	M66.1	Desactivado	Ninguno
13	BombaF3	BOOL	Externo	S71200CPU1212...	M66.2	Desactivado	Ninguno
14	BombaF4	BOOL	Externo	S71200CPU1212...	M66.3	Desactivado	Ninguno
15	BombaC1	BOOL	Externo	S71200CPU1212...	M66.4	Desactivado	Ninguno
16	BombaC2	BOOL	Externo	S71200CPU1212...	M66.5	Desactivado	Ninguno
17	BombaC3	BOOL	Externo	S71200CPU1212...	M66.6	Desactivado	Ninguno
18	BombaC4	BOOL	Externo	S71200CPU1212...	M66.7	Desactivado	Ninguno
19	MRearme	BOOL	Externo	S71200CPU1212...	M100.4	Desactivado	Ninguno
20	MParamPID	BOOL	Externo	S71200CPU1212...	M100.6	Desactivado	Ninguno
21	PresionSP	INT	Externo	S71200CPU1212...	MW6	Desactivado	Ninguno
22	VelocidadMotor	INT	Externo	S71200CPU1212...	MW10	Desactivado	GrupoDeRegistr...

La comunicación entre el PLC y la HMI se logra una vez que ambos componentes estén conectados mediante cable de red a través de un switch y cuando el programa diseñado en Vijeo Designer sea cargado a la HMI que está implementada en la maqueta. Para cargar el proyecto, se debe dar clic derecho en “Destino1” ubicado en el árbol del proyecto y seleccionar la opción “Descargar a (Ethernet 192.168.0.155)” tal y como muestra la Figura 74.

Figura 74

Instrucciones para descargar el proyecto diseñado en Vijeo en la HMI



Evidentemente, el programa diseñado en el PLC debe estar cargado para que todas las variables que se declaran en Vijeo Designer estén asociadas mediante sus direcciones con las variables utilizadas en el PLC.

Conexión PLC y Raspberry Pi

Antes de describir la utilización de la librería python-snap7 para la comunicación entre el PLC S7-1200 y la Raspberry Pi 4, se crea un entorno virtual para manejar de forma independiente todas las librerías que se ocupan para el laboratorio remoto. La Raspberry Pi utiliza el sistema operativo Raspberry Pi OS, el cual es la distribución oficial de GNU/Linux basado en Debian de Raspberry. En Raspberry Pi OS viene instalado por defecto Python 2 y 3. Las versiones instaladas son la 2.7.16 y 3.7.3., siendo esta última la utilizada dado que Python 2 ya se encuentra descontinuado y muchas librerías no están disponibles para esta versión.

Para crear un entorno virtual, primero se instala la librería “venv” corriendo el siguiente comando en el terminal:

python3 -m venv ~/my_venv

luego se crea un entorno virtual dentro de la carpeta my_venv en el directorio de inicio del usuario actual ejecutando el siguiente comando:

python3 -m venv ~/my_venv

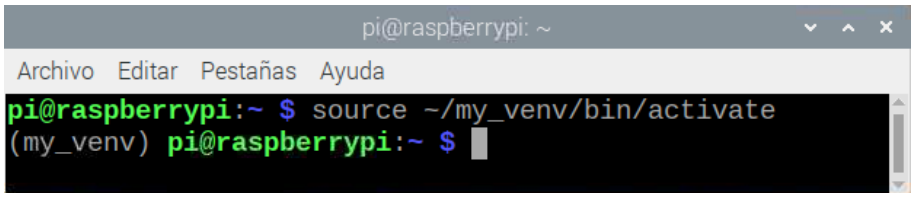
Una vez creado el entorno virtual, hay que activar este entorno para instalar los paquetes y correr los scripts de Python, para lo cual, se ejecuta la siguiente línea de comando:

source ~/my_venv/bin/activate

si el nombre del entorno virtual aparece entre paréntesis junto a la línea de comando tal y como se muestra en la Figura 75, significa que el entorno virtual está activado.

Figura 75

Activación del entorno virtual en la Raspberry Pi



Para utilizar “python-snap7” primero hay que instalar la librería en el entorno, para lo cual, se ingresa la siguiente línea de comando:

pip install python-snap7

Se utiliza la versión 1.1 para el desarrollo del programa para la lectura y escritura de datos del PLC.

Con “python-snap7” se va a crear un cliente Snap7 para que la comunicación con el PLC pueda ser controlada desde Python, para lo cual, con el código mostrado en la Figura 76 se crea el cliente y se conecta con el PLC utilizando la dirección IP, el RACK y el SLOT en el que se encuentra.

Figura 76

Creación del cliente Snap7 y conexión con el PLC

```
import snap7

IP = '192.168.0.1'
RACK = 0
SLOT = 1

plc = snap7.client.Client()
plc.connect(IP, RACK, SLOT)
```

Creada la conexión con el PLC, se utiliza la función “read_area” para leer datos que pueden ser entradas, salidas, datos de memoria, bloques de datos, temporizadores o contadores. En la función se pasan cuatro parámetros:

- Área de lectura, el cual puede ser uno de los 6 indicados.
- Número de bloque de datos DB, en caso de que no se vaya a leer un bloque de datos se pasa 0.
- Índice de bytes para empezar a leer.
- Número de bytes a leer.

La función devuelve un buffer con todos los datos leídos del PLC, sin embargo, aún falta separar las variables y transformar al tipo de dato correspondiente, para lo cual, “python-snap7” también ofrece funciones para ello dentro del módulo “util”. Por ejemplo, existen las funciones “get_bool”, “get_int” y “get_real”, las cuales obtienen un boleano, un entero y un real, respectivamente, tomando como parámetros el buffer de datos leídos y la dirección de inicio de la variable dentro del buffer.

Con respecto a la escritura dentro del PLC, existe la función “write_area” la cual funciona de manera análoga a la función de lectura, con la única diferencia de que, como parámetro, no se indica el número de bytes a escribir, sino se pasa el buffer de datos. Para introducir un dato en el buffer, también se dispone de funciones, en este

caso “set_bool”, “set_int”, “set_real”, entre otros. Los parámetros que se pasan a estas funciones son las siguientes:

- Buffer donde se insertará datos.
- Índice del byte desde el que empezar a escribir.
- Valor a escribir en el buffer.

En las Figuras Figura 77 y Figura 78 se muestra el programa completo para intercambiar datos entre la Raspberry Pi y el PLC. Todas las variables que se leen o escriben en el PLC, junto con su dirección lógica se indican en la Tabla 13.

Tabla 13

Tabla de variables del PLC que intervienen en la comunicación con la Raspberry Pi

Acción	Símbolo	Dirección	Tipo
Lectura	LReadyPP	M52.0	Bool
	CaudalTotal(l/min)	MD62	Real
	Bombas100/C	MB66	Byte
	velocidadB1	MW92	Int
	velocidadB2	MW94	Int
	velocidadB3	MW96	Int
	velocidadB4	MW98	Int
	PresionBomba(bar)	MD114	Real
	CorrienteMotor	DB6: 6.0	UInt
	VoltajeMotor	DB6: 14.0	UInt
	PotenciaMotor	DB6: 20.0	Int
	VelocidadActualRPM	DB6: 22.0	Int
Escritura	Configurar	M2.0	Bool
	Arrancar	M2.1	Bool

Acción	Símbolo	Dirección	Tipo
	Detener	M2.2	Bool
	PI/PID	M8.0	Bool
	Demanda(l/min)	MW12	Int
	Kp	MD14	Real
	Ti	MD18	Real
	Td	MD22	Real
	Maqueta	MW26	Int
	SPVariador	MW28	Int
	T_aceleracion	MW30	Int
	T_desaceleracion	MW32	Int
	MarchaPP	M140.0	Bool
	ParoPP	M140.1	Bool
	aplicarParam	M140.2	Bool

Figura 77

Programa en Python para leer datos del PLC Siemens S7-1200

```
from snap7.util import get_bool, get_byte, get_int, \
    get_real, get_uint, set_bool, set_int, set_real

# OBTENCION DE DATOS DESDE EL PLC
data = plc.read_area(snap7.types.Areas.MK, 0, 52, 1)
LReadyPP = get_bool(data, 0, 0)

data = plc.read_area(snap7.types.Areas.MK, 0, 62, 5)
CaudalTotal = get_real(data, 0)
EstadoBombas = get_byte(data, 4)
bombasON = format(EstadoBombas,"#010b")

data = plc.read_area(snap7.types.Areas.MK, 0, 92, 8)
velocidadB1 = get_int(data, 0)
velocidadB2 = get_int(data, 2)
velocidadB3 = get_int(data, 4)
velocidadB4 = get_int(data, 6)

data = plc.read_area(snap7.types.Areas.MK, 0, 114, 4)
PresionBomba = get_real(data, 0)

data = plc.read_area(snap7.types.Areas.DB, 6, 6, 18)
CorrienteMotor = get_uint(data, 0)
VoltajeMotor = get_uint(data, 8)
PotenciaMotor = get_int(data, 14)
VelocidadActualRPM = get_int(data, 16)
```

Figura 78

Programa en Python para enviar datos al PLC Siemens S7-1200

```
# ENVIO DE DATOS AL PLC
buffer = bytearray([0])
set_bool(buffer, 0, 0, Configurar)
set_bool(buffer, 0, 1, Arrancar)
set_bool(buffer, 0, 2, Detener)
plc.write_area(snap7.types.Areas.MK, 0, 2, buffer)

buffer = bytearray([0])
set_bool(buffer, 0, 0, selectorPID)
plc.write_area(snap7.types.Areas.MK, 0, 8, buffer)

buffer = bytearray([0]*22)
set_int(buffer, 0, Demanda)
set_real(buffer, 2, Kp)
set_real(buffer, 6, Ti)
set_real(buffer, 10, Td)
set_int(buffer, 14, Maqueta)
set_int(buffer, 16, SPVariador*10)
set_int(buffer, 18, T_aceleracion*10)
set_int(buffer, 20, T_desaceleracion*10)
plc.write_area(snap7.types.Areas.MK, 0, 12, buffer)

buffer = bytearray([0])
set_bool(buffer, 0, 0, MarchaPP)
set_bool(buffer, 0, 1, ParoPP)
set_bool(buffer, 0, 2, aplicarParam)
plc.write_area(snap7.types.Areas.MK, 0, 140, buffer)
```

Aplicación web para el cliente

La aplicación web será la interfaz que permita al usuario llevar a cabo alguna de las prácticas utilizando la maqueta de forma remota. De acuerdo a las funcionalidades definidas para la aplicación web, se desarrolla la misma basado en las herramientas que brinda la librería de Python: Django. A continuación, se indica el procedimiento para

instalar y crear un proyecto en Django, para después, mostrar el desarrollo realizado en el diseño de la parte visual y lógica de la aplicación.

Creación de la aplicación Django

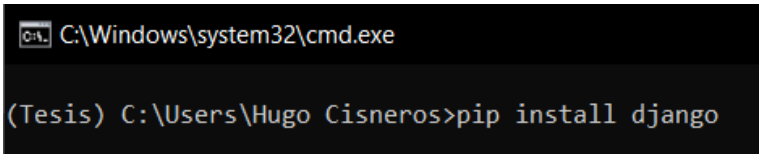
Django es un *framework* de aplicaciones web escrito en Python, de código abierto y acceso gratuito. Por lo tanto, para implementar una aplicación, el primer paso es la creación de un entorno virtual con una determinada versión de Python, que permita manejar de forma independiente todas las librerías requeridas.

En consecuencia, para el presente trabajo se crea un entorno virtual en la plataforma de distribución de Python conocida como Anaconda y que dispone de la versión 3.8.12 de este lenguaje de programación. El nombre del entorno virtual es “Tesis”.

La creación de la aplicación es manejada desde un terminal en el cual está activo el entorno virtual creado anteriormente. A partir de ahí, el primer paso es instalar Django tal y como muestra la Figura 79.

Figura 79

Comando para instalar Django en el entorno virtual



Después, es importante dirigirse mediante el comando *cd* a un directorio cualquiera del ordenador para que se almacene toda la estructura del proyecto. A continuación, se crea el proyecto Django bajo el nombre de “labremoto” ejecutando el siguiente comando:

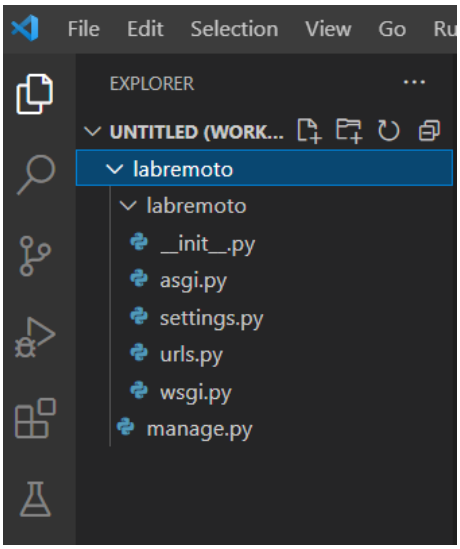
django-admin startproject labremoto

Por medio del editor Visual Studio Code, se puede visualizar el resultado de la ejecución del comando anterior tal y como se puede ver en la Figura 80, destacando la

creación del fichero “manage.py” que va a permitir interactuar, configurar, modificar, actualizar y ejecutar a la aplicación y a sus componentes como las bases de datos.

Figura 80

Estructura resultante después de crear el proyecto Django



Además, se crean los siguientes ficheros:

- *_init_.py:* fichero vacío que permite a Django reconocer a la carpeta total como un paquete Python (MDN Web Docs, 2022).
- *asgi.py:* fichero para que los servidores web compatibles con ASGI puedan ejecutarse de forma asíncrona (Universidad Rey Juan Carlos, s.f.).
- *settings.py:* se refiere a las configuraciones del proyecto.
- *urls.py:* son las direcciones URL del proyecto y la aplicación. Cada URL tiene una función específica.
- *wsgi.py:* fichero para que los servidores web compatibles con WSGI puedan ejecutarse.

A continuación, mediante el comando *cd* es indispensable dirigirse al directorio donde se encuentre el fichero “manage.py”, ya que, a partir de la ejecución de este, se

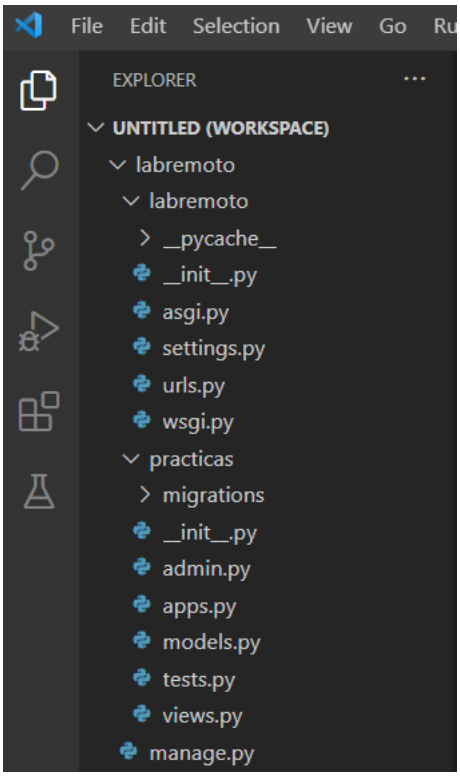
procede a crear la aplicación bajo el nombre “practicas” dentro del proyecto ejecutando el siguiente comando:

python manage.py startapp practicas

Dando como resultado la estructura del proyecto que muestra la Figura 81.

Figura 81

Árbol del proyecto después de crear la aplicación Django



Los ficheros que se crean en la aplicación a tomar en cuenta son:

- *models.py:* fichero donde se crean los modelos de la aplicación. Estos son definidos bajo la estructura de tablas y campos, los cuales van a ser accedidos y modificados según los eventos que se manifiesten en la aplicación.
- *admin.py:* fichero que permite registrar los modelos creados.

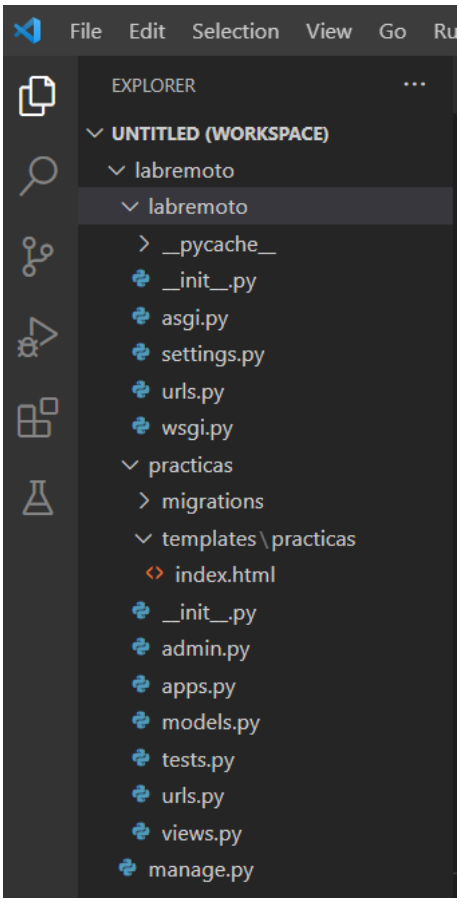
- *views.py:* fichero que almacena las funciones o clases que determinan el funcionamiento de la aplicación. Cada función o clase está asociada con una dirección URL.

Además, dentro de la carpeta “practicas” que corresponde a la aplicación, se crea un fichero adicional llamado “urls.py” el cual permite llamar a las funciones implementadas en el fichero “views.py”. Adicionalmente, se crea una carpeta llamada “templates/practicas” en donde se almacenan todas las páginas HTML diseñadas y llamadas acorde a lo programado por los desarrolladores de la aplicación.

En definitiva, el árbol del proyecto y específicamente de la aplicación del laboratorio remoto queda definida como muestra la Figura 82.

Figura 82

Árbol del proyecto y aplicación del laboratorio remoto



Modelos

Un modelo es la característica incorporada que utiliza Django para crear tablas, campos, objetos y restricciones. En otras palabras, los modelos de Django funcionan de forma análoga a SQL para las bases de datos. Por un lado, SQL o lenguaje de consulta estructurado, es complejo e implica muchas consultas diferentes para crear, eliminar, actualizar o cualquier otra acción para manejas las bases de datos. En cambio, los modelos de Django simplifican las tareas y organizan las tablas de manera similar a como los lenguajes orientados a objetos declaran las clases y atributos. Generalmente, cada modelo se asigna a una sola tabla de base de datos (Arora, 2019).

Todos los modelos son definidos en el archivo “models.py”. En la Figura 83 se muestra un ejemplo de la declaración de un modelo con tres campos para la base de datos. El modelo es una clase de Python que subclasifica “django.db.models.Model”. Cada atributo del modelo representa un campo de la base de datos, estableciendo el tipo de dato a partir del módulo “models”. Django mapea los campos definidos en los modelos en campos de tabla de la base de datos como se muestra en la Figura 84.

Figura 83

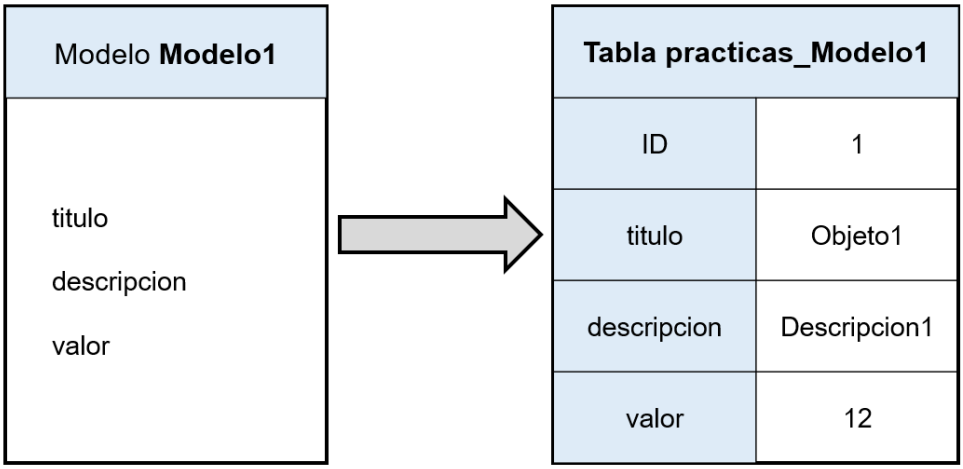
Ejemplo de la declaración de un modelo

```
from django.db import models

class Modelo1(models.Model):
    titulo = models.CharField(max_length = 200)
    descripcion = models.TextField()
    valor = models.IntegerField()
```

Figura 84

Representación de los modelos en Django para generar las tablas en la base de datos



Cada vez que se crea o elimina un modelo, o se modifica alguno de sus campos, es necesitamos ejecutar dos comandos “makemigrations” y “migrate”. Al ejecutar “makemigrations” se generan los comandos SQL para las aplicaciones preinstaladas (declaradas en el archivo “settings.py”) y los modelos creados para la aplicación; mientras que “migrate” ejecuta esos comandos SQL para generar las tablas en el archivo de la base de datos.

Figura 85

Modelos de la aplicación 1/3

```
class Usuario(models.Model):
    nombres = models.CharField(max_length=50)
    apellidos = models.CharField(max_length=50)
    usuario = models.CharField(max_length=30)
    clave = models.CharField(max_length=30)
    practica = models.BooleanField()
    horario = models.DateTimeField()
    num_practica = models.IntegerField()
    cuenta_profesor = models.BooleanField()

    def __str__(self):
        return self.usuario

class Horario(models.Model):
    dia = models.DateField()
    hora = models.TimeField()
    franja = models.IntegerField()
    num_practica = models.IntegerField()

class Practica(models.Model):
    imagen = models.ImageField(upload_to='images/')
    titulo = models.CharField(max_length=100)
    descripcion = models.CharField(max_length=1000)

    def __str__(self):
        return self.titulo

class Anuncio(models.Model):
    anuncio_Practica1 = models.TextField()
    anuncio_Practica2 = models.TextField()
    anuncio_Practica3 = models.TextField()
```

A continuación, en las Figuras Figura 85 a Figura 87 se muestran los modelos utilizados en la aplicación web. El modelo “Usuario” es usado para registrar los datos de usuarios, a fin de que puedan tener acceso a la aplicación para reservar y realizar

prácticas, además de permite controlar el acceso único al manejo de la maqueta. Los campos definidos para este modelo son:

- nombres: nombres del usuario. Máximo 50 caracteres.
- apellidos: apellidos del usuario. Máximo 50 caracteres.
- usuario: identificador único para el usuario. Máximo 30 caracteres.
- clave: clave de acceso del usuario.
- practica: campo booleano para constatar si el usuario tiene reservado un horario.
- horario: campo de fecha y hora en la cual el usuario reservó la práctica
- num_practica: identificador entero de la práctica reservada por el usuario (1, 2 o 3).

El modelo “Horario” permite llevar el registro de las prácticas agendadas, con el fin de impedir que dos usuarios escojan los mismos horarios y puedan manipular la maqueta de forma simultánea. Los campos definidos para este modelo son:

- dia: fecha reservada.
- hora: hora reservada.
- franja: identificador entero de la jornada escogida (7:15 a 9:15 = 0, 9:30 a 11:30 = 1, 12:00 a 14:00 = 2, 14:00 a 16:00 = 3).
- num_practica: identificador entero de la práctica reservada (1, 2 o 3).

En el modelo “Practica” se almacena una reseña de las prácticas elaboradas. Su propósito es brindar la capacidad de agregar nuevas prácticas al modelo, y que estos se muestren en el listado y descripción de prácticas sin necesidad de modificar los archivos HTML. Los campos definidos para este modelo son:

- imagen: almacena la ubicación dentro del directorio donde se la imagen relacionada a la práctica.

- titulo: título de la práctica
- descripcion: descripción corta de la práctica.

El modelo “Anuncio” es utilizado para desplegar indicaciones por parte del docente al estudiante durante el desarrollo de la práctica. Este modelo tiene los siguientes campos:

- anuncio_Practica1: indicaciones para la practica 1.
- anuncio_Practica2: indicaciones para la practica 2.
- anuncio_Practica3: indicaciones para la practica 3.

Figura 86

Modelos de la aplicación 2/3

```
class Escritura(models.Model):
    # Escritura
    maqueta = models.IntegerField()
    SPfrecuencia = models.FloatField()
    Taceleracion = models.FloatField()
    Tdesaceleracion = models.FloatField()
    SPdemanda = models.IntegerField()
    selectorPID = models.BooleanField()
    kp = models.FloatField()
    ti = models.FloatField()
    td = models.FloatField()
    configurarParam = models.BooleanField()
    configurar = models.BooleanField()
    arrancar = models.BooleanField()
    detener = models.BooleanField()
    marchaPP = models.BooleanField()
    paroPP = models.BooleanField()
```

El modelo “Escritura” declara los campos que la aplicación web modifica en la base de datos para controlar el funcionamiento de la maqueta. Mientras que, el modelo “Lectura”, contiene las variables que la aplicación web utiliza para conocer el estado de la maqueta y parámetros del motor y del sistema. Los campos declarados en ambos modelos son los mismos que se muestran en la Tabla 13.

Figura 87

Modelos de la aplicación 3/3

```
class Lectura(models.Model):
    # Lectura
    ready = models.BooleanField()
    # Practica1
    corriente = models.FloatField()
    voltaje = models.IntegerField()
    potencia = models.FloatField()
    velocidad = models.IntegerField()
    # Practicas 2 y 3
    presion = models.FloatField()
    demanda = models.FloatField()
    bombaF1 = models.BooleanField()
    bombaF2 = models.BooleanField()
    bombaF3 = models.BooleanField()
    bombaF4 = models.BooleanField()
    bombaC1 = models.BooleanField()
    bombaC2 = models.BooleanField()
    bombaC3 = models.BooleanField()
    bombaC4 = models.BooleanField()
    vel1 = models.IntegerField()
    vel2 = models.IntegerField()
    vel3 = models.IntegerField()
    vel4 = models.IntegerField()
```

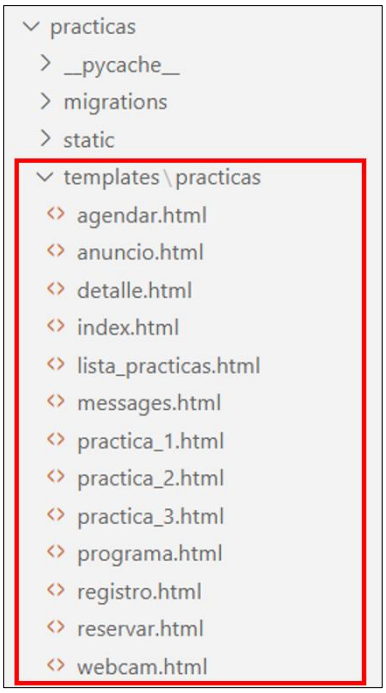
Templates

En una aplicación Django, los *templates* son los archivos que son programados en lenguaje HTML. Además, acorde a la programación desarrollada, se despliegan como páginas web y son aquellas ventanas en las cuales los clientes pueden interactuar con la aplicación del laboratorio remoto.

A continuación, en la Figura 88 se muestra todos los archivos .html desarrollados en el presente proyecto.

Figura 88

Templates utilizados en la aplicación



- **agendar.html:** esta ventana permite al estudiante visualizar los horarios disponibles para luego agendar o cambiar uno de ellos. Por otro lado, el profesor solo puede observar el calendario de los horarios disponibles y reservados. La Figura 89 muestra la interfaz que se despliega para agendar o cambiar una práctica y la Figura 90 la ventana a la que solo tiene acceso el docente.

Figura 89

Template para agendar o cambiar una práctica

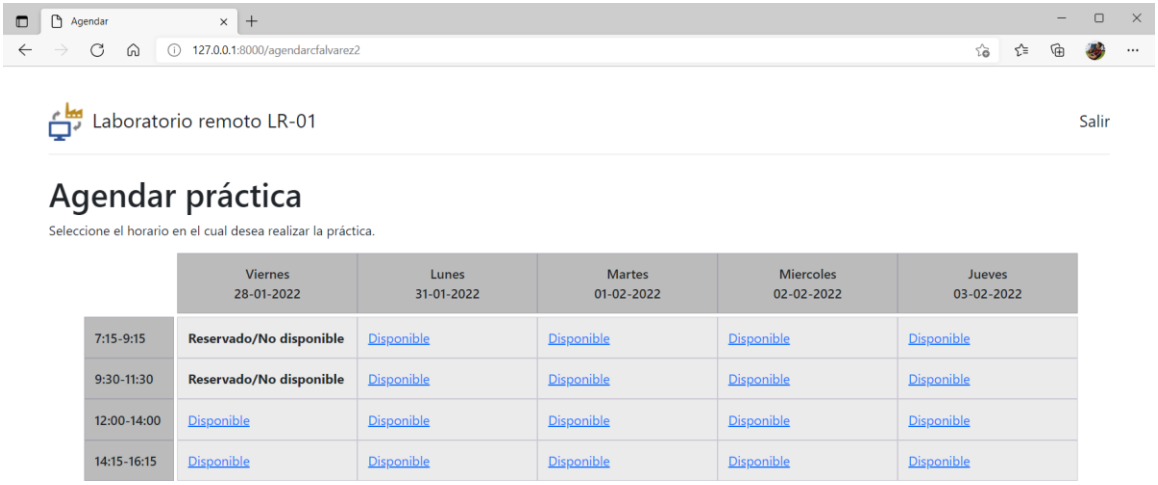
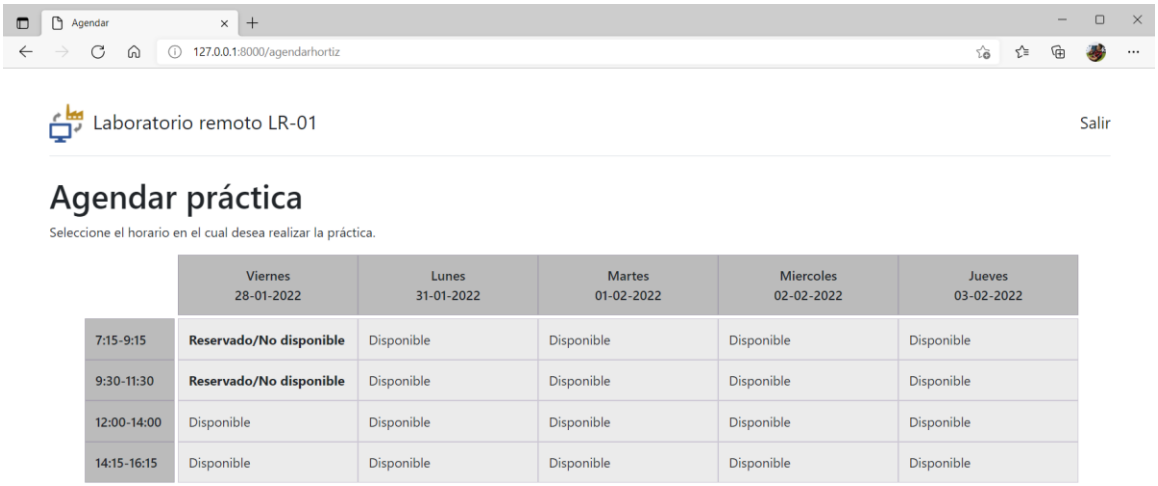


Figura 90

Template para visualizar las prácticas agendadas desde el usuario de docente

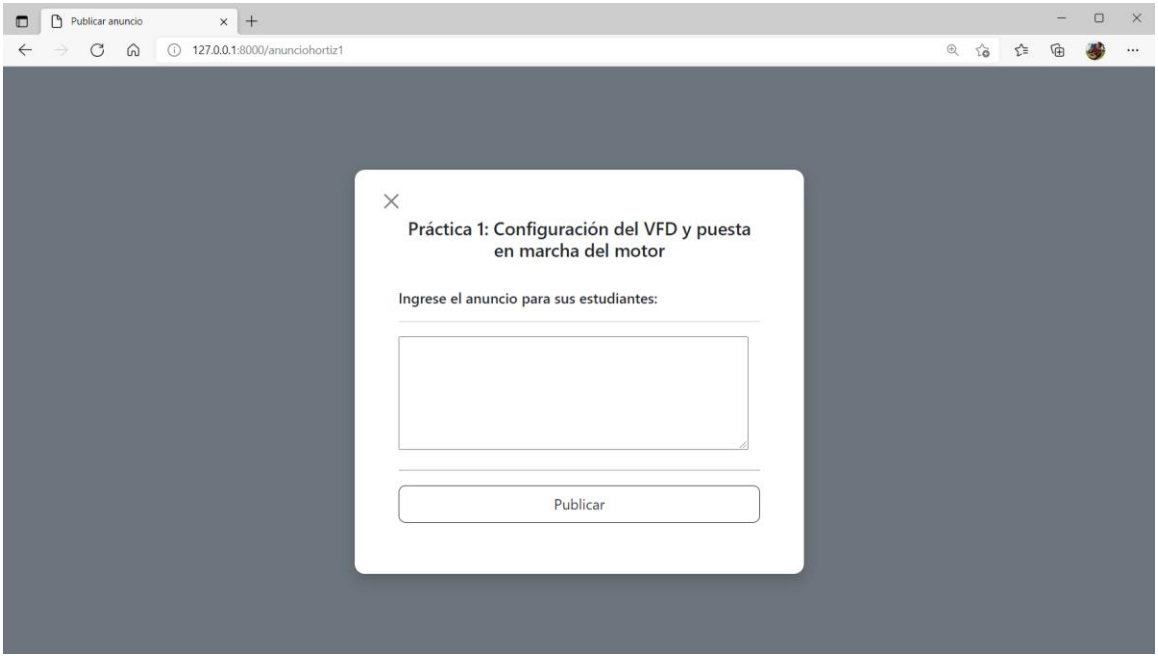


- **anuncio.html:** a esta página web solo es posible acceder a través de la cuenta del profesor. Dentro de ella, puede publicar anuncios para cada una de las tres prácticas, de tal forma que los estudiantes cuando inicien el

experimento remoto puedan leer comentarios, recomendaciones o avisos publicados por el docente. La Figura 91 muestra la ventana descrita.

Figura 91

Template para que el instructor publique un anuncio



- **detalle.html:** tanto el alumno como el docente pueden ingresar en una ventana que describa a las tres prácticas de laboratorio implementadas tal y como muestra la Figura 92.

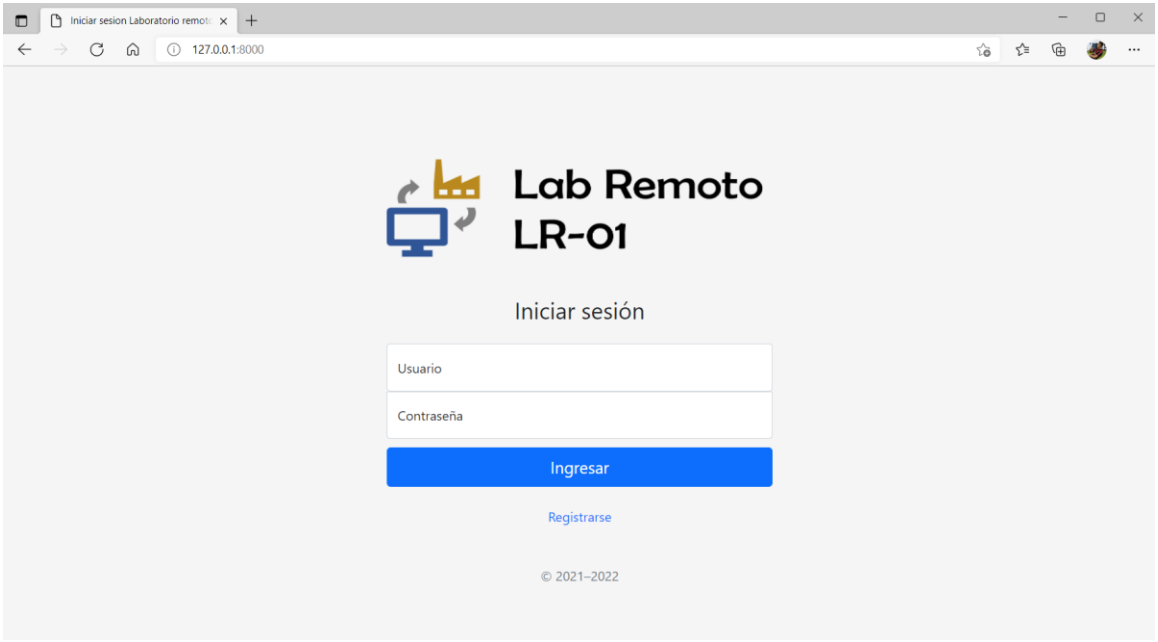
Figura 92

Template que se despliega para describir la información de una práctica

- **index.html:** es la ventana inicial de la aplicación tal y como muestra la Figura 93. En esta, el estudiante o profesor ingresan su nombre de usuario y contraseña para acceder al laboratorio remoto. Además, pueden acceder a otra ventana para registrar un nuevo usuario.

Figura 93

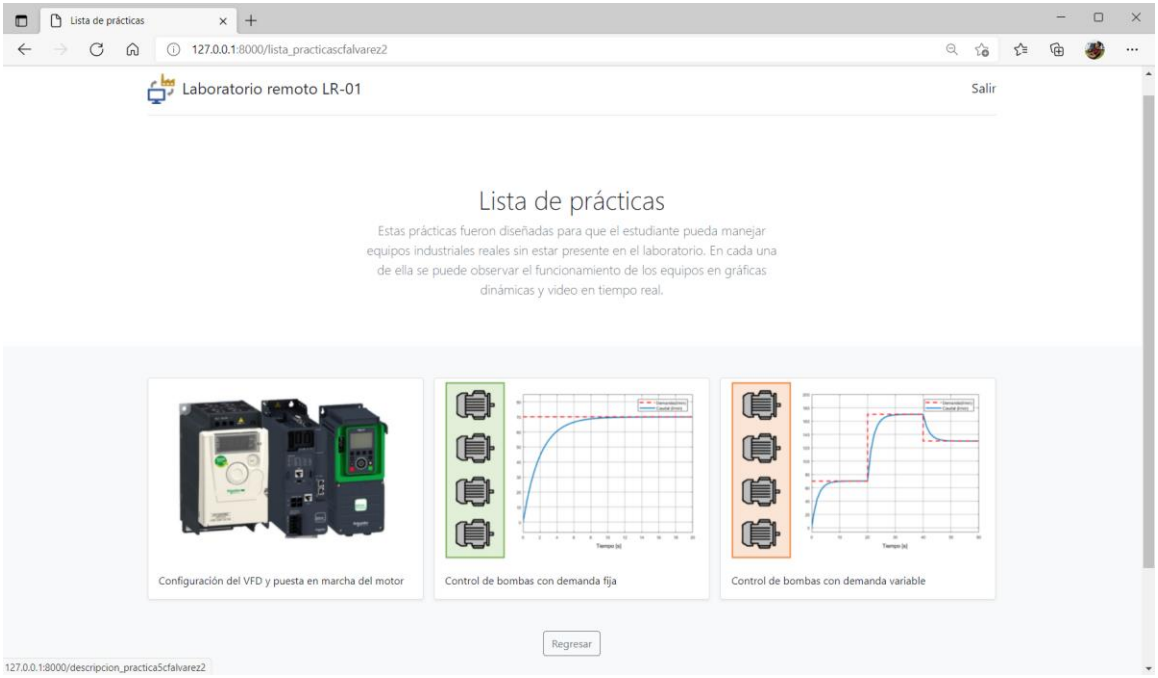
Template que se despliega para ingresar al laboratorio remoto



- **lista_practicas:** es una ventana que permite al usuario identificar las tres prácticas disponibles en el laboratorio remoto. El usuario puede acceder a una descripción específica de cada práctica pulsando las imágenes tal y como muestra la Figura 94.

Figura 94

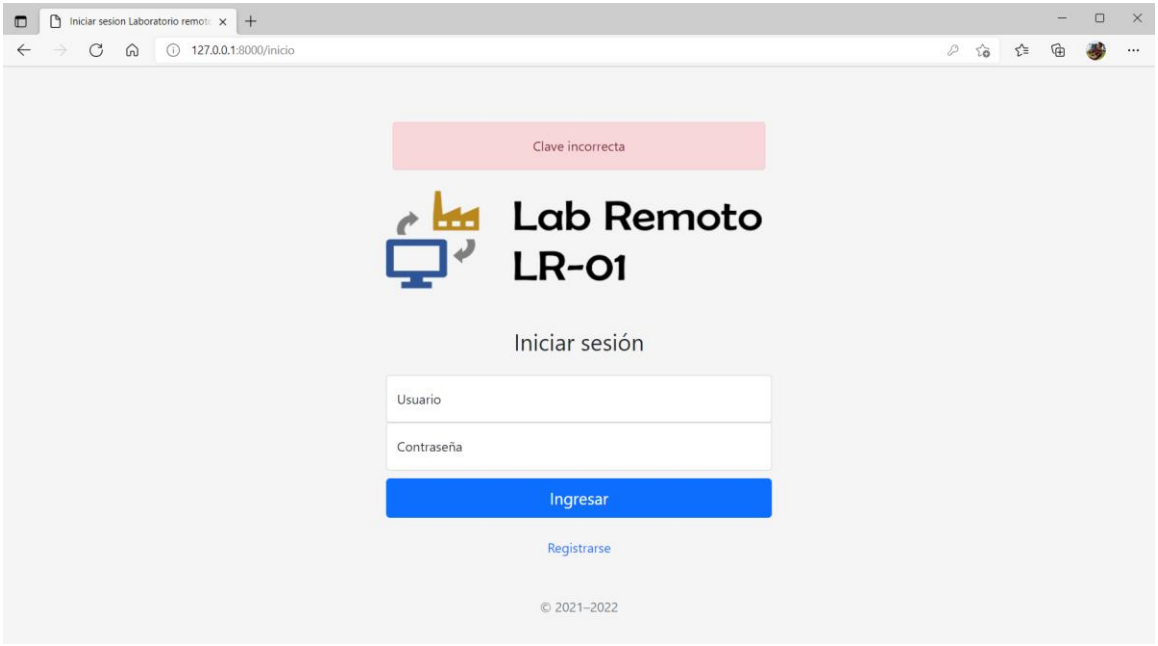
Template que muestra la lista de prácticas del laboratorio remoto



- **messages.html:** es un fichero programado en lenguaje HTML que se muestra en diversas ventanas, especialmente en la de ingreso y registro, para mostrar mensajes. Por ejemplo, cuando la contraseña de un usuario es mal ingresada se despliega un mensaje de error como muestra la Figura 95.

Figura 95

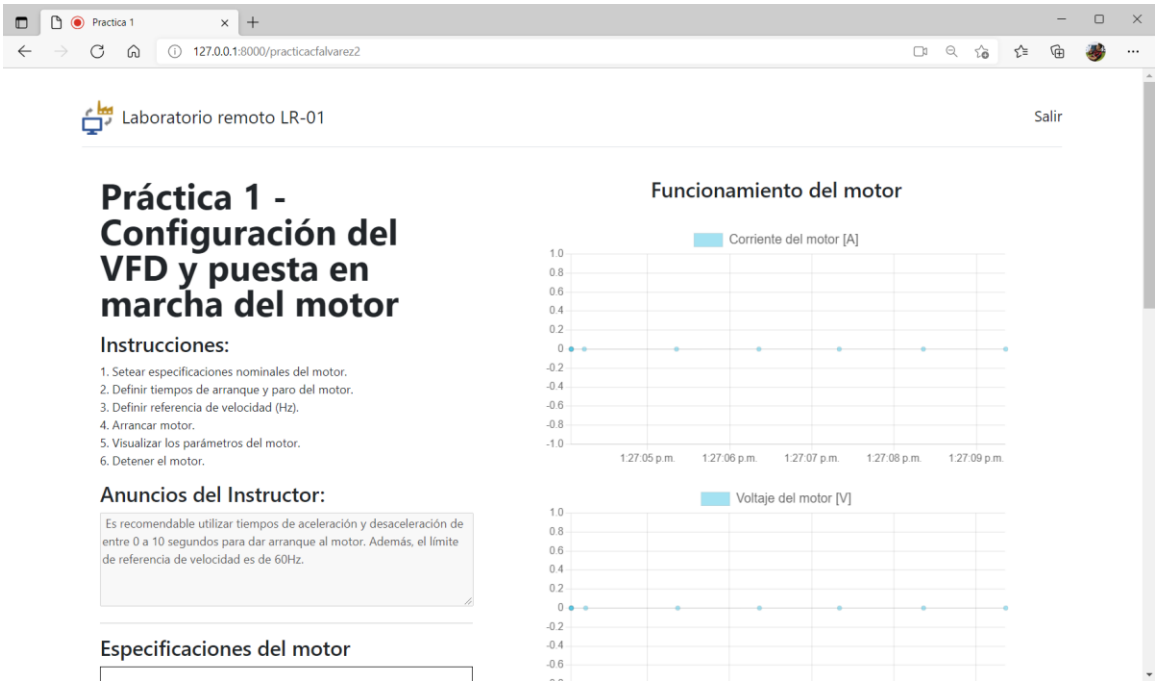
Ejecución del template “messages.html”



- **practica_1.html:** es la ventana en la que se despliega la primera práctica de laboratorio referente al manejo y configuración del variador de frecuencia. En ella, el estudiante desarrolla la práctica de laboratorio, configura parámetros, arranca el motor, observa mediante una cámara el funcionamiento de la maqueta, visualiza gráficas en tiempo real y además puede descargar información útil para realizar informes o análisis. La Figura 96 muestra dicha interfaz.

Figura 96

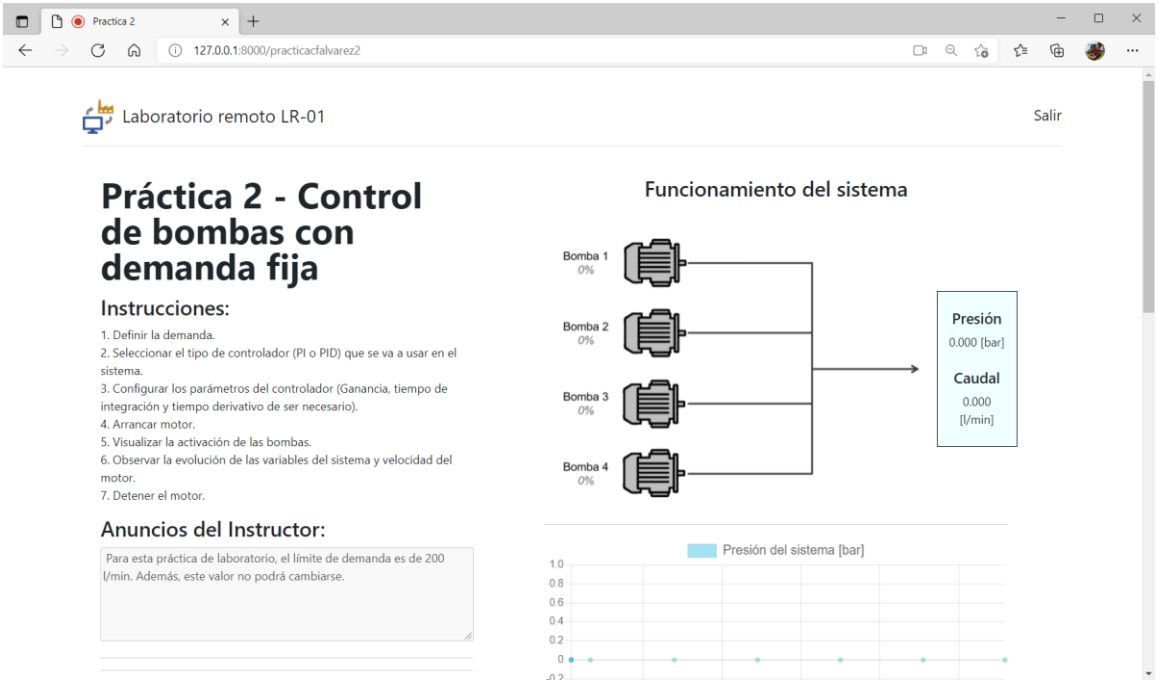
Template de la práctica de laboratorio: configuración del variador y puesta en marcha del motor



- **practica_2.html:** es la ventana que muestra la interfaz de la segunda práctica de laboratorio referente al manejo, control y sintonización del sistema de bombas eléctricas implementado en una maqueta. El estudiante puede configurar parámetros, visualizar el funcionamiento de la maqueta, ver y descargar gráficas de las variables del sistema. Recordar que la demanda requerida por el estudiante para esta práctica es fija en todo momento. La Figura 97 muestra este *template*.

Figura 97

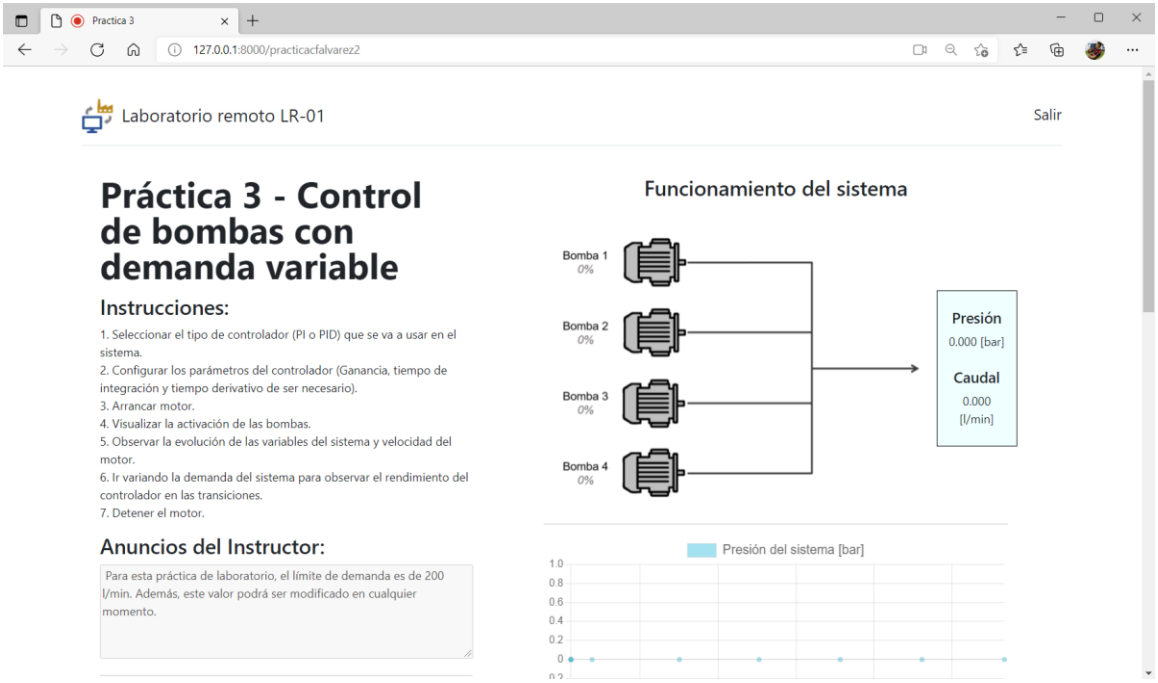
Template de la práctica de laboratorio: manejo de bombas con demanda fija



- **practica_3.html:** esta ventana es similar a la de practica_2.html, con la única diferencia que la demanda requerida por el estudiante puede ser modificada en cualquier momento como muestra la Figura 98.

Figura 98

Template de la práctica de laboratorio: manejo de bombas con demanda variable



- **programa.html:** es la interfaz que se despliega después de acceder mediante usuario y contraseña. En ella se puede ver el usuario que ingresó al laboratorio, información y horario de la práctica reservada (si es el caso), agendar o cambiar un horario para el experimento y la descripción de todas las prácticas tal y como muestra la Figura 99. Si la cuenta con la que se accede pertenece a la del instructor docente, se despliega la ventana mostrada en la Figura 100.

Figura 99

Template “programa.html”

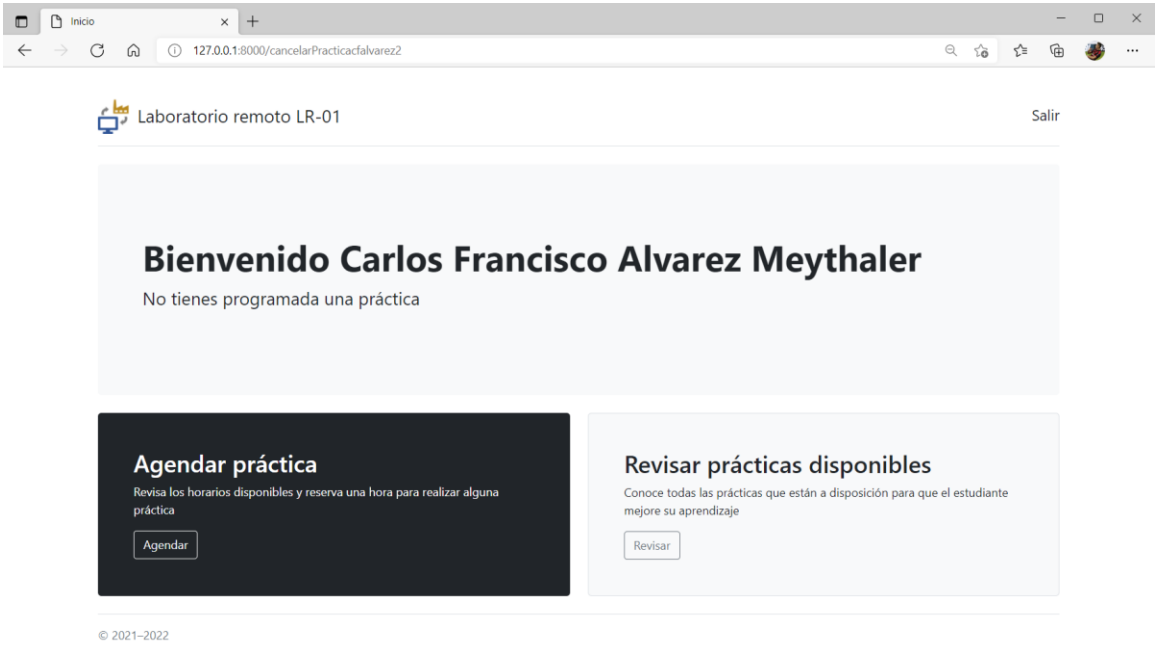
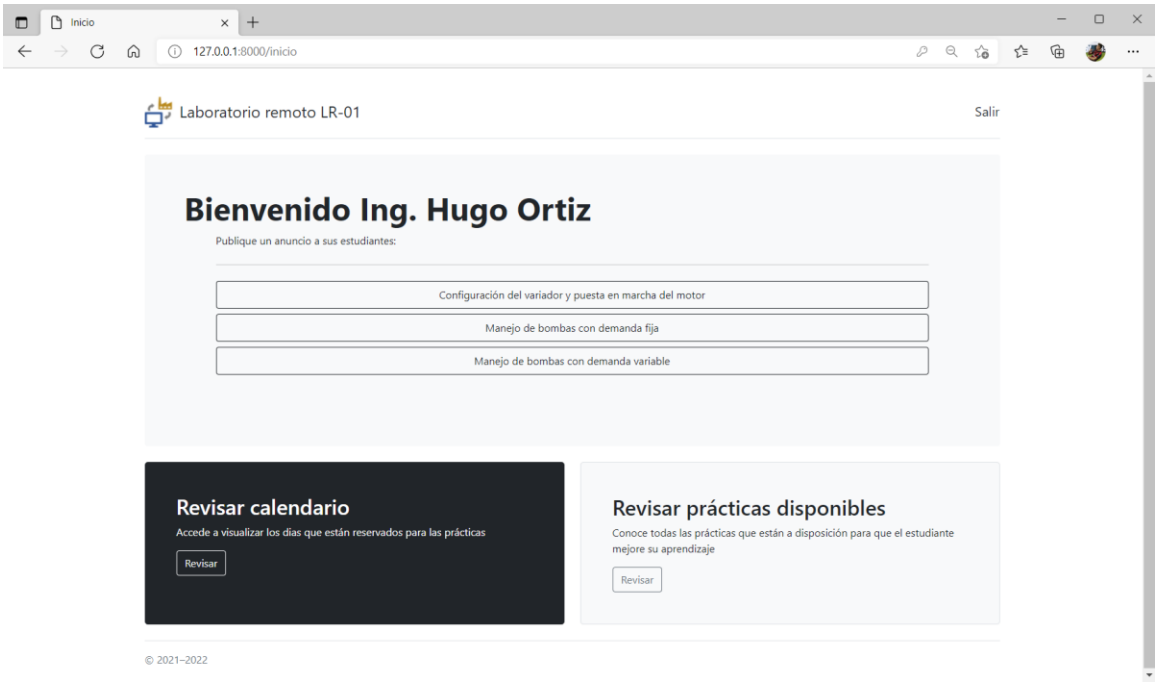


Figura 100

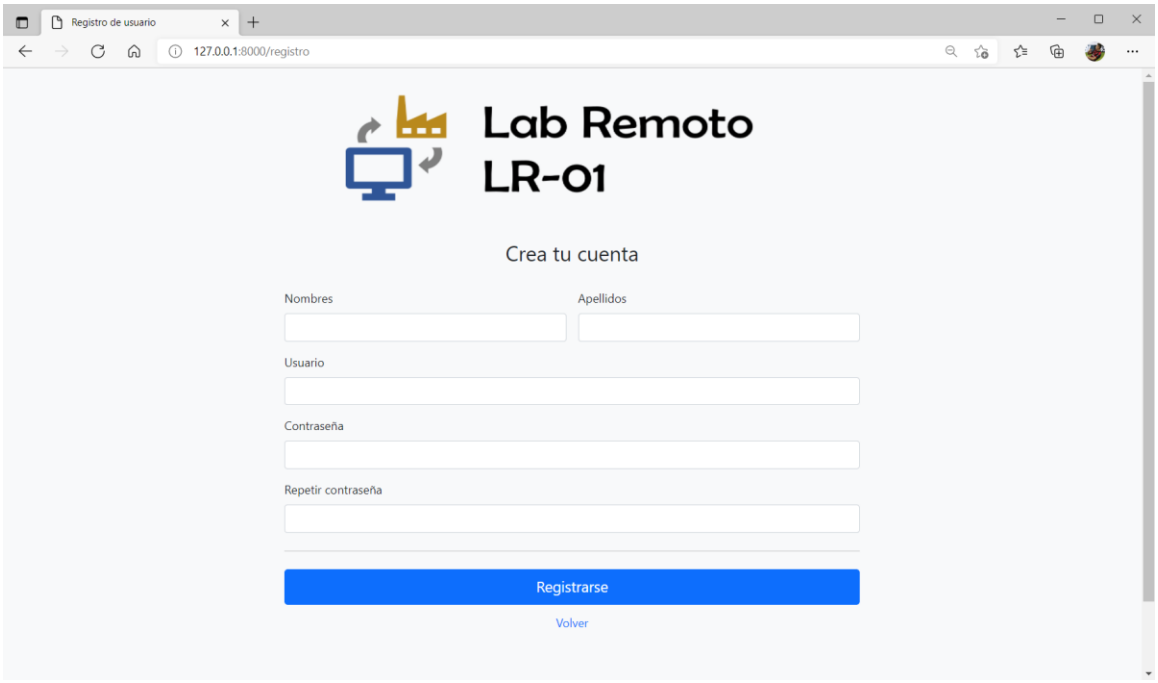
Template “programa.html” para una cuenta de docente



- **registro.html:** es la ventana en donde un usuario puede crear un nuevo registro. Se debe ingresar nombres, apellidos, usuario y contraseña como muestra la Figura 101.

Figura 101

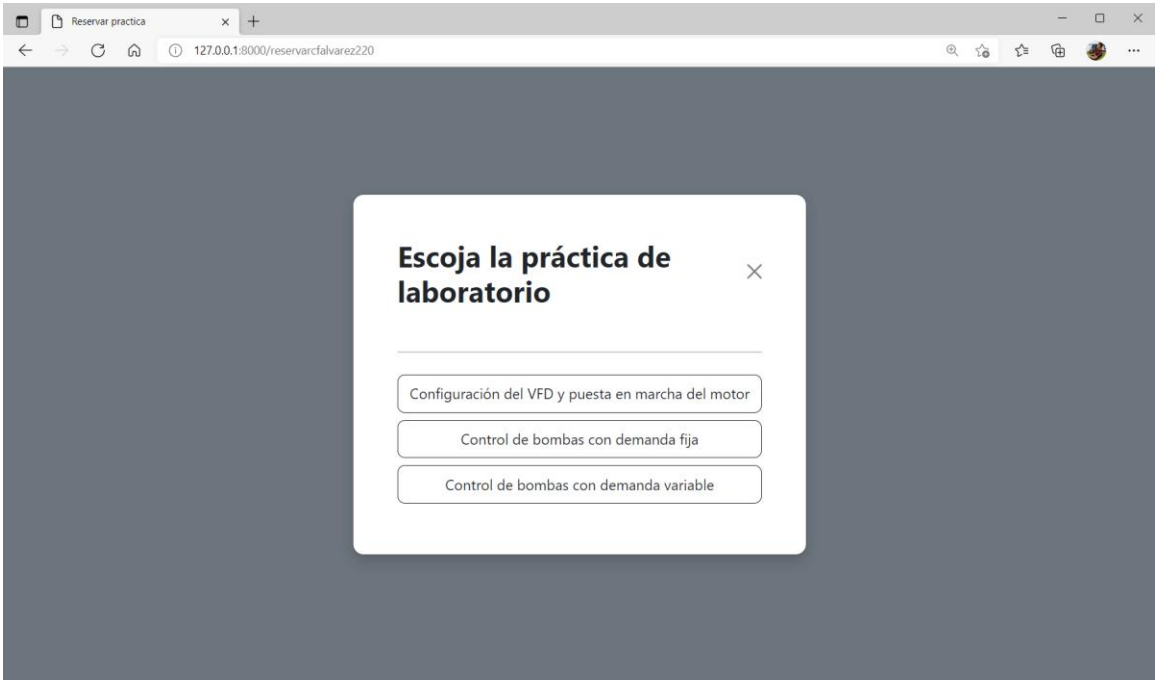
Template de registro de un nuevo usuario



- **reservar.html:** es la interfaz que se despliega una vez que el estudiante reserve un horario. Por lo cual, permite seleccionar una de las tres prácticas de laboratorio a realizar de forma remota tal y como muestra la Figura 102.

Figura 102

Template para escoger una práctica de laboratorio en una reserva



- **webcam.html:** Este archivo programado en lenguaje HTML no va a ser visualizado en la aplicación, pero es útil para establecer conexión con la cámara web situada en la maqueta para proporcionar el video en tiempo real del funcionamiento de los sistemas.

Urls y Views

En Django, las acciones que se llevan a cabo del lado del servidor, se desarrollan mediante Python y, en concreto, creando funciones dentro del archivo “views.py” en caso de peticiones HTML. Estás acciones pueden ser la validación de campos de entrada, lectura y registro en la base de datos, entre otras; sin embargo, la función principal es el despliegue de las páginas HTML. El nexo para llamar una de las funciones desde la página web, es a través de la dirección URL, las cuales se declaran en el archivo “urls.py”.

En la Figura 103, se muestran todas las direcciones URL declaradas para la aplicación web. Mediante la función “path” se define las direcciones URL dentro de “urlpatterns”, indicando el patrón de URL, la función que debe ejecutarse declarada en “views.py”, y el nombre de la dirección. A continuación, se muestran las funciones llamadas para cada dirección URL.

Figura 103

Direcciones URL de la aplicación web

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('inicio', views.index, name='index'),
    path('registro', views.registro, name='registro'),
    path('programa<str:user_usuario>', views.programa, name='programa'),
    path('agendar<str:user_usuario>', views.agendar, name='agendar'),
    path('reservar<str:user_usuario><int:fila><int:columna>', views.reservar, name='reservar'),
    path('guardarPractica<str:user_usuario><int:fila><int:columna><int:num_practica>', \
        views.guardarPractica, name='guardarPractica'),
    path('lista_practicas<str:user_usuario>', views.lista_practicas, name='lista_practicas'),
    path('descripcion_practica<int:practica_id><str:user_usuario>', views.detalle, \
        name="descripcion_practica"),
    path('comunicacion', views.comunicacion, name="comunicacion"),
    path('practica<str:user_usuario>', views.iniciar_practica, name="practica"),
    path('cancelarPractica<str:user_usuario>', views.cancelar_practica, name="cancelarPractica"),
    path('webcam', views.webcam, name="webcam"),
    path('configurarVariador', views.configurarVariador, name="configurarVariador"),
    path('asignarArrancar', views.asignarArrancar, name="asignarArrancar"),
    path('asignarDetener', views.asignarDetener, name="asignarDetener"),
    path('parametrosPractica2', views.parametrosPractica2, name="parametrosPractica2"),
    path('asignarMarcha', views.asignarMarcha, name="asignarMarcha"),
    path('asignarParo', views.asignarParo, name="asignarParo"),
    path('asignarDemanda', views.asignarDemanda, name="asignarDemanda"),
    path('parametrosPractica3', views.parametrosPractica3, name="parametrosPractica3"),
    path('anuncio<str:user_usuario><int:numP>', views.anuncio, name="anuncio"),
    path('guardarAnun<str:user_usuario><int:numP>', views.guardarAnun, name="guardarAnun"),
]
```

Figura 104

Librerías importadas en "views.py"

```
from django.shortcuts import render, get_object_or_404

from django.http import HttpResponse
from django.contrib import messages
from django.utils import timezone
from datetime import datetime, time, timedelta
import numpy as np
from django.utils.timezone import get_current_timezone
from django.views.decorators.csrf import csrf_exempt
from practicas.serializers import EscrituraSerializer
from django.http.response import JsonResponse

from .models import Horario, Usuario, Practica, Lectura, Escritura
```

Figura 105

Función "index"

```
def index(request):
    if request.method == "POST":
        input_usuario = request.POST["signin_usuario"]
        input_clave = request.POST["signin_clave"]
        usuarios_totales = Usuario.objects
        user = usuarios_totales.filter(usuario = input_usuario)
        if user.exists():
            usuario = user[0]
            clave = usuario.clave
            if input_clave == clave:
                tz = get_current_timezone()
                horario0=usuario.horario
                horario = horario0 + tz.utcoffset(horario0)
                if usuario.practica and horario + timedelta(hours=2) < timezone.now():
                    fechaAnterior=horario.date()
                    horaAnterior=horario.time()
                    horarios_totales = Horario.objects.all()
                    fechaExistente=horarios_totales.get(dia=fechaAnterior, hora=horaAnterior)
                    fechaExistente.delete()
                    usuario.practica = False
                    usuario.save()
                return render(request, 'practicas/programa.html', {'user': usuario})
            else:
                messages.error(request, 'Clave incorrecta')
                return render(request, 'practicas/index.html')
        else:
            messages.warning(request, 'Usuario no registrado')
            return render(request, 'practicas/index.html')
    else:
        datos = Escritura.objects.all()[0]
        datos.maqueta = 0
        datos.save()
        return render(request, 'practicas/index.html')
```

En la Figura 104 se indican todas las librerías, funciones y modelos importados dentro de “views.py”. La función “index” que se muestra en la Figura 105, renderiza el archivo “index.html”, sin embargo, cuando es llamado en el caso de que el estudiante trata de ingresar con su cuenta, la función ejecuta el código dentro del método “POST”, el cual valida que sea un usuario registrado y que la clave sea la correcta para redirigir la aplicación a “programa.html”. Antes de pasar a la HTML, valida si el horario que tenía reservado el usuario aún está vigente, de lo contrario, lo elimina de la base de datos de “Horarios” y del propio usuario.

Figura 106

Función “registro”

```
def registro(request):
    if request.method == "POST":
        nombres = request.POST.get("nombres","usuario1")
        apellidos = request.POST.get("apellidos","apellido1")
        username = request.POST.get("username","user1")
        clave = request.POST.get("clave","123")
        clave2 = request.POST.get("clave2","123")
        usuarios_totales = Usuario.objects
        user = usuarios_totales.filter(usuario=username)
        if user.exists():
            messages.warning(request, 'Este usuario ya existe')
            return render(request, 'practicas/registro.html')
        else:
            if clave == clave2:
                nuevo_usuario = Usuario(nombres=nombres, apellidos=apellidos, \
                    usuario=username, clave=clave, horario = timezone.now(), \
                        practica = False, num_practica = 1)
                nuevo_usuario.save()
                messages.success(request, 'Usuario creado correctamente')
                return render(request, 'practicas/index.html')
            else:
                messages.error(request, 'Las claves ingresadas no coinciden')
                return render(request, 'practicas/registro.html')
    else:
        return render(request, 'practicas/registro.html')
```

En la Figura 106 se encuentra el código de la función “registro”, el cual funciona de forma similar a la función “index”. En este caso, por defecto despliega el HTML “registro”; pero, cuando se ejecuta por el registro de un nuevo usuario, valida que el

identificador de usuario no exista ya dentro de la base de datos, y que la clave ingresada coincida con el segundo campo de confirmación; en ese caso, regresa a “index.html”, de lo contrario, renderiza de nuevo “registro.html” agregando un mensaje de error.

Figura 107

Función “programa”

```
def programa(request, user_usuario):
    datos = Escritura.objects.all()[0]
    datos.maqueta = 0
    datos.save()
    usuarios_totales = Usuario.objects.all()
    usuario = usuarios_totales.filter(usuario=user_usuario)[0]
    tz = get_current_timezone()
    horario0=usuario.horario
    horario = horario0 + tz.utcoffset(horario0)
    if usuario.practica and horario + timedelta(hours=2) < timezone.now():
        fechaAnterior=horario.date()
        horaAnterior=horario.time()
        horarios_totales = Horario.objects.all()
        fechaExistente=horarios_totales.get(dia=fechaAnterior, hora=horaAnterior)
        fechaExistente.delete()
        usuario.practica = False
        usuario.save()
    return render(request, 'practicas/programa.html', {'user': usuario})
```

La función “programa” de la Figura 107, renderiza el HTML del mismo nombre, pasando como parámetros el modelo del usuario que inició sesión, para mostrar información específica de la persona y modificar la interfaz según si tiene reservada una práctica o no. Al igual que la función “index”, antes de pasar a la HTML comprueba si el usuario tenía reservado un horario, y si este horario es de un tiempo anterior al de ingreso del usuario, de ser así, lo elimina de la base de datos de “Horarios” y del propio usuario.

El calendario que permite escoger el horario para reservar una práctica en la HTML “agendar”, mostrado en la Figura 89, se genera según la fecha en la cual ingresemos a la aplicación web, es decir, determina la fecha actual y genera el calendario con los siguientes 5 días laborables. Además, en las horas disponibles se

crea un enlace para poder reservar ese horario y desplegar “reservar.html” donde se escoge una de las tres prácticas. Estas dos acciones se desarrollan en la función “agendar” previo a desplegar “agendar.html”. Como se muestra en la Figura 108, utiliza el modelo “Horario” para comprobar los días y horas donde el laboratorio remoto ya fue reservado, con el fin de no crear un hipervínculo en la casilla correspondiente.

Figura 108

Función “agendar”

```
def agendar(request, user_usuario):
    dia0 = datetime.today()
    calendario_dias = ['','','','','']
    ndias = np.zeros(5, dtype=int)
    n = 0
    m = 0
    while n<7:
        dia = dia0 + timedelta(days=n)
        nsemana = dia.weekday()
        if nsemana < 5:
            ndias[m] = nsemana
            calendario_dias[m] = dia.strftime('%d-%m-%Y')
            m += 1
        n += 1
    nombres = ['','','','','']
    j = 0
    for i in ndias:
        nombres[j] = nombre_dias[i]
        j += 1
    horarios = Horario.objects.all()
    calendario = [[False, False, False, False, False],[False, False, False, False, False],\
        [False, False, False, False, False],[False, False, False, False, False]]
    n = 0
    for i in range(7):
        dia = dia0 + timedelta(days=i)
        dia1 = dia.date()
        nsemana = dia.weekday()
        if nsemana<5:
            for j in range(4):
                existe = horarios.filter(dia=dia1, franja=j)
                if j == 0:
                    hora=time(9,15,0)
                elif j == 1:
                    hora=time(11,30,0)
                elif j == 2:
                    hora=time(14,0,0)
                elif j == 3:
                    hora=time(16,15,0)
                horario = datetime.combine(dia1,hora,None)
                if existe.exists() or horario < datetime.now():
                    calendario[j][n] = True
            n += 1
    usuarios_totales = Usuario.objects.all()
    user = usuarios_totales.filter(usuario=user_usuario)
    return render(request, 'practicas/agendar.html', {'nombres': nombres,\
        "fecha": calendario_dias, 'calendario': calendario, 'user': user[0]})
```

Figura 109

Función “guardarPractica”

```
def guardarPractica(request, user_usuario, fila, columna, num_practica):
    print(fila)
    usuarios_totales = Usuario.objects
    horarios_totales = Horario.objects
    dia0 = datetime.today()
    n = 0
    m = 0
    while n<7:
        dia = dia0 + timedelta(days=n)
        nsemana = dia.weekday()
        if nsemana < 5:
            if m == columna:
                fecha=dia.date()
                break
            m += 1
        n += 1

    if fila == 0:
        hora=time(7,15,0)
    elif fila == 1:
        hora=time(9,30,0)
    elif fila == 2:
        hora=time(12,0,0)
    elif fila == 3:
        hora=time(14,0,0)

    user = usuarios_totales.get(usuario=user_usuario)
    tz = get_current_timezone()
    if user.practica:
        horarioAnterior0=user.horario
        horarioAnterior = horarioAnterior0 + tz.utcoffset(horarioAnterior0)
        fechaAnterior=horarioAnterior.date()
        horaAnterior=horarioAnterior.time()
        fechaExistente=horarios_totales.get(dia=fechaAnterior, hora=horaAnterior)
        fechaExistente.delete()

    nuevo_horario=Horario(dia=fecha, hora=hora, franja=fila, num_practica=num_practica)
    nuevo_horario.save()
    user = usuarios_totales.get(usuario=user_usuario)
    user.practica=True
    user.num_practica=num_practica
    user.horario=datetime.combine(fecha,hora,None)
    user.save()
    return render(request, 'practicas/programa.html', {'user': user})
```

Una vez que el estudiante selecciona el horario y la práctica que desea reservar, se ejecuta la función “guardarPractica” de la Figura 109, donde se añade un nuevo

objeto o fila en el modelo “Horario” y se registra esta información en los campos “horario” y “num_practica” del usuario. Si el estudiante ya tenía una práctica reservada y decidió cambiar, la práctica anterior se elimina del modelo “Horario”.

Figura 110

Función “lista_practicas”

```
def lista_practicas(request, user_usuario):
    usuarios_totales = Usuario.objects
    user = usuarios_totales.filter(usuario=user_usuario)
    practicas = Practica.objects.all()
    return render(request, 'practicas/lista_practicas.html', \
        {'practicas': practicas, 'user': user[0]})
```

Figura 111

Función “detalle”

```
def detalle(request, practica_id, user_usuario):
    usuarios_totales = Usuario.objects
    user = usuarios_totales.filter(usuario=user_usuario)
    practica_detail = get_object_or_404(Practica, pk=practica_id)
    return render(request, 'practicas/detalle.html', \
        {'practica': practica_detail, 'user': user[0]})
```

Las funciones “lista_practicas” y “detalle” que se muestran en las Figuras Figura 110 y Figura 111 respectivamente, renderizan los archivos “lista_practicas.html” y “detalle.html” los cuales muestran la información de las prácticas desarrolladas. En el primero envía como parámetro todas las prácticas guardadas en el modelo “Practica”, mientras que, en el segundo, solo envía la práctica seleccionada en el HTML “lista_practicas” para desplegar mayor detalle de la misma.

Figura 112

Función “iniciar_practica”

```
def iniciar_practica(request, user_usuario):
    usuarios_totales = Usuario.objects
    user = usuarios_totales.filter(usuario=user_usuario)[0]
    lecturas = Lectura.objects.all()[0]
    tz = get_current_timezone()
    horario0=user.horario
    horario = horario0 + tz.utcoffset(horario0)
    if user.practica and horario < timezone.now():
        if lecturas.ready:
            anuncios = Anuncio.objects.all()[0]
            num_practica = user.num_practica
            datos = Escritura.objects.all()[0]
            datos.configurar = False
            datos.arrancar = False
            datos.detener = False
            datos.marchaPP = False
            datos.paroPP = False
            if num_practica == 1:
                datos.maqueta = 1
                datos.save()
                return render(request, 'practicas/practica_1.html', {'user': user,\
                    'anuncios' : anuncios})
            else:
                datos.maqueta = 2
                datos.save()
                if num_practica == 2:
                    return render(request, 'practicas/practica_2.html', {'user': user,\
                        'anuncios' : anuncios})
                else:
                    return render(request, 'practicas/practica_3.html', {'user': user,\
                        'anuncios' : anuncios})
        else:
            messages.warning(request, 'Espere a que la maqueta este lista')
            return render(request, 'practicas/programa.html', {'user': user})
    else:
        messages.warning(request, 'Aún no puede realizar la práctica porque no \
            está dentro del horario')
        return render(request, 'practicas/programa.html', {'user': user})
```

La función “iniciar_practica” que se encuentra en la Figura 112, despliega el HTML de la práctica que reservó el estudiante. En caso de que trate de iniciar la práctica fuera de horario o que la maqueta aún no esté lista para funcionar, permanece en la misma ventana y envía un mensaje de error.

Figura 113

Función “cancelar_practica”

```
def cancelar_practica(request, user_usuario):
    usuarios_totales = Usuario.objects.all()
    usuario = usuarios_totales.filter(usuario=user_usuario)[0]
    horarios_totales = Horario.objects.all()
    tz = get_current_timezone()
    horarioAnterior0=usuario.horario
    horarioAnterior = horarioAnterior0 + tz.utcoffset(horarioAnterior0)
    fechaAnterior=horarioAnterior.date()
    horaAnterior=horarioAnterior.time()
    fechaExistente=horarios_totales.get(dia=fechaAnterior, hora=horaAnterior)
    fechaExistente.delete()
    usuario.practica = False
    usuario.save()
    return render(request, 'practicas/programa.html', {'user': usuario})
```

En la Figura 113 se encuentra el código de la función “cancelar_practica”. Esta función es llamada cuando se pulsa el botón para cancelar la práctica agendada. Únicamente elimina el horario en el modelo del usuario y su correspondiente en la tabla de “Horario”.

Figura 114

Función “anuncio”

```
def anuncio(request, user_usuario, numP):
    usuarios_totales = Usuario.objects
    user = usuarios_totales.filter(usuario=user_usuario)[0]
    return render(request, 'practicas/anuncio.html', \
        {'user': user, 'numP' : numP})
```

Por último, se tiene dos funciones especiales para generar anuncios en las prácticas por parte del docente. La función “anuncio” que se muestra en la Figura 114, despliega el template “anuncio.html”. En esta se coloca el mensaje que desea postear al momento que el estudiante realice la práctica previamente seleccionada, y cuando se da clic en “Publicar”, se ejecuta la función “guardarAnum” de la Figura 115, donde se guarda el

mensaje en el campo correspondiente del modelo “Anuncio” y renderiza el HTML “programa”.

Figura 115

Función “guardarAnum”

```
def guardarAnun(request,user_usuario, numP):
    if request.method == "POST":
        anuncio = request.POST.get("anuncio")
        usuarios_totales = Usuario.objects
        user = usuarios_totales.filter(usuario=user_usuario)[0]
        anuncios = Anuncio.objects.all()[0]
        if numP == 1:
            anuncios.anuncio_Practica1 = anuncio
            anuncios.save()
        if numP == 2:
            anuncios.anuncio_Practica2 = anuncio
            anuncios.save()
        if numP == 3:
            anuncios.anuncio_Practica3 = anuncio
            anuncios.save()
    return render(request, 'practicas/programa.html', \
        {'user': user})
```

Implementación del enlace para el acceso remoto

Con la aplicación web finalizada y totalmente funcional de forma local, se implementan los recursos que permiten que el usuario acceda de manera remota a la aplicación y en consecuencia al laboratorio remoto. En esta sección, se describe cómo se aloja la aplicación desarrollada en la plataforma de servicios en la nube “Heroku” y también el procedimiento que se utiliza para realizar la lectura y escritura de datos en el servidor de la nube desde la tarjeta Raspberry Pi 4.

Alojamiento de la aplicación web en Heroku

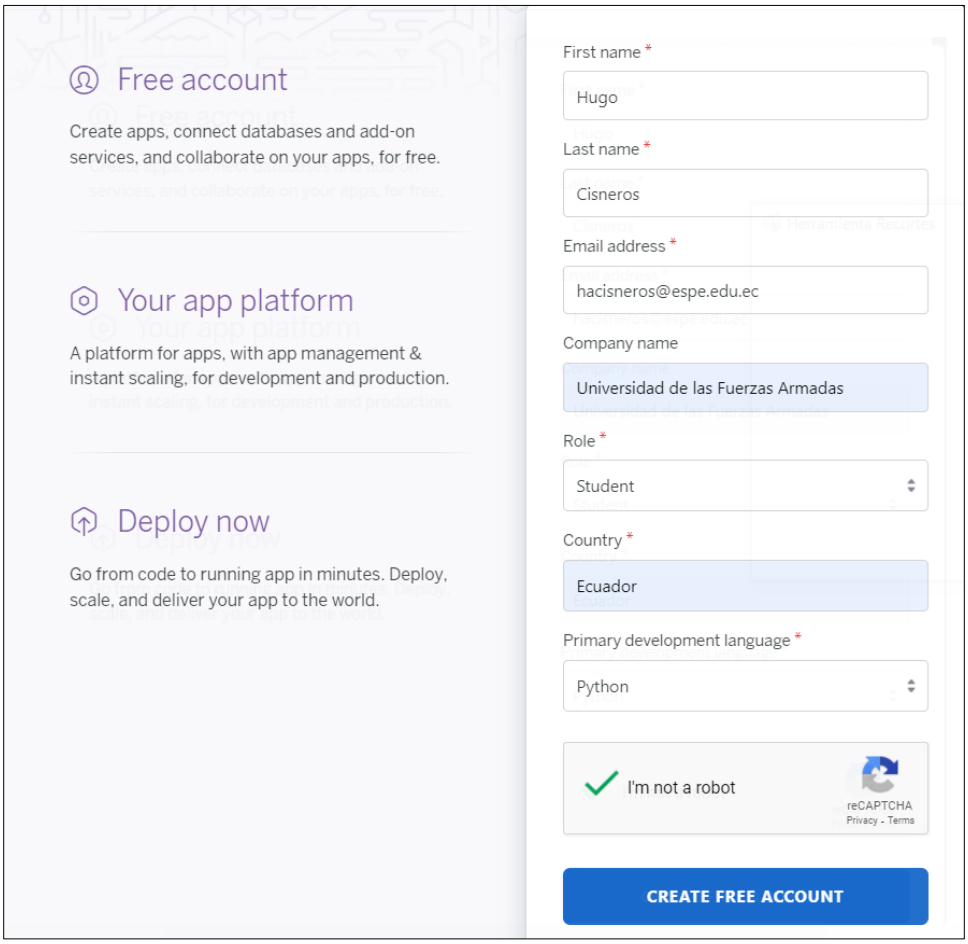
Heroku es una plataforma de servicios cuya principal función es almacenar y correr una aplicación en la nube. Una vez que la aplicación es probada corriendo el servidor local del computador, es necesario realizar una secuencia de pasos para poder subir dicha aplicación a Heroku.

1. Crear una cuenta en Heroku

Se ingresa a la página web oficial de Heroku https://signup.heroku.com/login y se crea una cuenta gratuita con el correo institucional de la Universidad de las Fuerzas Armadas ESPE, como muestra la Figura 116.

Figura 116

Creación de una cuenta en Heroku



2. Instalar el CLI (Interfaz de línea de comandos) de Heroku

El CLI de Heroku permite crear y administrar la aplicación que se va a subir a la nube desde un terminal. Para utilizar esta interfaz, es necesario instalar en el ordenador el sistema de control de versiones GIT cuya función es administrar y mantener las versiones de la aplicación debido a su gran cantidad de archivos de código fuente y sus

diversos lenguajes de programación: Python, HTML, Java, CSS. Adicional a esto, GIT permite que la copia de trabajo del código sea también un repositorio que almacene todos los cambios de la aplicación (Atlassian Bitbucket, 2022).

Para instalar GIT, basta con dirigirse al sitio https://git-scm.com/download/win y dar click en la opción de descarga de la última versión para Windows. Luego, hay que ejecutar el instalador que se genera al descargarse.

A continuación, se descarga el instalador del CLI de Heroku para Windows - 64 bits desde el enlace https://devcenter.heroku.com/articles/heroku-cli y se procede a ejecutarlo.

Cuando estas dos herramientas de software estén instaladas, el proceso se realiza desde un terminal que tenga activado el entorno virtual.

3. Iniciar sesión en Heroku

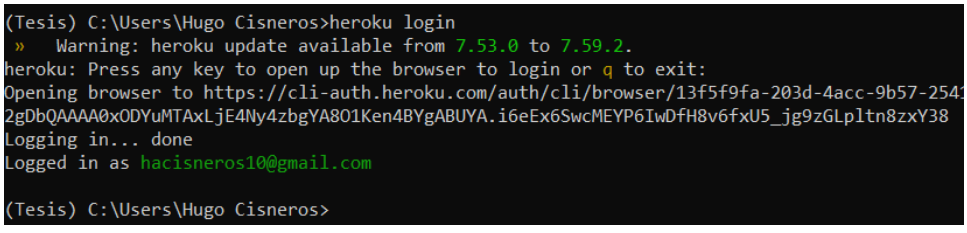
En el terminal, se ejecuta el siguiente comando:

heroku login

De esta manera, automáticamente se abre el navegador de Heroku solicitando que se inicie sesión con la información de la cuenta creada anteriormente. Si la sesión fue iniciada satisfactoriamente, en el terminal se ve como resultado tal y como muestra la Figura 117.

Figura 117

Inicio de sesión satisfactorio en Heroku



4. Crear nueva aplicación en Heroku

Se crea una aplicación en la nube con el nombre “labremotoespe” ejecutando la siguiente línea de código en el terminal:

heroku create labremotoespe

5. Crear un nuevo repositorio Git

A través del comando *cd* hay que ubicarse en la carpeta donde se encuentre la aplicación desarrollada, específicamente el archivo “manage.py”. Es ahí en donde se va a crear un nuevo repositorio GIT que permita administrar y manejar las versiones de la aplicación guardando todo el historial de los cambios. Después, en el terminal se ejecutan las siguientes líneas de código:

git init

heroku git:remote -a labremotoespe

Es oportuno mencionar que “labremotoespe” es el nombre que se asigna a la aplicación previamente.

6. Instalación de paquetes adicionales

Para subir el proyecto a la nube es necesario de la instalación bibliotecas adicionales: daphne, psycopg2, dj-database-url, decouple y whitenoise. En el terminal se ejecuta:

pip install daphne

pip install psycopg2

pip install dj-database-url

pip install database decouple

pip install database whitenoise

La librería “daphne” es un servidor de terminación HTTP y Websocket de Python que permite todo el despliegue de la aplicación. Sin esta librería, es imposible la ejecución de la aplicación en la nube.

La librería “psycopg2” permite la conexión con el gestor de base de datos “Postgres”, que precisamente es la herramienta de almacenamiento y manejo de datos de la aplicación para el laboratorio remoto. Por otro lado, la librería “dj-database-url” permite la conexión entre el proyecto y el gestor de base de datos, debido a que Heroku provee la base de datos en un servidor distinto al de la aplicación. Es así, que la librería mencionada recientemente provee una URL para la comunicación entre la aplicación y la base de datos.

La librería “decouple” sirve para iniciar el servidor a través de la lectura de variables de entorno establecidas en la configuración de la aplicación.

Finalmente, la librería “whitenoise” permite que la aplicación Heroku pueda servir archivos estáticos en la nube.

7. Incluir los requerimientos

La aplicación que se desarrolla requiere de una gran cantidad de librerías para su funcionamiento. Por lo tanto, la aplicación en la nube necesita conocer de todos estos requerimientos. Esto se consigue generando un archivo de texto en el directorio de la aplicación con el nombre “requirements.txt”. Para esto, se ejecuta el comando:

pip freeze > requirements.txt

8. Modificar el archivo “settings.py”

El primer aspecto que se modifica son los “ALLOWED_HOSTS” que son los dominios permitidos para el uso de la aplicación. Para el caso del presente trabajo, se configura para que el acceso sea público. Además, como en el laboratorio remoto se requiere del uso de sentencias “POST” y “GET” para el envío y recepción de datos en el PLC, se debe configurar los orígenes confiables “CSRF_TRUSTED_ORIGINS” para estas solicitudes. La configuración resulta como muestra la Figura 118. La base de datos debe ser configurada como muestra la Figura 119.

Figura 118

Configuración de dominios permitidos y orígenes confiables para la aplicación

```
ALLOWED_HOSTS = ['*']
CSRF_TRUSTED_ORIGINS = ['https://*.labremotoespe.herokuapp.com/']
```

Figura 119

Configuración de la base de datos

```
import dj_database_url
from decouple import config

DATABASES = {
    'default': dj_database_url.config(
        default=config('DATABASE_URL')
    )
}
```

"DATABASE_URL" es la variable de entorno que se genera por defecto una vez que se crea la aplicación en Heroku, y va a proporcionar la dirección URL con la cual la aplicación se va a conectar con la base de datos a través de la librería "dj_database_url".

Para que la aplicación en la nube pueda servir archivos estáticos, es necesario de la inclusión de la librería "whitenoise" tal y como se indica en la Figura 120.

Figura 120

Inclusión del middleware "whitenoise" en el archivo settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
]
```

Además, al final del archivo “settings.py” es necesaria la creación de un almacén de componentes estáticos como muestra la línea de código en la Figura 121.

Figura 121

Configuración para el almacenamiento de archivos estáticos en la nube

```
STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

9. Creación del fichero “Procfile”

“Procfile” es el primer archivo que se ejecuta cuando arranca el servidor en la nube y se debe crear al mismo nivel del archivo “manage.py” de tal forma que el árbol del proyecto resulte como la Figura 122. La configuración de este fichero es el que se muestra en la Figura 123.

Figura 122

Árbol del proyecto añadido el archivo “Procfile”

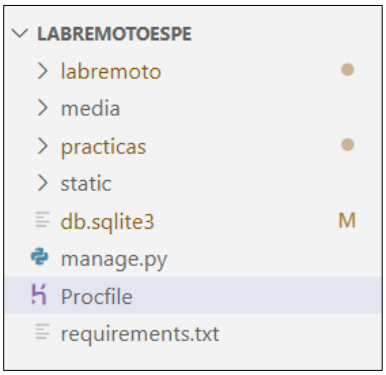
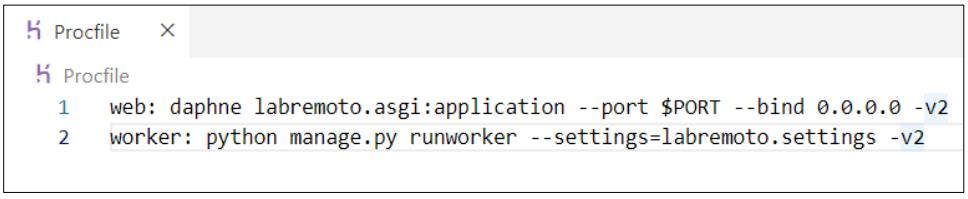


Figura 123

Configuración del archivo Procfile



La primera línea de código que utiliza la librería “daphne” da inicio a la aplicación ejecutando el archivo “labremoto.asgi”. Mientras que la segunda línea ejecuta el archivo “manage.py” para inicializar las configuraciones de la aplicación implantadas en el “settings.py”.

10. Desplegar la aplicación en Heroku

Luego, a través del terminal, con el entorno virtual activo y posicionados en el directorio en donde se encuentra el archivo “manage.py” de la aplicación, se debe confirmar el código programado en el repositorio a través del siguiente comando:

git add –all

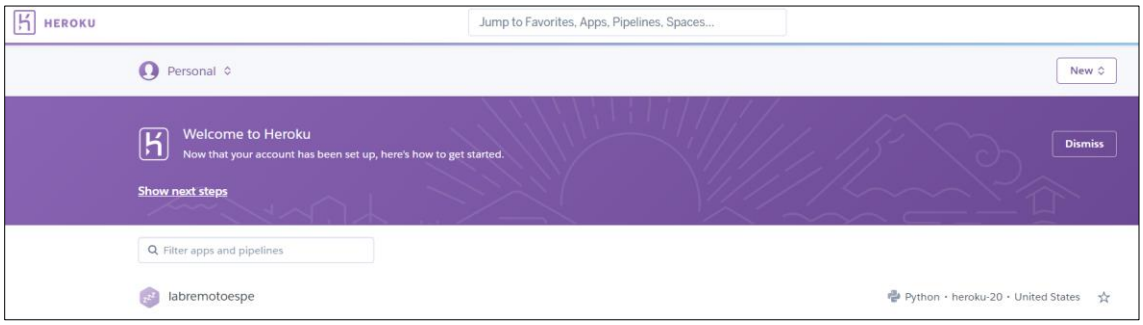
Posterior a esto, se implementa la aplicación en Heroku con el comando:

git push heroku master

De esta forma, la aplicación está alojada en la plataforma de servicios Heroku y consecuentemente en la nube para ser accedida por cualquier usuario. La Figura 124 muestra el resultado del alojamiento de la aplicación.

Figura 124

Aplicación del laboratorio remoto alojada en Heroku

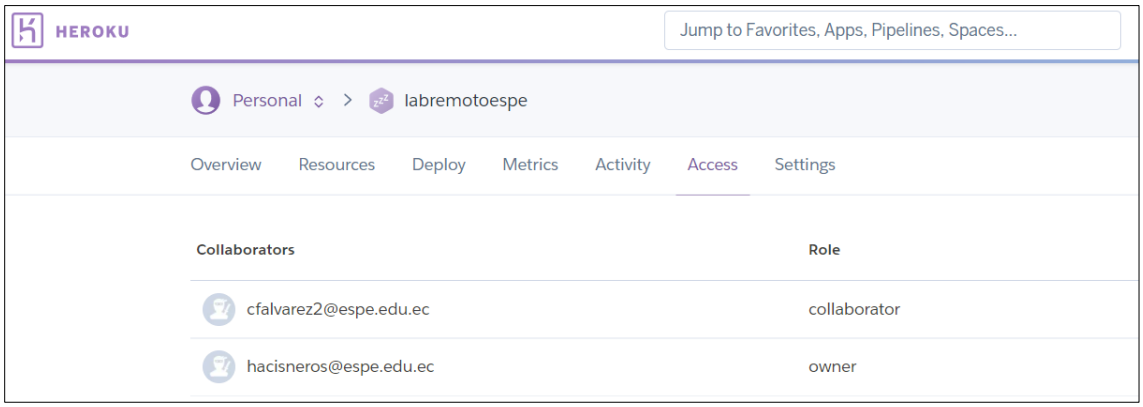


11. Permisos compartidos en la aplicación

La aplicación puede ser actualizada, clonada y modificada por varias personas. En la pestaña de *Access* se puede añadir un colaborador a la aplicación. La Figura 125 muestra la lista de autores de la aplicación del laboratorio remoto.

Figura 125

Lista de autores de la aplicación con permisos

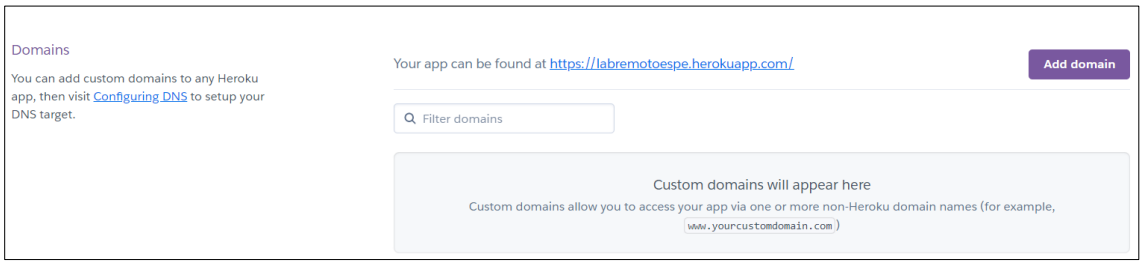


12. Ingreso a la aplicación

Para acceder a la URL de la aplicación, se debe ir a la pestaña de configuraciones del proyecto subido, y posteriormente a la sección de dominios como muestra la Figura 126.

Figura 126

Dominio otorgado por Heroku para la aplicación del laboratorio remoto



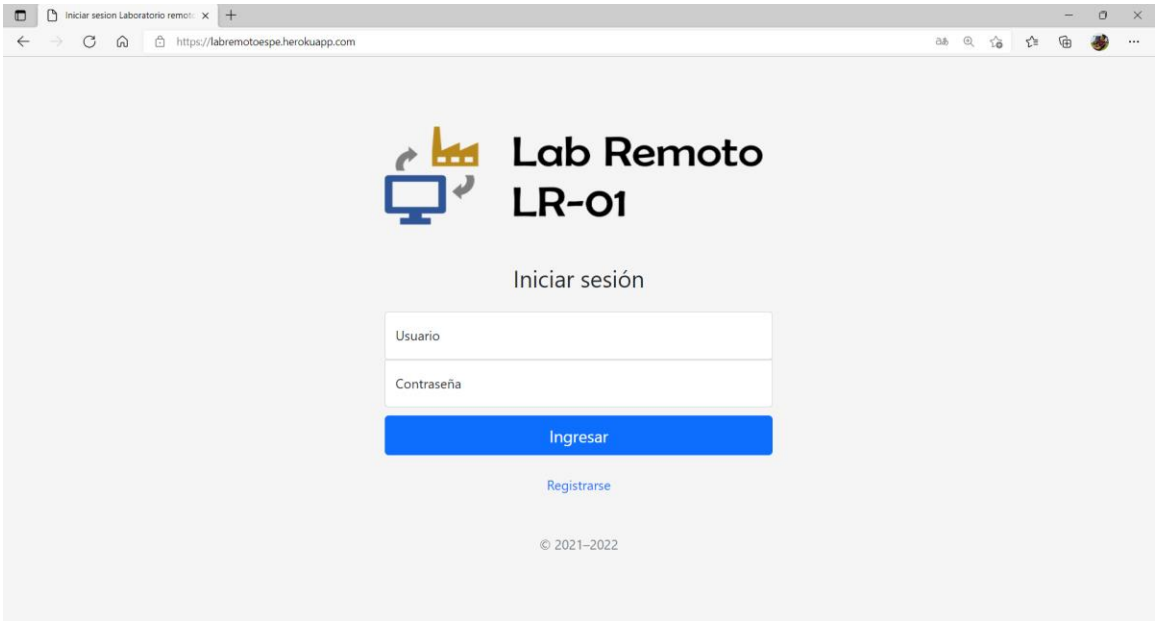
En definitiva, la URL de la aplicación del laboratorio remoto es:

https://labremotoespe.herokuapp.com/

Si se coloca mencionada URL en cualquier navegador, se despliega la ventana de inicio de la aplicación tal y como indica la Figura 127.

Figura 127

Despliegue de la ventana de inicio de la aplicación por medio de la URL https://labremotoespe.herokuapp.com/



Lectura y envío de datos hacia el servidor desde la Raspberry Pi 4

Una vez que se instala la librería “requests” en el entorno virtual de la Raspberry Pi, se tiene a disposición funciones para realizar peticiones a HTTP a cualquier URL. Para comunicarnos con el servidor que aloja la aplicación web se utilizan las funciones “GET” y “POST” para leer y enviar datos respectivamente.

Para emplear los métodos “get” y “post” de la librería requests, únicamente se ingresa la dirección URL a realizar la petición, aunque también es posible pasar parámetros dependiendo de la acción que se va a realizar. Con lo cual, primero se define la URL y las acciones en cada una de las peticiones dentro de la aplicación web.

En la Figura 128 se resalta la URL definida para intercambiar datos con la Raspberry Pi 4. Esta URL referencia a la función “comunicación” la cual se observa en la Figura 129. Cuando le llega una petición “POST” significa que debe recibir información respecto al funcionamiento de la maqueta, con lo cual, extrae los

parámetros añadidos en la petición y los almacena en el modelo “Lectura” sin retornar ninguna respuesta al cliente.

Figura 128

URL utilizada para comunicar el servidor con la Raspberry Pi 4

```
path('descripcion_practica<int:practica_id><str:user_usuario>', views.detalle, \
    name="descripcion_practica"),
path('comunicacion', views.comunicacion, name="comunicacion"),
path('practica<str:user_usuario>', views.iniciar_practica, name="practica"),
```

Figura 129

Función “comunicacion”

```
@csrf_exempt
def comunicacion(request):
    if request.method == "POST":
        anteriores = Lectura.objects.all().get(pk=1)
        anteriores.ready = request.POST.get("ready")
        # Practica1
        anteriores.corriente = request.POST.get("corriente")
        anteriores.voltaje = request.POST.get("voltaje")
        anteriores.potencia = request.POST.get("potencia")
        anteriores.velocidad = request.POST.get("velocidad")
        # Practica 2 y 3
        anteriores.presion = request.POST.get("presion")
        anteriores.demanda = request.POST.get("demanda")
        anteriores.bombaF1 = request.POST.get("bombaF1")
        anteriores.bombaF2 = request.POST.get("bombaF2")
        anteriores.bombaF3 = request.POST.get("bombaF3")
        anteriores.bombaF4 = request.POST.get("bombaF4")
        anteriores.bombaC1 = request.POST.get("bombaC1")
        anteriores.bombaC2 = request.POST.get("bombaC2")
        anteriores.bombaC3 = request.POST.get("bombaC3")
        anteriores.bombaC4 = request.POST.get("bombaC4")
        anteriores.vel1 = request.POST.get("vel1")
        anteriores.vel2 = request.POST.get("vel2")
        anteriores.vel3 = request.POST.get("vel3")
        anteriores.vel4 = request.POST.get("vel4")
        anteriores.save()
```

En cambio, si es una petición “GET”, debe retornar los campos alojados en el modelo “Escritura”, sin embargo, si se pasan los datos como instancia del modelo, la

Raspberry no podrá acceder a la información; con lo cual, se utiliza serializadores que permiten convertir datos complejos, como conjuntos de consultas e instancias de modelos, en tipos de datos nativos de Python. En la Figura 130 se observa la clase EscrituraSerializer, definida en el archivo “serializers.py”, que nos permite serializar datos del modelo Escritura. Se utiliza la clase “ModelSerializer” de la librería “rest_framework” para crear un serializador de modelos necesitando únicamente especificar los campos que se pretende almacenar en un dato tipo “diccionario”. Con los datos serializados, se envía al cliente mediante “JsonResponse”, el cual es una clase de respuesta HTTP que recoge los datos para ser serializados a JSON.

Figura 130

Serializador “EscrituraSerializer”

```
from rest_framework import serializers
from practicas.models import Escritura

class EscrituraSerializer(serializers.ModelSerializer):

    class Meta:
        model = Escritura
        fields = ('maqueta', 'SPfrecuencia', 'Taceleracion', 'Tdesaceleracion', \
            'SPdemanda', 'selectorPID', 'kp', 'ti', 'td', 'configurar', \
                'arrancar', 'detener', 'marchaPP', 'paroPP', 'configurarParam')
```

Habiendo definida la acción en ambos métodos, ahora en la Raspberry Pi 4 se realizan las peticiones como se muestran en las Figuras Figura 131 y Figura 132. Los datos obtenidos desde el PLC son almacenados en un diccionario para pasar como parámetros del método “post”. Por otro lado, cuando se realiza la petición “get”, se almacena en “response” todos los datos que fueron enviados desde la aplicación web. Mediante el método “json()” de la clase “Response” se deserializa los datos obtenidos en formato JSON a un tipo de dato diccionario.

Figura 131

Envío de datos a la aplicación web alojada en Heroku

```
import time
import requests
url = 'https://labremotoespe.herokuapp.com/comunicacion'

# ENVÍO DE DATOS A LA APLICACIÓN WEB
query = {'ready': LReadyPP,
         'corriente': CorrienteMotor,
         'voltaje': VoltajeMotor,
         'potencia': PotenciaMotor,
         'velocidad': VelocidadActualRPM,
         'presion': PresionBomba,
         'demanda': CaudalTotal,
         'bombaF1': {"0": False, "1": True}[bombasON[9]],
         'bombaF2': {"0": False, "1": True}[bombasON[8]],
         'bombaF3': {"0": False, "1": True}[bombasON[7]],
         'bombaF4': {"0": False, "1": True}[bombasON[6]],
         'bombaC1': {"0": False, "1": True}[bombasON[5]],
         'bombaC2': {"0": False, "1": True}[bombasON[4]],
         'bombaC3': {"0": False, "1": True}[bombasON[3]],
         'bombaC4': {"0": False, "1": True}[bombasON[2]],
         'vel1': velocidadB1,
         'vel2': velocidadB2,
         'vel3': velocidadB3,
         'vel4': velocidadB4,
         }
time.sleep(0.1)
res = requests.post(url, data=query)
```

Figura 132

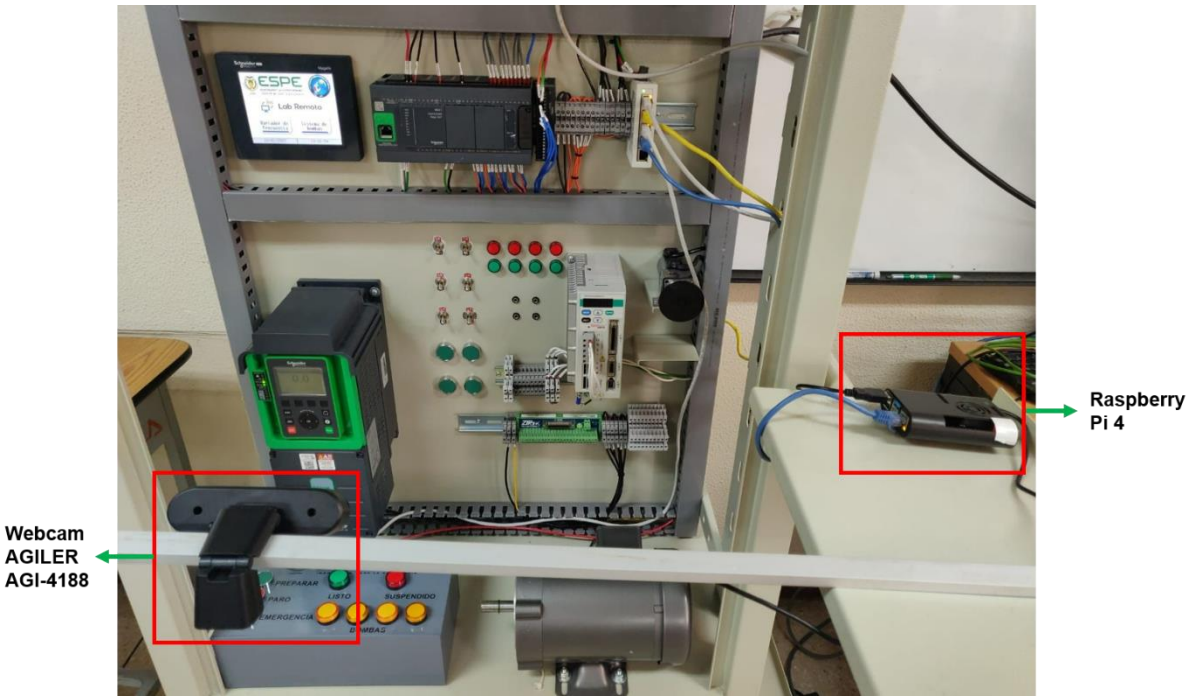
Lectura de datos de la aplicación web alojada en Heroku

```
# OBTENCION DE INFORMACION DESDE LA APLICACION PARA ENVIAR AL PLC
time.sleep(0.1)
response = requests.get(url)
Maqueta = response.json()["maqueta"]
SPVariador = response.json()["SPfrecuencia"]
T_aceleracion = response.json()["Taceleracion"]
T_desaceleracion = response.json()["Tdesaceleracion"]
Demanda = response.json()["SPdemanda"]
selectorPID = response.json()["selectorPID"]
Kp = response.json()["kp"]
Ti = response.json()["ti"]
Td = response.json()["td"]
Configurar = response.json()["configurar"]
Arrancar = response.json()["arrancar"]
Detener = response.json()["detener"]
MarchaPP = response.json()["marchaPP"]
ParoPP = response.json()["paroPP"]
aplicarParam = response.json()["configurarParam"]
```

La Figura 133, muestra la integración de la maqueta con la tarjeta Raspberry Pi 4, cuya función se centra en el enlace remoto. Además, se puede observar la ubicación de la webcam para poder transmitir video hacia la aplicación web.

Figura 133

Integración de la maqueta con la tarjeta Raspberry Pi 4 y la webcam AGILER AGI-4188



Generación de gráficas dinámicas

Django actualmente maneja únicamente conexiones HTTP, lo cual no es suficiente para mostrar gráficas en tiempo real, ya que, implicaría que el usuario refresque la página en múltiples ocasiones para notar cambios. Para corregir esta limitación de Django, existe una función llamada “Channels”, la cual permiten manejar otro tipo de conexiones diferentes a las HTTP, como por ejemplo WebSockets, y que a su vez permiten establecer conexiones asíncronas de larga duración.

WebSockets es un protocolo bidireccional cliente-servidor, que permite intercambiar datos de manera asíncrona mediante una sola petición. Es decir, el usuario no tiene que refrescar el navegador. Este protocolo se puede integrar con Django mediante la función Channels que se mencionó anteriormente.

Adicional a esto, se debe utilizar “Redis-Channel”, que es la única capa de canal oficial de Django. Esta permite utilizar a Redis como base de datos de respaldo y admite

configuraciones de un solo servidor. En términos básicos, Redis-Channel abre uno o varios canales en la aplicación para que puedan establecerse una o varias conexiones WebSockets al mismo tiempo, es decir, para que varios usuarios puedan acceder a los componentes dinámicos de la aplicación de manera paralela.

Para utilizar la funcionalidad Channels y Channels-Redis dentro de la aplicación, en el terminal que tenga activo el entorno virtual, se ejecuta el siguiente comando:

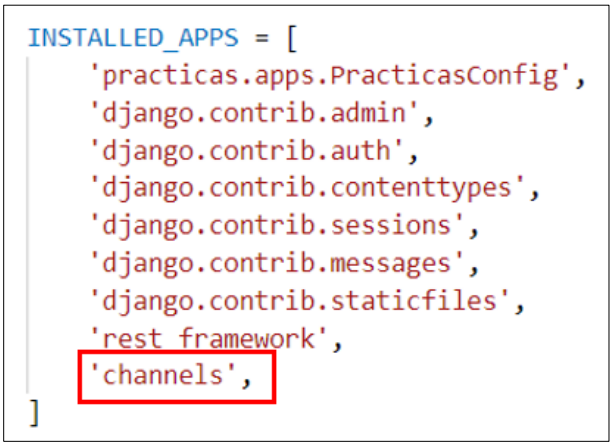
pip install channels

pip install channels-redis

Posterior a esto, hay que configurar a la función Channels en el archivo "settings.py" como muestra la Figura 134.

Figura 134

Configuración de Channels en el archivo "settings.py"



En el archivo "asgi.py" se debe declarar los tipos de conexiones que se van a realizar en la aplicación. La Figura 135 indica que en la aplicación existen dos tipos de protocolos que se van a utilizar durante su ejecución, el HTTP y WebSockets. El primero, se encarga del despliegue de páginas web estáticas, y el segundo, de interfaces dinámicas, que en este caso son las gráficas en tiempo real y la visualización de la cámara. Cabe también indicar, que ASGI permite que en la aplicación existan

métodos asíncronos, y por esto es necesaria su modificación antes de implementar los gráficos dinámicos.

Figura 135

Modificación del archivo "asgi.py" para declarar los protocolos HTTP y WebSockets

```
import os
import django

from django.core.asgi import get_asgi_application
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.auth import AuthMiddlewareStack

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'labremoto.settings')
django.setup()

from practicas.routing import ws_urlpatterns


application = ProtocolTypeRouter({
    'http':get_asgi_application(),
    'websocket' :  AuthMiddlewareStack(URLRouter(ws_urlpatterns))
})
```

Adicional a esto, se debe configurar el archivo "settings.py" para que la aplicación pueda ejecutar capas de canales en la nube. La Figura 136 indica la configuración de los caneles y Redis, estableciendo como *hosts* tanto a las capas Redis del servidor local como del servidor en la nube que se genera al ejecutar la aplicación.

Figura 136

Configuración de las capas de canales Redis en el archivo "settings.py"

```
CHANNEL_LAYERS = {
    'default' : {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
            "hosts": [os.environ.get('REDIS_URL','redis://localhost:6379')],
        }
    }
}
```

Una vez configurado los canales, Redis y el protocolo WebSocket, se crea el archivo “routing.py” cuya función va a ser generar las direcciones URL para asociar las instancias de WebSocket con el consumidor de las gráficas, tal y como muestra la Figura 137.

Figura 137

Archivo “routing.py” que genera las direcciones URL para asociar las instancias de WebSocket con los consumidores

```
from django.urls import path, re_path
from .consumers import GraphConsumer, CallConsumer, GraphConsumer2

ws_urlpatterns = [
    path('ws/graficas/', GraphConsumer.as_asgi()),
    path('ws/graficas2/', GraphConsumer2.as_asgi()),
    re_path(r'ws/call/', CallConsumer.as_asgi()),
]
```

El laboratorio remoto dispone de varias gráficas en todas las prácticas existentes. Por este motivo, se describe la implementación de una gráfica en tiempo real y en específico de corriente.

El primer paso, es desarrollar la programación HTML en el *template* donde se quiera desplegar la gráfica. El recurso HTML que permite la implementación de gráficas es “canvas” en donde se asigna un “id” para instanciar al componente, y se especifica las dimensiones de la misma, tal y como muestra la Figura 138.

Figura 138

Declaración de una gráfica con el recurso canvas en HTML

```
<div class="row">
    <canvas id="chartCorriente" width="400" height="200"></canvas>
</div>
```

Dentro del *template* se llama a un archivo estático escrito en JavaScript llamado en este caso “graf.js”, en donde se configura el tipo de gráfico, los nombres de los ejes, la escala, entre otros. Las Figuras Figura 139 y Figura 140 muestran la configuración

del gráfico de corriente de la primera práctica del laboratorio remoto que se refiere al manejo y configuración del variador de frecuencia.

Figura 139

Archivo "graf.js". Configuración de la gráfica de corriente 1/2

```
const ctx = document.getElementById('chartCorriente').getContext('2d');
const ctx2 = document.getElementById('chartVoltaje').getContext('2d');
const ctx3 = document.getElementById('chartPotencia').getContext('2d');
const ctx4 = document.getElementById('chartVelocidad').getContext('2d');
// const ctx5 = document.getElementById('chartTorque').getContext('2d');
Chart.defaults.font.size = 14;
var graphCorriente = {
    type: 'line',
    data: {
        //labels: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'],
        datasets: [{
            label: 'Corriente del motor [A]',
            data: [
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
                {x: Date.now(), y: 0},
            ],
            backgroundColor: [
                'rgba(73, 198, 230, 0.5)',
            ],
            borderWidth: 1
        }]
    },
```

Figura 140

Archivo "graf.js". Configuración de la gráfica de corriente 2/2

```
    options: {
        scales: {
            x: {
                parsing: false,
                type: 'time',
                time: {
                    unit: 'second'
                }
            }
        },
        plugins: {
            legend: {
                labels: {

                    font: {
                        size: 18
                    }
                }
            }
        }
    }
}
```

En el mismo archivo "graf.js" es donde se genera la conexión WebSocket como muestra la Figura 141. Esta declaración se asocia con la dirección URL generada anteriormente y mostrada en la Figura 137, y posteriormente accede a la clase del consumidor "GraphConsumer".

Figura 141

Archivo "graf.js". Conexión vía WebSocket para visualización de gráfica dinámica

```
let ws_scheme2 = window.location.protocol == "https:" ? "wss://" : "ws://";

var socket = new WebSocket(
        ws_scheme2
        + window.location.host
        + '/ws/graficas/'
    );
```

Luego, se implementa la clase "GraphConsumer" en el archivo "consumers.py" que se debe generar manualmente al mismo nivel que el archivo "views.py", ya que son similares. La diferencia es que los "consumers.py" se ejecutan cuando se requiere

conexión vía WebSocket, mientras que los “views.py” se ejecutan para peticiones HTTP.

El ”GraphConsumer” es un método asíncrono de larga duración y su única función es acceder a la base de datos para obtener el valor de corriente que ya fue almacenado a través de la petición “POST” explicada anteriormente. Después, envía el dato convertido en formato JSON para que pueda ser leído desde el archivo “graf.js”. Estos dos procedimientos, se pueden visualizar en la Figura 142.

Figura 142

Clase “GraphConsumer”

```
import json
from random import randint
from asyncio import sleep

from channels.generic.websocket import AsyncWebsocketConsumer

from asgiref.sync import async_to_sync
from channels.generic.websocket import WebsocketConsumer
from practicas.models import Lectura, Escritura
from channels.db import database_sync_to_async

class GraphConsumer (AsyncWebsocketConsumer):
    async def connect(self):
        await self.accept()
        while True:
            self.corriente = await database_sync_to_async(self.obtenerCorriente)()
            self.voltaje = await database_sync_to_async(self.obtenerVoltaje)()
            self.potencia = await database_sync_to_async(self.obtenerPotencia)()
            self.velocidad = await database_sync_to_async(self.obtenerVelocidad)()
            # self.torque = await database_sync_to_async(self.obtenerTorque)()
            #await self.send(json.dumps({'value': randint(-20,20)}))
            await self.send(json.dumps({'corriente': self.corriente, 'voltaje': self.voltaje,
            'potencia': self.potencia, 'velocidad': self.velocidad}))
            await sleep(1)

    def obtenerCorriente(self):
        return Lectura.objects.all()[0].corriente

    def obtenerVoltaje(self):
        return Lectura.objects.all()[0].voltaje

    def obtenerPotencia(self):
        return Lectura.objects.all()[0].potencia

    def obtenerVelocidad(self):
        return Lectura.objects.all()[0].velocidad
```

Finalmente, mediante protocolo WebSocket y desde el archivo “graf.js” se lee el dato proveniente del consumidor en formato JSON, se desplaza con el comando “shift” el primer dato de la gráfica, y se incluye con el comando “push” al dato final, tal y como muestra la Figura 143.

Figura 143

Lectura del dato en formato JSON e inclusión dentro de la gráfica de forma dinámica

```
socket.onmessage = function(e){
    var djangoData = JSON.parse(e.data);
    console.log(djangoData);

    var newGraphCorriente = graphCorriente.data.datasets[0].data;
    newGraphCorriente.shift();
    newGraphCorriente.push({x: Date.now(), y: djangoData.corriente});
    graphCorriente.data.datasets[0].data = newGraphCorriente;
    chartCorriente.update();
```

De esta manera, el canal permanece abierto todo el tiempo generando un intercambio de información bidireccional mediante WebSockets y desplegando de forma dinámica datos de corriente en la gráfica.

Transmisión de imagen en tiempo real

Al igual que en la generación de gráficas dinámicas, para utilizar WebRTC, “channels” debe estar incluido dentro de las aplicaciones instaladas, tiene que estar configurado el canal “Redis” en el archivo “settings.py”, y debe estar incluido “websocket” dentro del mapeo de tipo de protocolos de la aplicación.

Para transmitir la imagen desde la webcam instalada en el laboratorio al usuario en la aplicación web, con WebRTC se va a establecer una conexión punto a punto entre el navegador del usuario y el navegador “Chromium” instalado en el sistema operativo de la Raspberry. En primer lugar, en las Figuras Figura 144 a Figura 146 se observa la clase “CallConsumer” donde se define las acciones que el servidor debe realizar al

momento de recibir alguna orden desde el WebSocket creado del lado cliente. La clase contiene tres funciones: “connect”, “disconnect” y “receive”. El primero crea la conexión WebSocket, el segundo cierra la conexión, y el tercero maneja las peticiones enviadas del lado del cliente, las cuales pueden ser “call” para solicitar la conexión con la Raspberry, “answer_call” que envía la aceptación desde la Raspberry al cliente, y “ICEcandidate” que proporciona información sobre la dirección IP y el puerto desde donde se intercambiarán los datos. En la Figura 147 se muestra la dirección URL para instanciar un WebSocket con la clase “CallConsumer”.

Figura 144

Consumidor “CallConsumer” 1/3

```
class CallConsumer(WebsocketConsumer):
    def connect(self):
        self.accept()
        self.send(text_data=json.dumps({
            'type': 'connection',
            'data': {
                'message': "Connected"
            }
        }))

    def disconnect(self, close_code):
        async_to_sync(self.channel_layer.group_discard)(
            self.my_name,
            self.channel_name
        )
```

Figura 145

Consumidor “CallConsumer” 2/3

```
def receive(self, text_data):
    text_data_json = json.loads(text_data)
    eventType = text_data_json['type']
    if eventType == 'login':
        name = text_data_json['data']['name']
        self.my_name = name
        async_to_sync(self.channel_layer.group_add)(
            self.my_name,
            self.channel_name
        )
    if eventType == 'call':
        name = text_data_json['data']['name']
        async_to_sync(self.channel_layer.group_send)(
            name,
            {
                'type': 'call_received',
                'data': {
                    'caller': self.my_name,
                    'rtcMessage': text_data_json['data']['rtcMessage']
                }
            }
        )
    if eventType == 'answer_call':
        caller = text_data_json['data']['caller']
        async_to_sync(self.channel_layer.group_send)(
            caller,
            {
                'type': 'call_answered',
                'data': {
                    'rtcMessage': text_data_json['data']['rtcMessage']
                }
            }
        )
    if eventType == 'ICEcandidate':
        user = text_data_json['data']['user']
        async_to_sync(self.channel_layer.group_send)(
            user,
            {
                'type': 'ICEcandidate',
                'data': {
                    'rtcMessage': text_data_json['data']['rtcMessage']
                }
            }
        )
```

Figura 146

Consumidor "CallConsumer" 3/3

```
def call_received(self, event):
    self.send(text_data=json.dumps({
        'type': 'call_received',
        'data': event['data']
    }))


def call_answered(self, event):
    self.send(text_data=json.dumps({
        'type': 'call_answered',
        'data': event['data']
    }))


def ICEcandidate(self, event):
    self.send(text_data=json.dumps({
        'type': 'ICEcandidate',
        'data': event['data']
    }))
```

Figura 147

Dirección URL para asociar la instancia de WebSocket con el consumidor "CallConsumer"

```
ws_urlpatterns = [
    path('ws/graficas/', GraphConsumer.as_asgi()),
    path('ws/graficas2/', GraphConsumer2.as_asgi()),
    re_path(r'ws/call/', CallConsumer.as_asgi()),
]
```

De lado del cliente, mediante JavaScript se maneja la conexión con WebSocket, el acceso a los recursos multimedia (video) y la creación de la conexión WebRTC. En el código mostrado en las Figuras Figura 148 y Figura 149 se crea una conexión con

WebSocket. Luego escucha los eventos del socket; según el tipo de evento, maneja la interfaz de usuario y los datos.

Figura 148

Creación y manejo de eventos del WebSocket 1/2

```
let callSocket;
function connectSocket() {
    let ws_scheme = window.location.protocol == "https:" ? "wss://" : "ws://";
    callSocket = new WebSocket(
        ws_scheme
        + window.location.host
        + '/ws/call/'
    );

    callSocket.onopen = event =>{
        callSocket.send(JSON.stringify({
            type: 'login',
            data: {
                name: myName
            }
        }));
    }

    callSocket.onmessage = (e) =>{
        let response = JSON.parse(e.data);
        let type = response.type;

        if(type == 'connection') {
            console.log(response.data.message)
        }

        if(type == 'call_received') {
            onNewCall(response.data)
        }

        if(type == 'call_answered') {
            onCallAnswered(response.data);
        }
```

Figura 149

Creación y manejo de eventos del WebSocket 2/2

```
        if(type == 'ICEcandidate') {
            onICECandidate(response.data);
        }
    }

    const onNewCall = (data) =>{
        otherUser = data.caller;
        remoteRTCMessage = data.rtcMessage;
    }

    const onCallAnswered = (data) =>{
        remoteRTCMessage = data.rtcMessage
        peerConnection.setRemoteDescription(new RTCSessionDescription(remoteRTCMessage));
        callProgress()
    }

    const onICECandidate = (data) =>{
        let message = data.rtcMessage
        let candidate = new RTCIceCandidate({
            sdpMLineIndex: message.label,
            candidate: message.candidate
        });

        if (peerConnection) {
            peerConnection.addIceCandidate(candidate);
        } else {
            iceCandidatesFromCaller.push(candidate);
        }
    }
}
```

Por otro lado, en las Figuras Figura 150 y Figura 151 se observan las funciones para definir los recursos multimedia que van a transmitirse entre el cliente y la maqueta, además de la creación del canal para la comunicación en tiempo real. Si observamos el diagrama mostrado en la Figura 36, es necesario un servidor STUN para descubrir las direcciones IP públicas del usuario y la Raspberry Pi 4 en el laboratorio para crear el enlace directo o, en su defecto, un servidor TURN para transmitir el video a través de este. Para la aplicación, solo se utiliza un servidor STUN, en concreto, *stun:stun.l.google.com:19302* de Google, que se indica en la Figura 152.

Figura 150

Definición de los recursos multimedia y creación de la conexión WebRTC 1/2

```
function beReady() {
    return navigator.mediaDevices.getUserMedia({
        audio: false,
        video: true,
    })
        .then(stream => {
            localStream = stream;
            localVideo.srcObject = stream;
            var videoTracks = stream.getVideoTracks();
            videoTracks[0].enabled = false;
            return createConnectionAndAddStream()
        })
        .catch(function (e) {
            alert('getUserMedia() error: ' + e.name);
        });
}

function createConnectionAndAddStream() {
    createPeerConnection();
    peerConnection.addStream(localStream);
    return true;
}

function createPeerConnection() {
    try {
        peerConnection = new RTCPeerConnection(pcConfig);
        peerConnection.onicecandidate = handleIceCandidate;
        peerConnection.onaddstream = handleRemoteStreamAdded;
        peerConnection.onremovestream = handleRemoteStreamRemoved;
        return;
    } catch (e) {
        alert('Cannot create RTCPeerConnection object.');
        return;
    }
}
```

Figura 151

Definición de los recursos multimedia y creación de la conexión WebRTC 2/2

```
function handleIceCandidate(event) {
    if (event.candidate) {
        console.log("Local ICE candidate");
        sendICEcandidate({
            user: otherUser,
            rtcMessage: {
                label: event.candidate.sdpMLineIndex,
                id: event.candidate.sdpMid,
                candidate: event.candidate.candidate
            }
        })
    } else {
        console.log('End of candidates.');
    }
}

function handleRemoteStreamAdded(event) {
    remoteStream = event.stream;
    remoteVideo.srcObject = remoteStream;
}

function handleRemoteStreamRemoved(event) {
    remoteVideo.srcObject = null;
    localVideo.srcObject = null;
}
```

Figura 152

Definición del servidor STUN

```
let pcConfig = {
    "iceServers":
        [
            {"url": "stun:stun.l.google.com:19302"}
        ]
};
```

Por último, en la Figura 153 se muestran las funciones que permiten al usuario, enviar una oferta de conexión punto a punto a la Raspberry Pi 4 al momento de iniciar la

práctica. Esto se realiza enviando una oferta tipo “call” con el nombre “Lab” a través del WebSocket para que el servidor sea en el encargado de enviar la oferta a la sesión en la Raspberry. Una vez que sea respondida la oferta, se ejecuta la función mostrada en la Figura 154, donde se añade la descripción de sesión de la Raspberry en el canal y comienza la transmisión.

Figura 153

Funciones para enviar una oferta de conexión a la Raspberry Pi 4

```
function call() {
    let userToCall = "Lab";
    otherUser = userToCall;
    beReady()
        .then(bool => {
            processCall(userToCall)
        })
}

function processCall(userName) {
    peerConnection.createOffer((sessionDescription) => {
        peerConnection.setLocalDescription(sessionDescription);
        sendCall({
            name: userName,
            rtcMessage: sessionDescription
        })
    }, (error) => {
        console.log("Error");
    });
}

/**
 *
 * @param {Object} data
 * @param {number} data.name
 * @param {Object} data.rtcMessage
 */
function sendCall(data) {
    console.log("Send Call");
    callSocket.send(JSON.stringify({
        type: 'call',
        data
    }));
}
```

Figura 154

Configuración de la descripción de sesión remota para iniciar la transmisión

```
const onCallAnswered = (data) =>{
    remoteRTCMessage = data.rtcMessage
    peerConnection.setRemoteDescription(new RTCSessionDescription(remoteRTCMessage));
    localStream.getTracks().forEach(track => track.stop());
    callProgress()
}
```

En la aplicación web se crea una dirección URL y *template* único para la Raspberry Pi 4 con el fin de poder crear el canal y transmitir la imagen al usuario. Dentro del *template* se añade un script con la misma programación utilizada del lado del usuario, a excepción de las funciones que se encuentran en las Figuras Figura 153 y Figura 154; en su lugar, se crean las funciones de las Figuras Figura 155 y Figura 156 para responder a la oferta de conexión con la descripción de sesión. Esta descripción incluye información sobre el tipo de medios que se envían, su formato, el protocolo de transferencia que se utiliza, la dirección IP y el puerto del extremo.

Figura 155

Funciones para responder a la oferta de conexión desde la Raspberry 1/2

```
function answer() {
    beReady()
        .then(bool => {
            processAccept();
        })
}

function processAccept() {
    peerConnection.setRemoteDescription(new RTCSessionDescription(remoteRTCMessage));
    peerConnection.createAnswer((sessionDescription) => {
        peerConnection.setLocalDescription(sessionDescription);
        if (iceCandidatesFromCaller.length > 0) {
            for (let i = 0; i < iceCandidatesFromCaller.length; i++) {
                let candidate = iceCandidatesFromCaller[i];
                try {
                    peerConnection.addIceCandidate(candidate).then(done => {
                        console.log(done);
                    }).catch(error => {
                        console.log(error);
                    })
                } catch (error) {
                    console.log(error);
                }
            }
            iceCandidatesFromCaller = [];
        }
        answerCall({
            caller: otherUser,
            rtcMessage: sessionDescription
        })
    }, (error) => {
        console.log("Error");
    })
}
```

Figura 156

Funciones para responder a la oferta de conexión desde la Raspberry 2/2

```
/**
 *
 * @param {Object} data
 * @param {number} data.caller
 * @param {Object} data.rtcMessage
 */
function answerCall(data) {
    callSocket.send(JSON.stringify({
        type: 'answer_call',
        data
    }));
    callProgress();
}
```

Librerías utilizadas en la aplicación

Después de desarrollar e implementar toda la aplicación, el entorno virtual almacena todas las librerías que fueron utilizadas con el objetivo de cumplir múltiples funcionalidades dentro del proyecto. Para visualizar todos estos paquetes, se ejecuta el siguiente comando:

pip freeze

De forma resumida, se exponen las librerías utilizadas en la aplicación del laboratorio remoto con su respectiva versión.

- aioredis==1.3.1
- asgiref==3.4.1
- async-timeout==4.0.2
- attrs==21.4.0
- autobahn==21.11.1
- Automat==20.2.0
- backports.zoneinfo==0.2.1

- bitstring==3.1.9
- certifi==2021.10.8
- cffi==1.15.0
- channels==3.0.4
- channels-redis==3.3.1
- charset-normalizer==2.0.10
- constantly==15.1.0
- cryptography==36.0.1
- cycler==0.11.0
- daphne==3.0.2
- Deprecated==1.2.13
- dj-database-url==0.5.0
- Django==4.0
- django-filter==21.1
- django-redis==5.2.0
- djangorestframework==3.13.1
- fonttools==4.28.3
- gunicorn==20.1.0
- hiredis==2.0.0
- hyperlink==21.0.0
- idna==3.3
- importlib-metadata==4.10.0
- incremental==21.3.0
- kiwisolver==1.3.2
- Markdown==3.3.6
- matplotlib==3.5.0

- msgpack==1.0.3
- netifaces==0.11.0
- numpy==1.21.4
- opencv-python==4.5.4.60
- packaging==21.3
- pandas==1.3.4
- Pillow==8.4.0
- psycopg2==2.9.3
- pyasn1==0.4.8
- pyasn1-modules==0.2.8
- pycparser==2.21
- pyOpenSSL==21.0.0
- pyparsing==3.0.6
- python-dateutil==2.8.2
- python-decouple==3.5
- python-snap7 @ git+https://github.com/gijzelaerr/python-snap7.git@2e7e3668aa16497fa1630133d96a6e360c0128b3
- pytz==2021.3
- redis==4.1.0
- requests==2.27.1
- service-identity==21.1.0
- setuptools-scm==6.3.2
- six==1.16.0
- sqlparse==0.4.2
- tomli==1.2.2
- Twisted==21.7.0

- txaio==21.2.1
- typing_extensions==4.0.1
- tzdata==2021.5
- urllib3==1.26.7
- whitenoise==5.3.0
- wincertstore==0.2
- wrapt==1.13.3
- zipp==3.7.0
- zope.interface==5.4.0

Capitulo V. Pruebas y Resultados

En este capítulo se pone a prueba el laboratorio remoto en aspectos de conectividad, funcionamiento del hardware y resultados de aprendizaje en cada una de las prácticas.

Pruebas de conectividad del laboratorio remoto

Un factor importante en el funcionamiento del laboratorio remoto es el flujo de información desde la aplicación web hacia la maqueta y viceversa, tanto para manejar los equipos en el momento oportuno, como en la visualización de las variables, a fin de observar fielmente su evolución y sacar las conclusiones debidas. Por lo tanto, se evalúa la conectividad de todo el sistema que conforma el laboratorio remoto en términos de fiabilidad y tiempos de respuesta.

Fiabilidad

En esta sección se prueba si existe garantía de recepción en la maqueta, al momento de realizar cualquier acción en la aplicación web que conlleve un cambio en el funcionamiento de la maqueta. Para lo cual, se plantea dos tipos de pruebas: la transmisión de datos desde la aplicación web hacia el variador de frecuencia y hacia el HMI.

Transmisión de datos entre la aplicación web y el variador de frecuencia. Para realizar esta prueba, se accede remotamente a la interfaz de la primera práctica de laboratorio, en donde se tiene que configurar los parámetros del variador de frecuencia y posteriormente arrancar el motor de la maqueta.

Una vez que se oprime el botón para configurar los parámetros del variador, se verifica si en la maqueta se asigna los valores establecidos en la interfaz web. Luego, se oprime el botón de marcha, para verificar si el motor arranca; y el botón de paro, para visualizar si el motor se detiene.

De esta manera, se pone en prueba la conectividad que tiene el laboratorio remoto con la maqueta. La Tabla 14 resume las pruebas realizadas.

Tabla 14

Pruebas de conectividad entre la aplicación web y el variador de frecuencia implemento en la maqueta

Número de prueba	Fecha y hora	Frecuencia [Hz]	Tiempo de aceleración [s]	Tiempo de desaceleración [s]	Respuesta en la maqueta
1	24/01/2022 13:15	20	4	4	Arranque y paro del motor en 1 solo clic
2	24/01/2022 13:25	35	5	5	Arranque y paro del motor en 1 solo clic
3	24/01/2022 13:40	50	6	6	Arranque y paro del motor en 1 solo clic

Para la prueba 1, se configura en la aplicación una frecuencia de 20 Hz y unos tiempos de aceleración y desaceleración de 4 segundos. Una vez oprimido el botón “Configurar”, se evidencia en la Figura 157, como la referencia de frecuencia del variador de la maqueta toma el valor de 20 Hz configurado en la aplicación web.

Figura 157

Resultado Prueba 1 - Captura de imagen de la aplicación web que muestra la maqueta en funcionamiento con una frecuencia de 20 Hz en el variador

De forma similar, se hicieron las pruebas 2 y 3, cuyas configuraciones de frecuencia en la aplicación web son de 35 y 50 Hz con unos tiempos de aceleración y desaceleración de 5 y 6 segundos, respectivamente. Los resultados de estas pruebas, se representan en las Figuras Figura 158 y Figura 159. Después de realizar las tres pruebas de comunicación entre la interfaz web y el laboratorio remoto, se demuestra que la conectividad del laboratorio remoto es óptima en términos de fiabilidad.

Figura 158

Resultado Prueba 2 - Captura de imagen de la aplicación web que muestra la maqueta en funcionamiento con una frecuencia de 35 Hz en el variador

Figura 159

Resultado Prueba 3 - Captura de imagen de la aplicación web que muestra la maqueta en funcionamiento con una frecuencia de 50 Hz en el variador

Transmisión de datos entre la aplicación web y el panel de operador. La comunicación entre la interfaz web y la HMI involucra indirectamente al PLC, ya que este último mantiene conexión directa con el panel de operador. Además, si la información llega de forma oportuna al HMI, se determina que el flujo de información es correcto al cruzar por todos los componentes del presente trabajo, y que se enumeran a continuación:

1. Aplicación web
2. Raspberry Pi 4
3. PLC Siemens S7-1200
4. Panel de operador Schneider HMI STU855

Las pruebas que se realizan en esta sección se enfocan en las prácticas de laboratorio 2 y 3, que consisten en el manejo del sistema de bombas. Para estos casos, se configura los parámetros del controlador desde la aplicación web, y se comprueba si los datos llegan hasta el panel de operador. La Tabla 15 resume mencionadas pruebas.

Tabla 15

Pruebas de conectividad entre la aplicación web y el HMI implemento en la maqueta

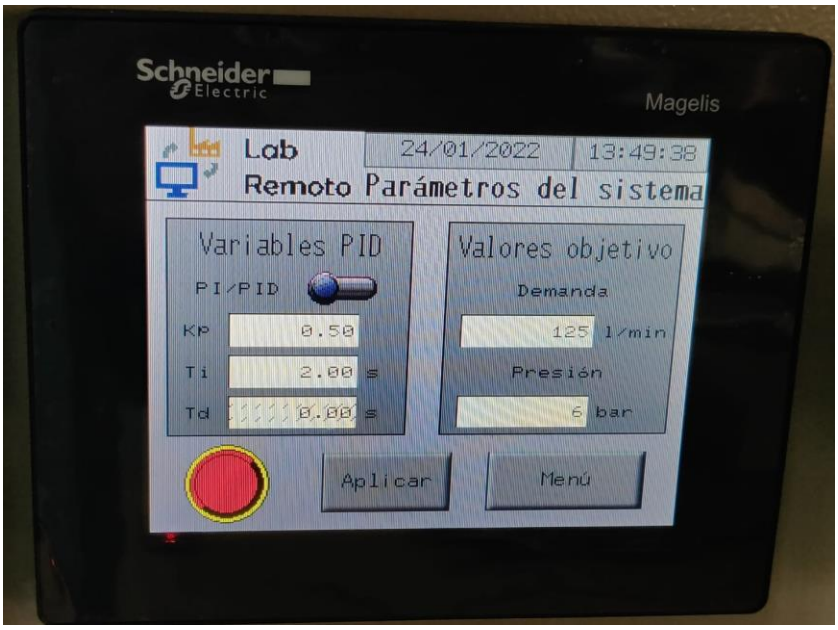
Número de práctica	Fecha y hora	Demanda [l/min]	Tipo de controlador	Ganancia Kp	Tiempo de integración Ti [s]	Tiempo de derivación Td [s]
2	24/01/2022 13:49	125	PI	0.5	2	0
3	24/01/2022 13:47	75	PID	0.7	1.5	0.1

La primera prueba se realiza al ejecutar la práctica de laboratorio 2, para lo cual, se selecciona un controlador PI, referencia de demanda 125 l/min, con constante

proporcional y tiempo de integración de 0.5 y 2 segundos, respectivamente. Cuando se configura estos parámetros desde la aplicación web, el resultado en el panel de operador se muestra en la Figura 160.

Figura 160

Resultado Prueba 1 - Conectividad entre la aplicación web y el panel de operador



En cambio, la segunda prueba se realiza al ejecutar la práctica de laboratorio 3. Los parámetros configurados y enviados desde la aplicación web son, controlador PID, demanda inicial 75 l/min, con constante proporcional de 0.7, tiempo de integración y derivación de 1.5 y 0.1 segundos respectivamente. El resultado en el panel de operador se indica a través de la Figura 161.

Figura 161

Resultado Prueba 2 - Conectividad entre la aplicación web y el panel de operador

Es importante mencionar que la transmisión de los datos desde la interfaz web hacia el panel de operador se consigue cuando se oprime por primera vez el botón de configurar parámetros del controlador, lo que indica, que la conectividad es instantánea e idónea. Adicionalmente, evita que el usuario pueda enviar reiterados datos que pongan en riesgo a los componentes implementos en la maqueta.

Al finalizar estas dos verificaciones, se demuestra también, que el flujo de datos recorre todos los componentes del entorno hardware que se implementaron.

Evidentemente, al ser una conexión remota, existe un pequeño tiempo de retardo en el envío de datos, sin embargo, este tema se describe en el siguiente apartado.

Tiempos de transmisión

En este punto, se mide el tiempo que tarda una acción realizada en la aplicación web en modificar el funcionamiento de la maqueta y, de manera similar, cuanto tiempo se demora en reflejarse en la interfaz un cambio en el estado de la maqueta.

Ciclos de envío y recepción de datos de la maqueta al servidor de la aplicación web. Los programas de lectura y escritura de datos en el PLC de las Figuras Figura 77 y Figura 78, al igual que los códigos para hacer las peticiones “POST” y “GET” al servidor de la aplicación web que se indican es las Figuras Figura 131 y Figura 132, se colocan en un solo *script* de Python dentro de un ciclo infinito para que constantemente se mantenga la comunicación de la aplicación con la maqueta. El programa de Python completo se encuentra en el ar B. La secuencia de ejecución se organizó de la siguiente manera:

1. Lectura de datos del PLC.
2. Petición HTTP “POST” para envío de datos al servidor.
3. Petición HTTP “GET” para recepción de datos desde el servidor.
4. Escritura de datos en el PLC.

En esta prueba, se mide el tiempo que tarda en ejecutarse estas cuatro acciones que corresponden a un ciclo dentro del *script*. En tres días diferentes se toman 20 tiempos, obteniendo los datos mostrados en la Tabla 16.

Tabla 16

Ciclos de envío y recepción de datos de la maqueta a la aplicación web

Lunes 24 de enero de 2022		**Martes 25 de enero de 2022**		**Miércoles 26 de enero de 2022**	
Número de ciclo	**Tiempo de ejecución [s]**	**Número de ciclo**	**Tiempo de ejecución [s]**	**Número de ciclo**	**Tiempo de ejecución [s]**

1	1.925	1	1.949	1	2.031
2	1.861	2	1.886	2	2.000
3	1.973	3	1.903	3	1.898
4	1.929	4	1.940	4	1.876
5	1.906	5	1.822	5	1.973
6	1.822	6	2.15	6	1.922
7	1.881	7	1.860	7	1.925
8	1.897	8	1.930	8	1.828
9	1.846	9	2.121	9	1.854
10	1.915	10	2.008	10	1.855
11	1.783	11	1.959	11	1.863
12	1.879	12	1.993	12	1.811
13	1.984	13	2.032	13	2.088
14	1.919	14	1.914	14	1.924
15	1.842	15	2.751	15	1.928
16	1.802	16	2.329	16	2.042
17	1.823	17	1.911	17	2.089
18	2.003	18	2.126	18	1.901
19	2.521	19	2.006	19	2.043
20	1.810	20	1.982	20	1.979

A partir de estas muestras se elabora el histograma de la Figura 162 y se obtiene algunos indicadores de estadística descriptiva detallados en la Tabla 17. Entre los datos a destacar, existe una media de tiempo de 1.962 s, llegando a un mínimo de 1.783 s y un máximo de 2.751 s. Adicionalmente, la desviación estándar es de 0.162 s, con lo cual, el 95% de los tiempos de ejecución se encuentran entre 1.8 s y 2.286 s,

rango aceptable para permitir al estudiante visualizar el funcionamiento de la maqueta y que pueda sacar conclusiones acordes al desempeño real del sistema. Cabe mencionar que, durante las pruebas realizadas, en un par de ocasiones el tiempo de ciclo se acercaba a los 4 segundos, tiempo que podría hacer que se pierda un dato y disminuir la veracidad de la evolución de las variables mostradas en la aplicación web; sin embargo, como se menciona, estos tiempos casi nunca se presentan, con lo cual, es muy poco probable que ocurra durante el arranque o parada del motor en el caso de la práctica 1, o en alguna transición de demanda durante las prácticas 2 o 3.

Figura 162

Histograma de los tiempos de ejecución

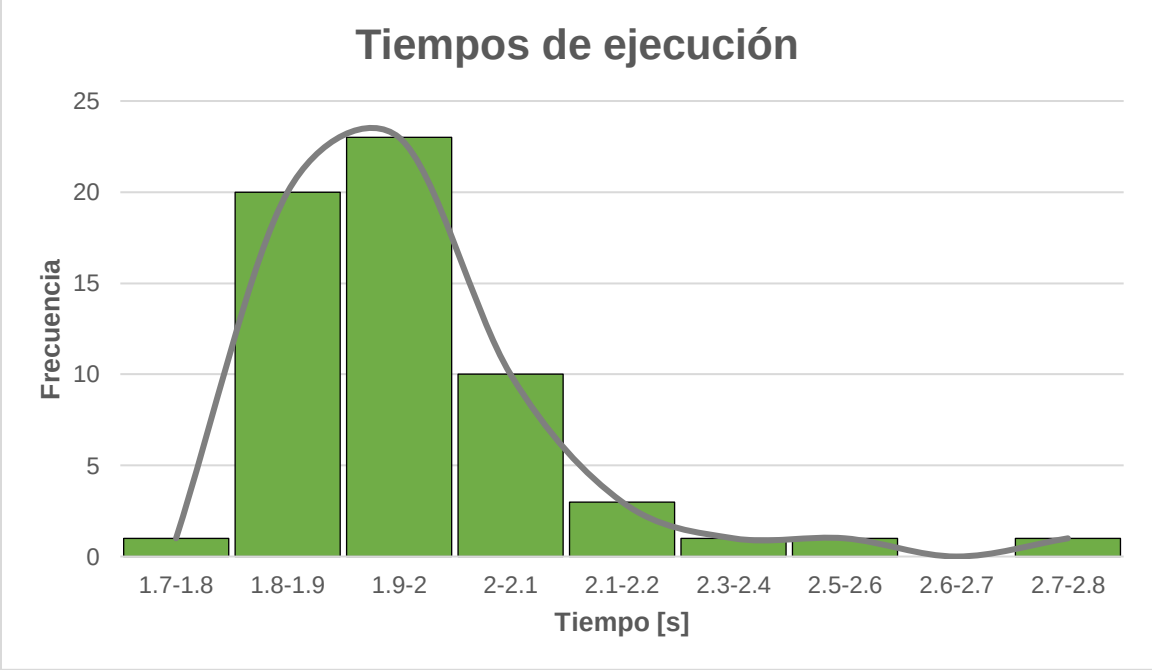


Tabla 17

Estadística descriptiva de los tiempos de ejecución

Media	1.962 s
Mediana	1.924 s
Moda	1.925 s

Desviación estándar	0.162 s
Rango	0.968 s
Mínimo	1.783 s
Máximo	2.751 s

Tiempos de respuesta de la maqueta. En esta prueba se mide el tiempo que tarda en responder la maqueta al momento de realizar alguna acción en la aplicación web. Para lo cual, se mide el tiempo que transcurre en tres escenarios: actualización de las configuraciones en el variador durante la práctica 1, arranque del motor luego de oprimir marcha y visualización en la HMI de los parámetros ingresados para el controlador PID durante la práctica 2.

Tabla 18

Tiempos de actualización de las configuraciones en el variador

Número de prueba	Tiempo [s]
1	2.67
2	2.48
3	1.95
4	2.6
5	2.96

Tabla 19

Tiempos de arranque del motor

Número de prueba	Tiempo [s]
1	1.52
2	1.6

3	2.12
4	1.97
5	1.92

Tabla 20

Tiempos de visualización en la HMI de los parámetros ingresados para el controlador PID

Número de prueba	Tiempo [s]
1	1.62
2	1.81
3	1.72
4	1.22
5	1.5

Para cada uno de los escenarios se tomó el tiempo en 5 pruebas diferentes. Los resultados se indican en las Tablas Tabla 18, Tabla 19 y Tabla 20. Si obtenemos la media de las 5 pruebas, tenemos que el tiempo promedio de actualización de las configuraciones en el variador es de 2.532 s, el tiempo promedio de arranque del motor es de 1.826 s, y para visualización en la HMI de los parámetros ingresados para el controlador PID el tiempo es de 1.574 s.

Según estos resultados, el menor tiempo es la actualización de la información en la HMI, mientras que, el arranque del motor y configuración de los parámetros en el variador tarda 0.3 s y casi 1 s más, respectivamente. Dado que la escritura en los registros para controlar el arranque y asignar la referencia de velocidad en el variador desde el PLC se realiza en la misma instancia de cliente MODBUS TCP/IP, estas acciones debería realizarse en el mismo tiempo, con lo cual, esta diferencia de casi 0.7

s, se le atribuye al lapso que tarda el variador en mostrar en la pantalla. Por lo tanto, los tiempos de respuesta aproximados del variador y el panel de operador son 1.826 y 1.574 segundos, respectivamente.

Pruebas de funcionamiento del laboratorio remoto

Habiendo comprobado la fiabilidad de manejo de la maqueta desde la aplicación, y que los tiempos de conexión son adecuados para realizar las prácticas de forma remota, ahora se comprueba que la aplicación web cumpla con las funcionalidades previstas, y que el funcionamiento del sistema durante el desarrollo de las prácticas sea el correcto.

Durante todas las pruebas de funcionamiento, un autor está presente en el laboratorio de Robótica y PLCs de la universidad, para preparar la maqueta y vigilar que no ocurra ningún evento imprevisto que pueda provocar algún daño, mientras que otro autor desarrolla las tres prácticas desde su domicilio en Tumbaco.

Funcionalidades del laboratorio remoto

En este punto se verifica que los apartados detallados en la sección de diseño como funcionalidades del laboratorio remoto, se encuentren en la aplicación web y que funcionen adecuadamente.

En primer lugar, se prueba el registro de usuario y posterior inicio de sesión como se indican en las Figuras Figura 163 y Figura 164. Además de comprobar que no exista ningún error, se verifica la validación de usuario único y coincidencia de claves durante el registro, y el ingreso de usuario con la clave correcta al momento de iniciar sesión.

Figura 163

Prueba de registro de usuario

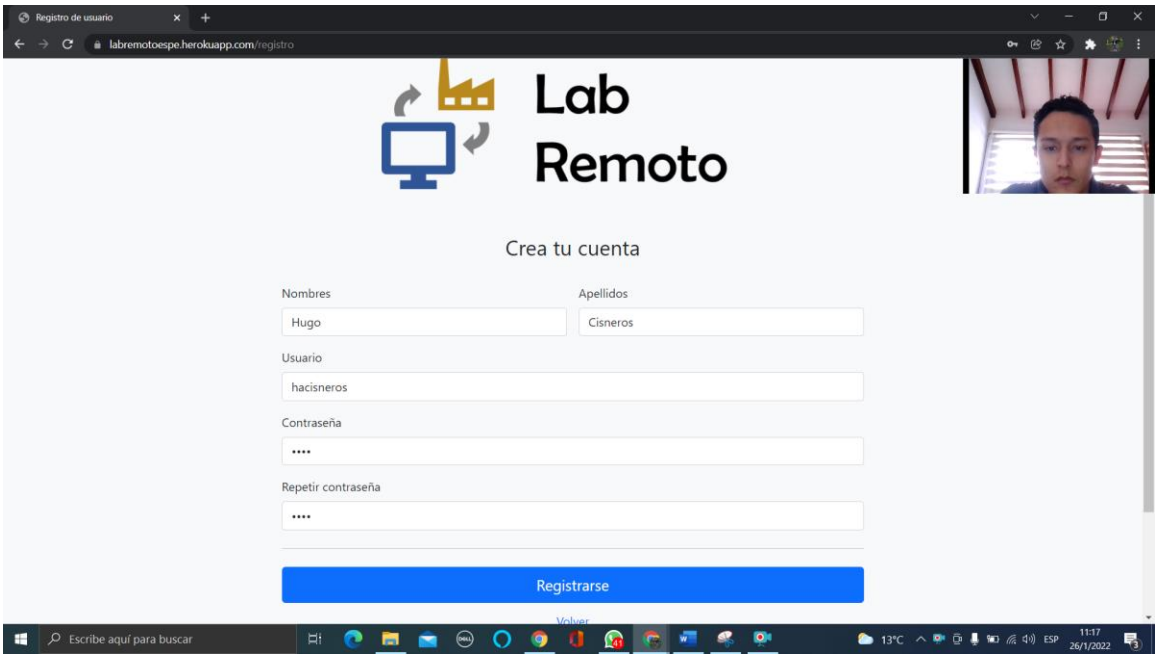
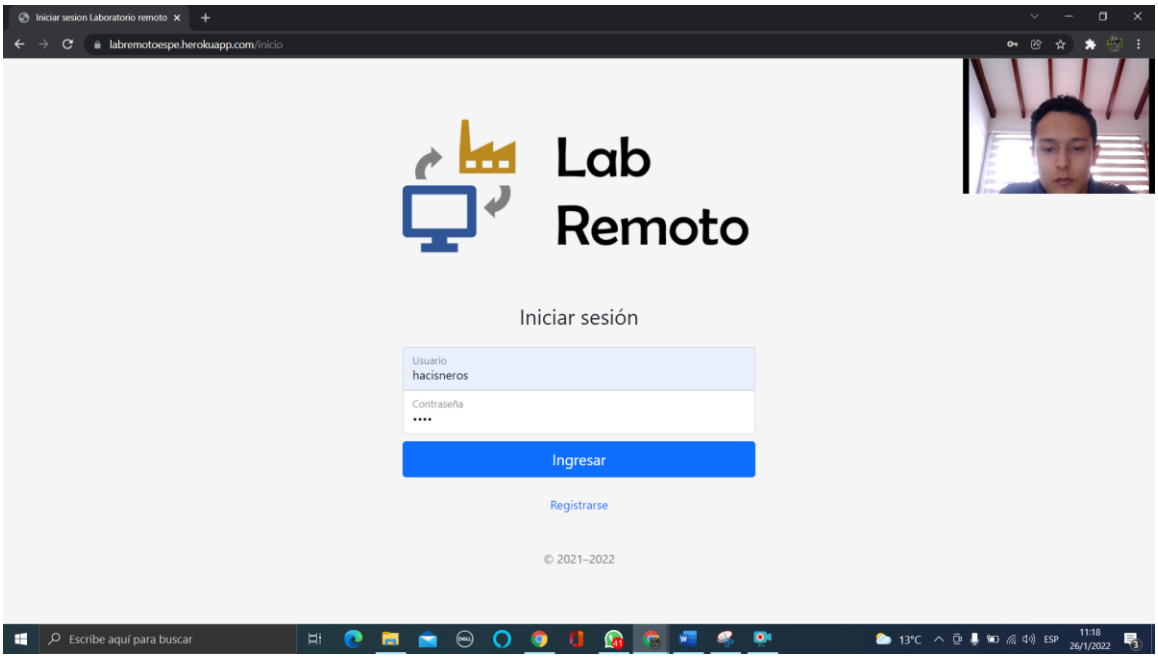


Figura 164

Inicio de sesión con el nuevo usuario creado



Después se comprueba el agendamiento de horario y selección de práctica. En la Figura 165 se observa la interfaz con el calendario para escoger el día y la hora de la práctica; en esta prueba se escoge como horario: miércoles 26 de enero de 2022, de 9:30 – 11:30 para tener acceso a la interfaz de la práctica en ese momento. Después, al desplegarse la interfaz de la Figura 166, escoge la primera práctica denominada “Configuración del variador y puesta en marcha del motor”. Al regresar a la pantalla principal mostrada en la Figura 167, se constata que la práctica fue reservada con las opciones escogidas.

Figura 165

Selección del horario para la práctica

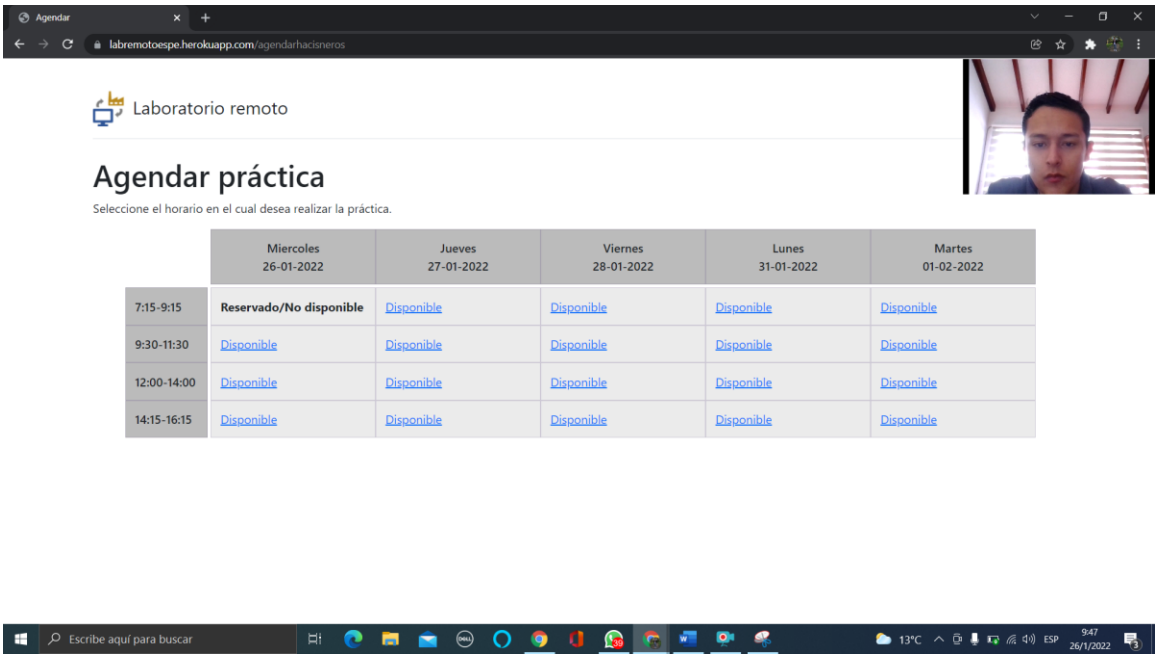


Figura 166

Selección de la práctica "Configuración del variador y puesta en marcha del motor"

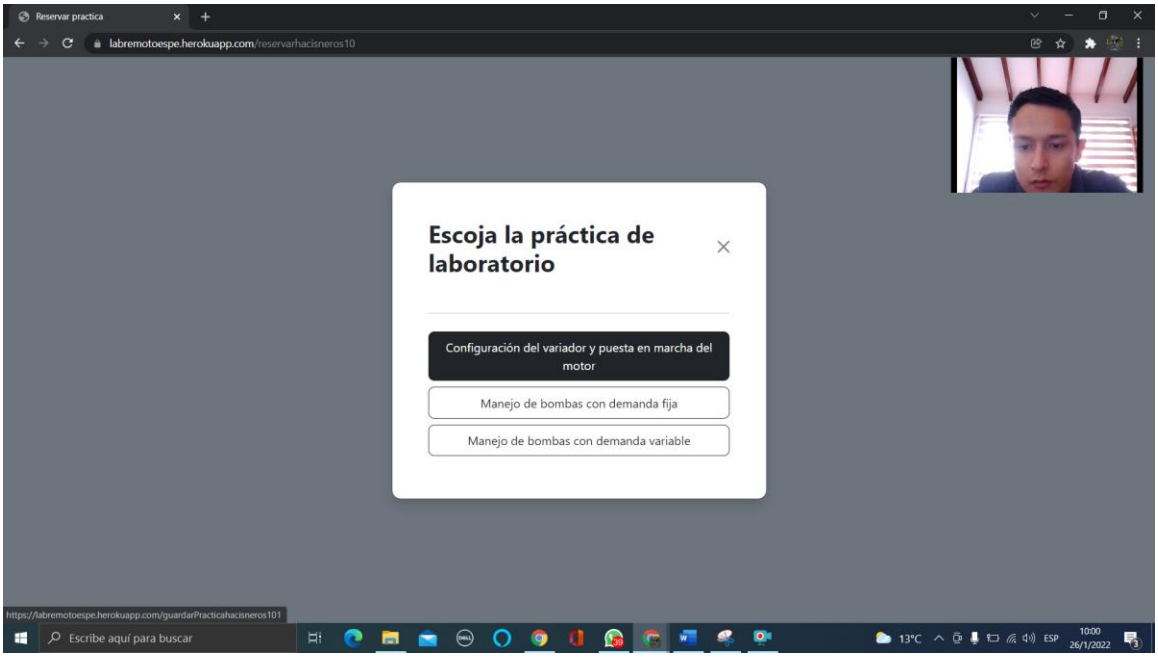
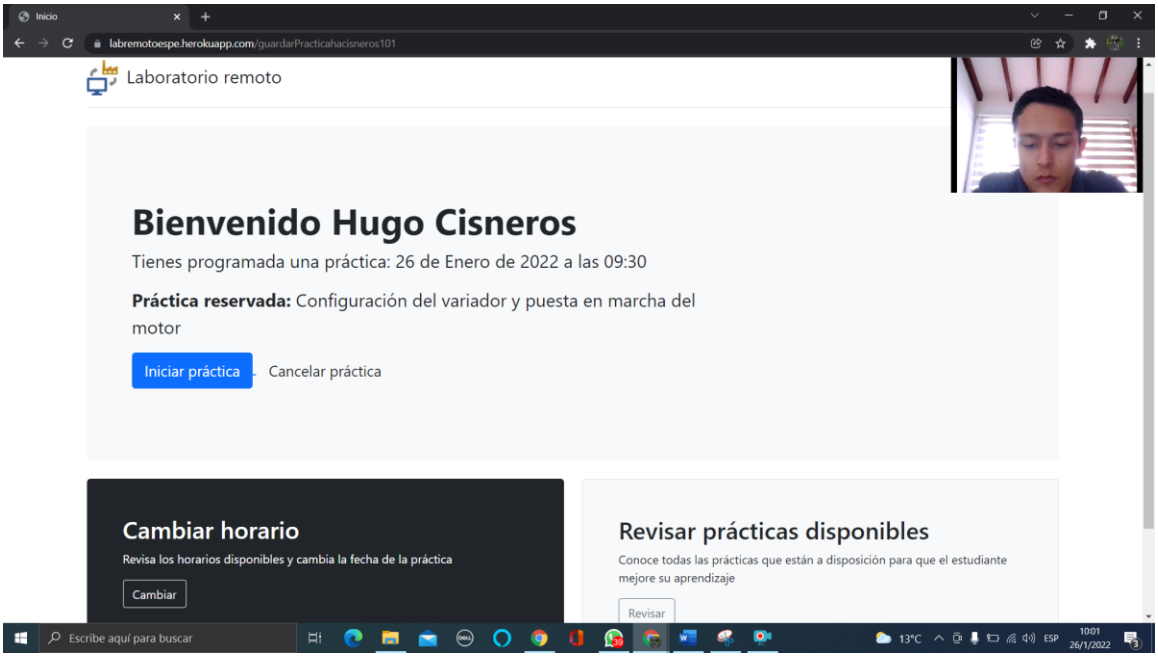


Figura 167

Pantalla principal con la práctica "Configuración del variador y puesta en marcha del motor" agendada



Realizando la misma secuencia, se puede cambiar el horario de la práctica sin ningún problema, mientras que, si se desea cambiar la práctica, mas no el horario, es necesario primero cancelar la práctica y agendar nuevamente. En las Figuras Figura 168 y Figura 169 se muestra la pantalla principal habiendo agendado las otras dos prácticas en el mismo horario.

Figura 168

Pantalla principal con la práctica "Manejo de bombas con demanda fija" agendada

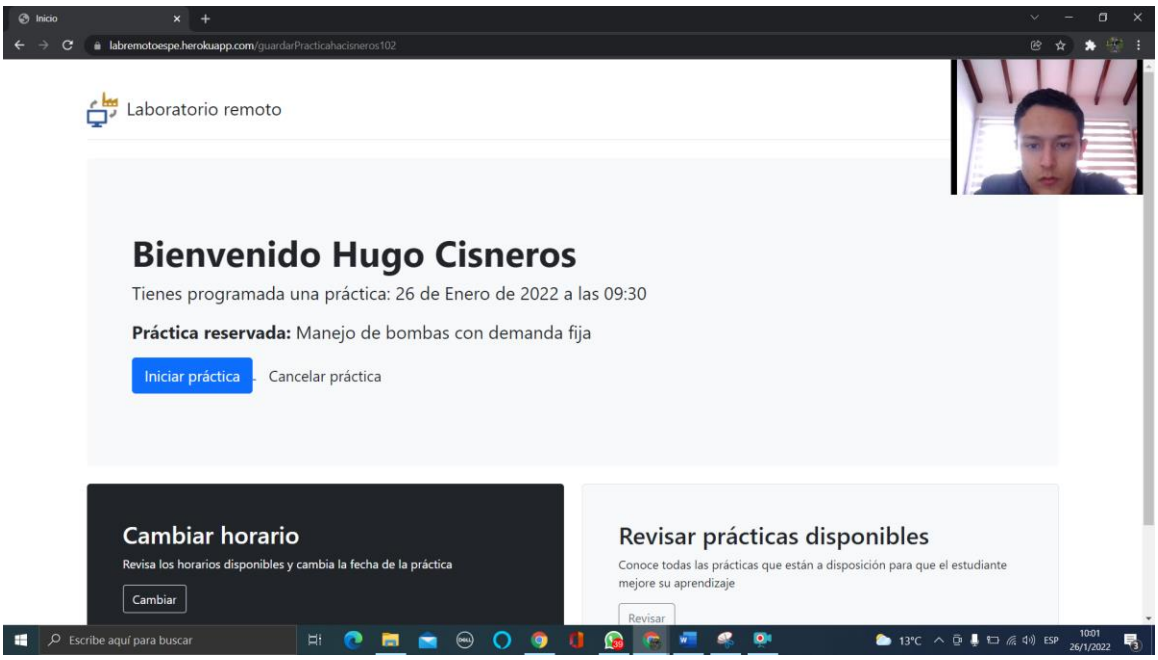
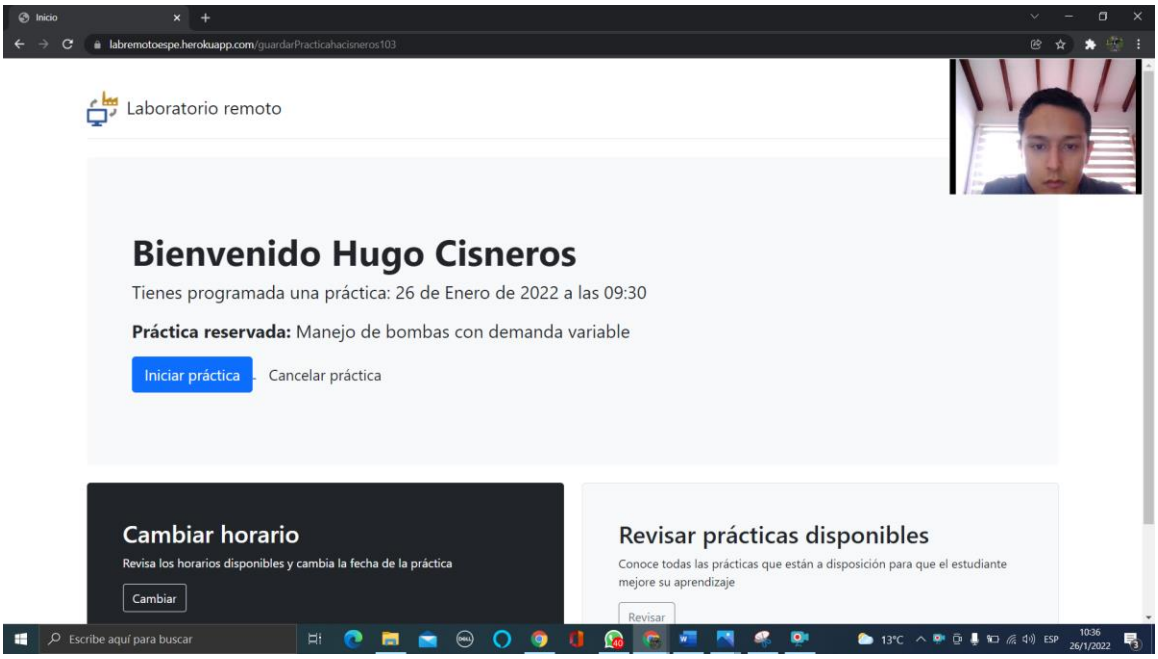


Figura 169

Pantalla principal con la práctica “Manejo de bombas con demanda variable” agendada



Por otro lado, para comprobar la opción de generar anuncios por parte del profesor, se crea un usuario tipo docente. El proceso para ello, es crear un usuario de la misma forma que se mostró anteriormente. Luego, desde la sección para manejar la base de datos que ofrece Django, se indica que dicha cuenta sea considerada como “Cuenta docente”, tal como se observa en la Figura 170. Durante las pruebas de todos los apartados, la aplicación web funciona sin errores y los datos se almacenan correctamente en la base de datos. Por otro lado, las funcionalidades de visualización de prácticas en tiempo real y descarga de gráficas en formato PNG, se verifican durante las pruebas de funcionamiento en el desarrollo de las prácticas.

Al momento de iniciar sesión con dicha cuenta, la interfaz principal cambia a la mostrada en la Figura 171. Allí se escoge la práctica en la cual se desea agregar el anuncio para que se muestre a la hora que el estudiante esté realizando la práctica. En la Figura 172 se observa el anuncio que será mostrado durante la práctica.

Durante las pruebas de todos los apartados, la aplicación web funciona sin errores y los datos se almacenan correctamente en la base de datos. Por otro lado, las funcionalidades de visualización de prácticas en tiempo real y descarga de gráficas en formato PNG, se verifican durante las pruebas de funcionamiento en el desarrollo de las prácticas.

Figura 170

Configuración del usuario como "Cuenta profesor"

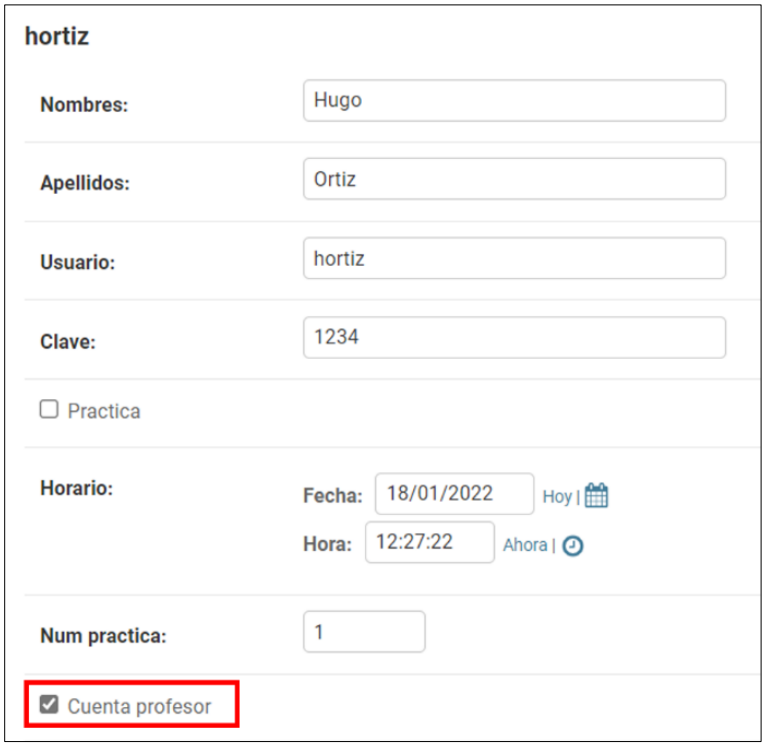


Figura 171

Pantalla principal en una "Cuenta profesor"

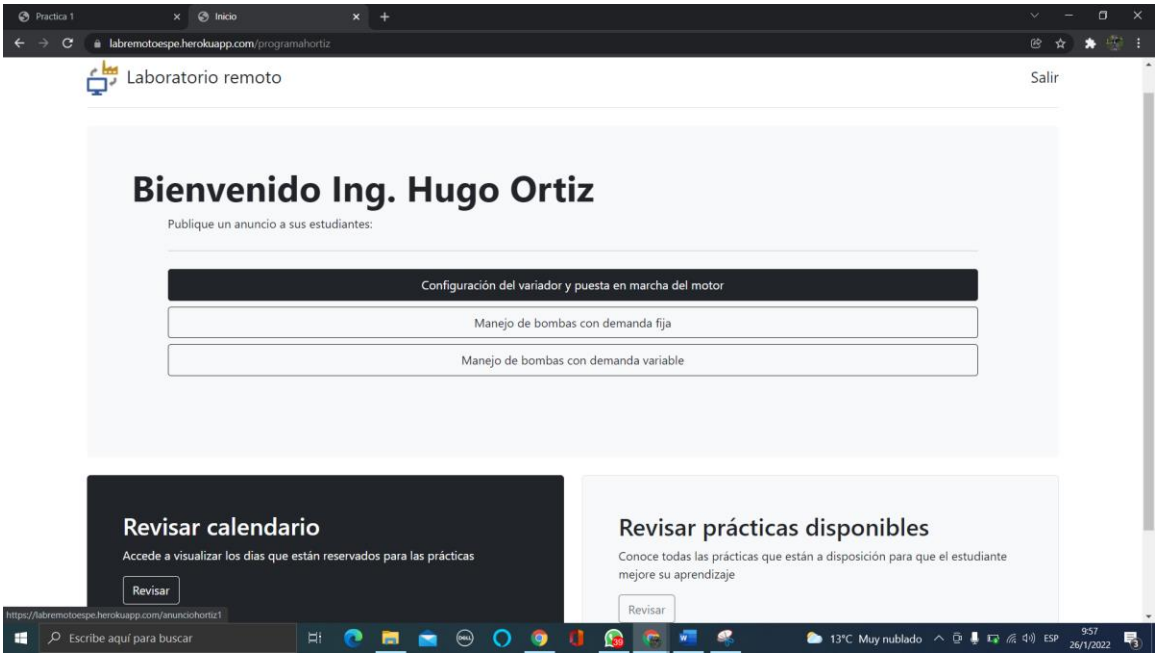
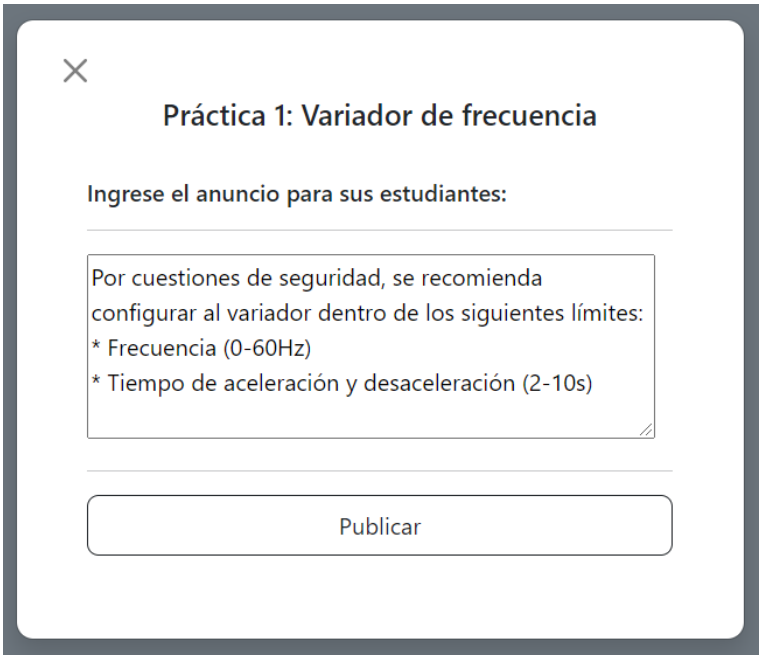


Figura 172

Ingreso de anuncio para la práctica 1

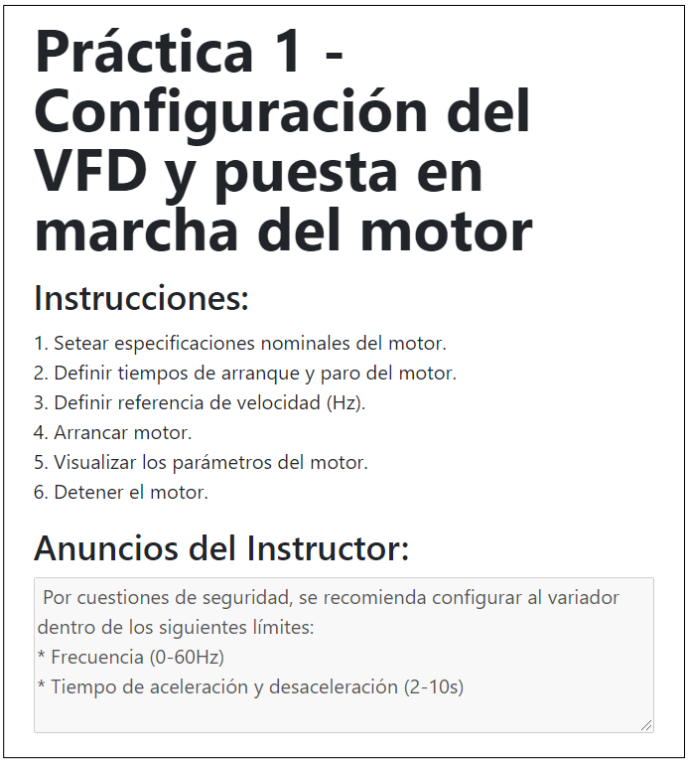


Práctica 1: Configuración y manejo del variador de frecuencia

A priori, resulta ser una práctica básica, pero que permite entender las características principales del funcionamiento del variador de frecuencia Schneider Altivar 630.

Figura 173

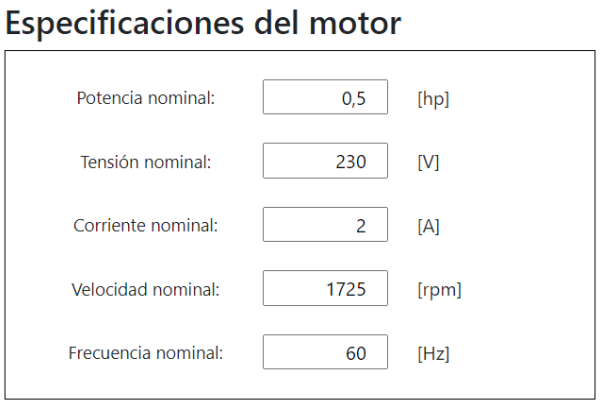
Indicaciones de la práctica 1



Dentro de la interfaz web de esta práctica, y después de que el usuario haya leído las indicaciones señaladas en la Figura 173, el primer paso a realizar es la verificación y asignación de los parámetros nominales del motor que está asociado al variador. Para el presente trabajo, ya se menciona anteriormente, que se utiliza un motor Baldor de 1/2 hp, cuyos valores nominales se deben asignar como en la Figura 174.

Figura 174

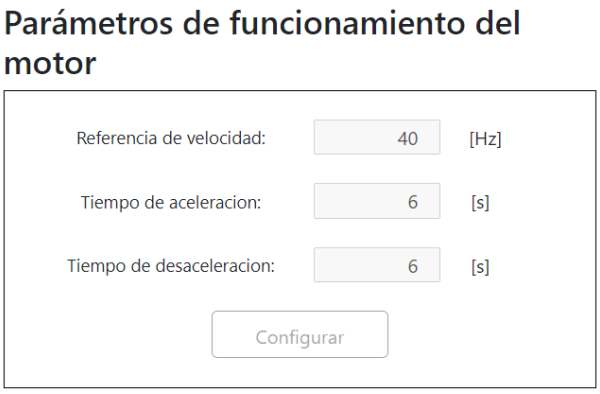
Ingreso de las especificaciones nominales del motor



A continuación, se configuran los parámetros de funcionamiento del motor como muestra la Figura 175.

Figura 175

Configuración de los parámetros de funcionamiento del motor



La práctica permite visualizar al variador de velocidad y la maqueta a través de una cámara en tiempo real, entonces, después de oprimir el botón de “Configurar”, los valores configurados se despliegan inmediatamente en el variador, tal y como indica la Figura 176.

Figura 176

Transmisión en tiempo real del funcionamiento del variador y motor a través de una cámara

La Figura 176 es una captura de la imagen que se despliega en el laboratorio remoto, y también muestra como el indicador verde del panel está encendido, señalando que la maqueta está lista para ser utilizada. Cabe indicar, que en el laboratorio de Robótica y PLCs de la Universidad de las Fuerzas Armadas es necesario la presencia de un instructor para preparar los equipos correspondientes.

Posterior a esto, se plantean dos escenarios que permitan probar el funcionamiento de la práctica 1, y que básicamente, se ejecutan las acciones de marcha y paro del motor que se encuentra conectado a la maqueta.

Escenario 1 – Marcha del motor. Una vez que se haya configurado la frecuencia de referencia y los tiempos de aceleración y desaceleración del motor, el primer escenario que se plantea para probar el funcionamiento de esta práctica, es el arranque del motor. Para lo cual, el usuario oprime el botón “Marcha” ubicado en la parte inferior de la interfaz web. Consecuentemente, se despliegan las gráficas en

tiempo real de corriente, voltaje, potencia y velocidad del motor, mostradas en las Figuras Figura 177-Figura 180.

Figura 177

Corriente del motor en arranque

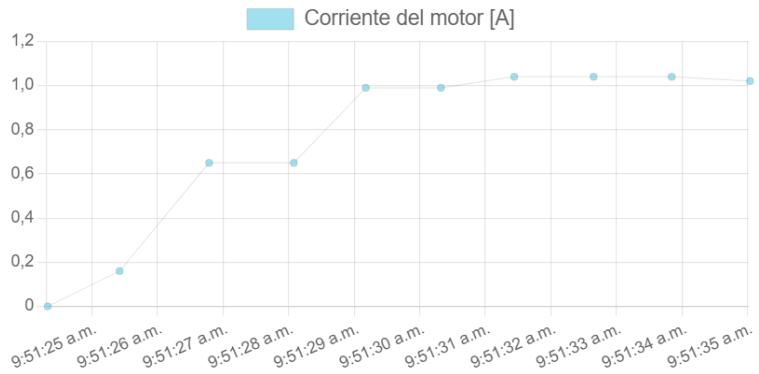


Figura 178

Voltaje del motor en arranque

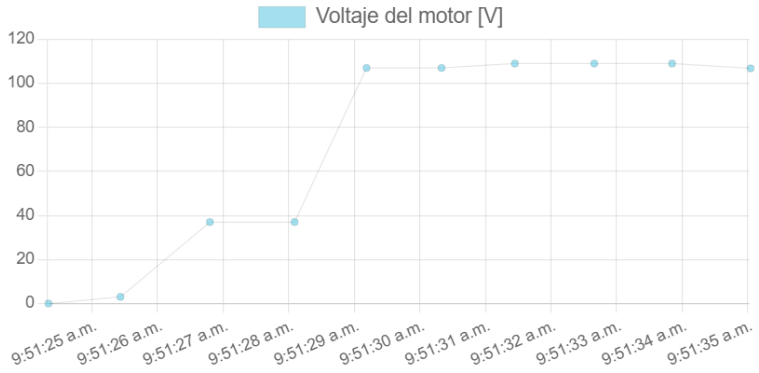


Figura 179

Potencia del motor en arranque

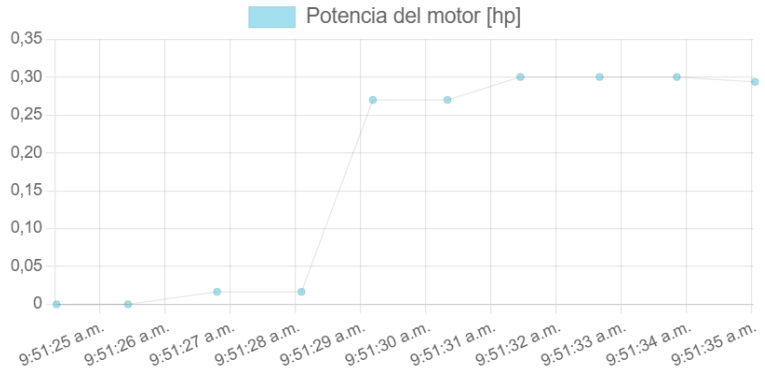
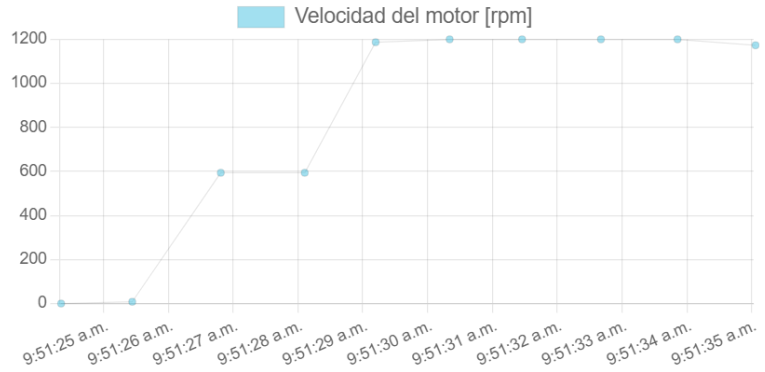


Figura 180

Velocidad del motor en arranque



La prueba de funcionamiento se considera válida, ya que como se puede ver en la Figura 180, la velocidad máxima que toma el motor después de arrancar es de 1200 rpm, equivalente a 40 Hz, que es el valor que se asigna inicialmente en la aplicación web.

Las Figuras Figura 177 y Figura 178 demuestran que tanto la corriente como el voltaje del motor llegan a sus valores máximos después de 5 a 6 segundos. Este valor coincide con el tiempo de aceleración que se configura inicialmente, y es otro aspecto que permite verificar que el funcionamiento de la práctica 1 es el esperado.

Escenario 2 – Paro del motor. En este punto, se detiene el motor oprimiendo el botón “Paro” ubicado también en la parte inferior de la interfaz web. Asimismo, en las Figuras Figura 181-Figura 184 se muestra el funcionamiento del motor cuando se ejecuta la acción de paro.

Figura 181

Corriente del motor en paro

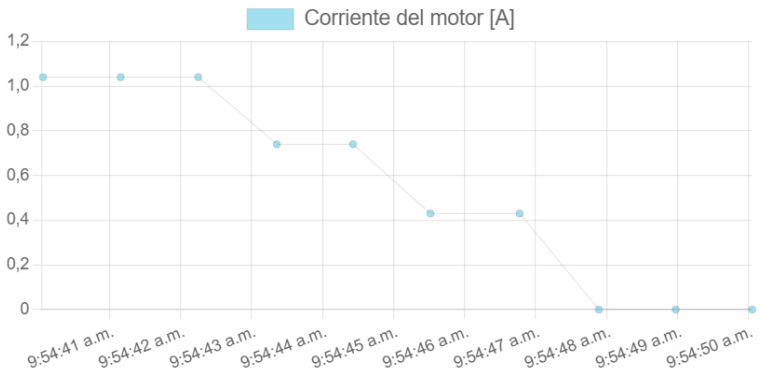


Figura 182

Voltaje del motor en paro

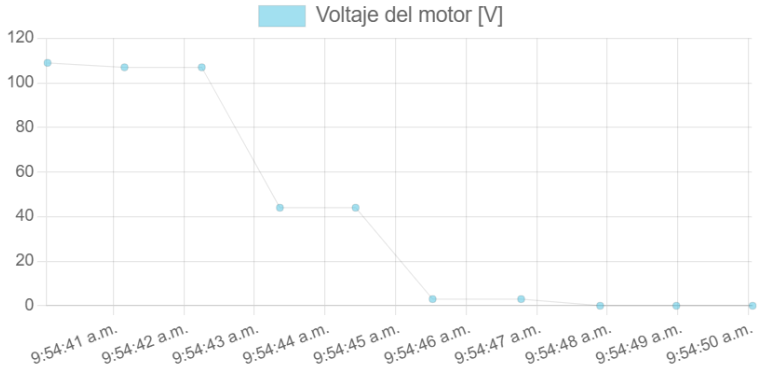


Figura 183

Potencia del motor en paro

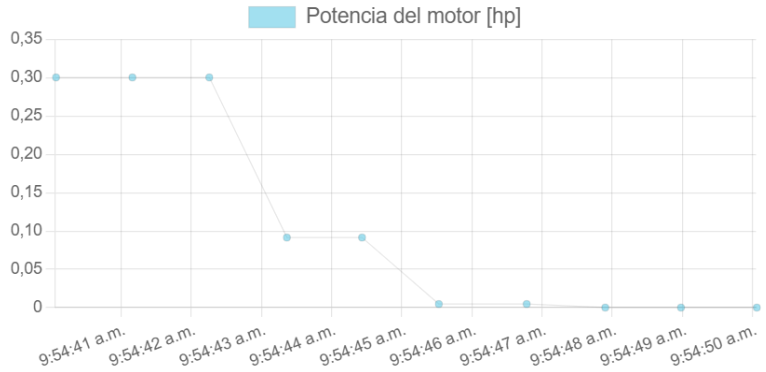
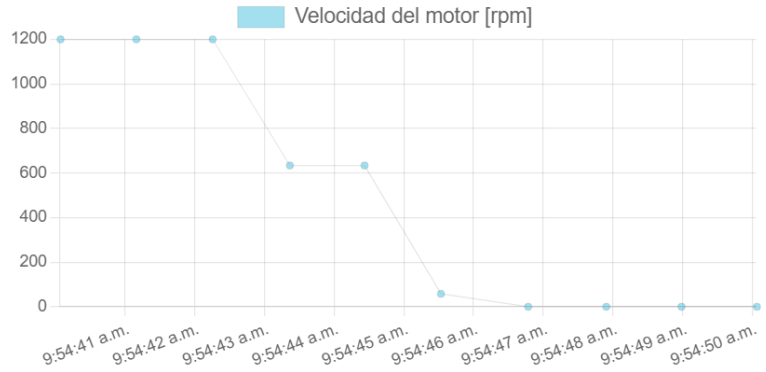


Figura 184

Velocidad del motor en paro



Este escenario también permite determinar que el funcionamiento del laboratorio remoto es correcto, ya que, inicialmente se asigna un tiempo de desaceleración de 6 segundos, y como se puede ver en la Figura 184, el motor retorna a una velocidad de 0 rpm aproximadamente entre 5 y 6 segundos después de ejecutar la acción de paro.

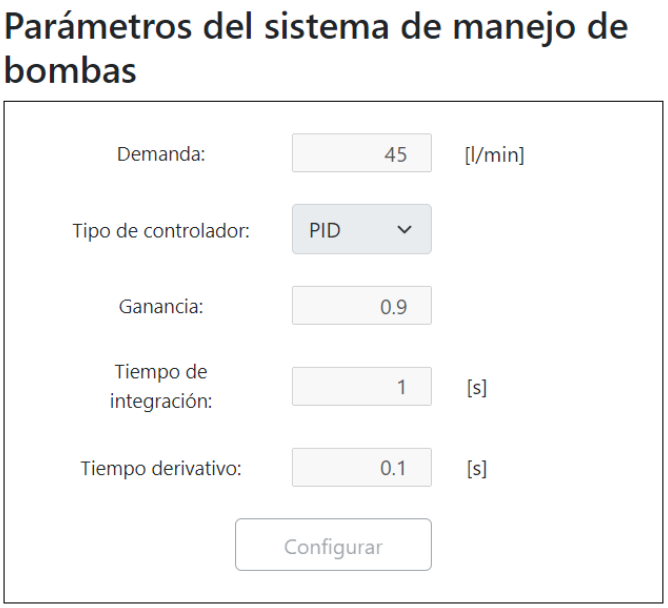
Práctica 2: Sistema con demanda fija

Al ingresar a la pantalla para realizar la práctica 2, se comprueba que las instrucciones y los “Anuncios del Instructor” se reflejen en la interfaz. Después se configura los parámetros del sistema con los valores que se indican en la Figura 185.

Se escoge un controlador PID con ganancia de 0.9, tiempo de integración 1, y tiempo derivativo 0.1. Además, se señala que la demanda deseada es de 45 l/min.

Figura 185

Escenario 1 – Configuración de los parámetros del sistema de control de bombas



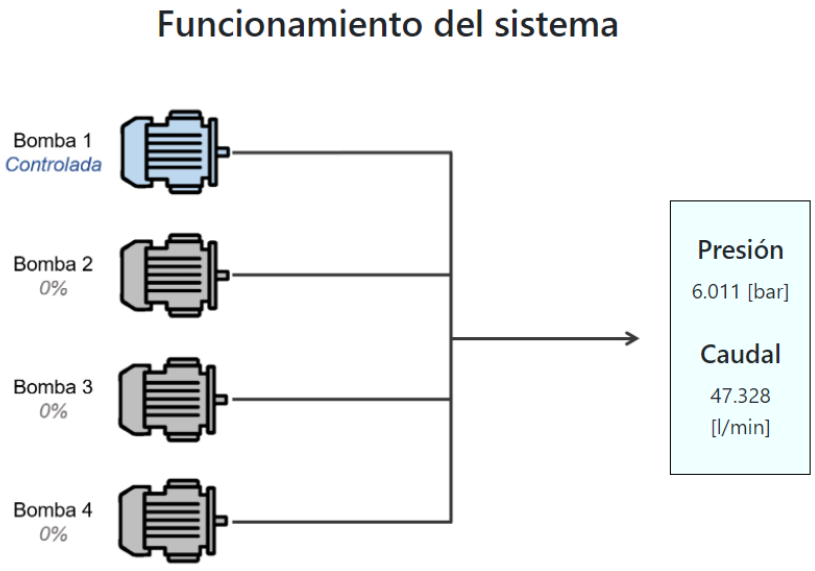
Una vez se da marcha, se puede notar como las variables dentro de las gráficas empiezan a crecer y en el video de la maqueta se observa que la referencia en el variador cambia de cero y el motor arranca. En la Figura 186, se observa que ninguno de los indicadores de la maqueta que representan las bombas que trabajan al 100% se encienden, debido a que la demanda es inferior a los 50 l/min; por lo mismo, en la Figura 187 se señala que la bomba 1 es la controlada por el PID; además, en la parte derecha se observa los valores de presión y caudal una vez que el sistema ya se estabilizó y se alcanzó la referencia de 6 bares.

Figura 186

Transmisión en tiempo real de la maqueta durante el desarrollo de la práctica 2

Figura 187

Funcionamiento del sistema para una demanda de 45 l/min



En las Figuras Figura 188 a Figura 190 se observa la evolución del caudal, la presión y las velocidades de las bombas, respectivamente. Fueron descargadas instantes previos a que la presión llegue a los 6 bares. Al observar las gráficas, se

puede notar que el controlador utilizado hace un seguimiento a referencia un poco lento, pero que el sistema está controlado.

Figura 188

Evolución del caudal del sistema en el escenario 1

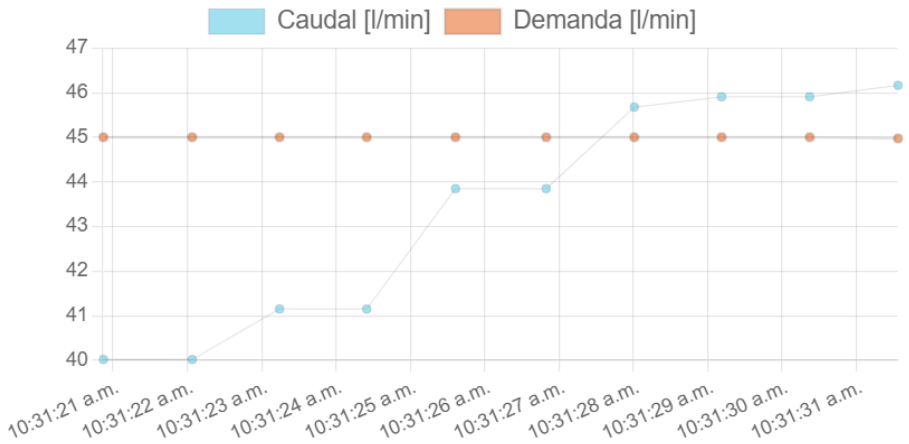


Figura 189

Evolución de la presión del sistema en el escenario 1

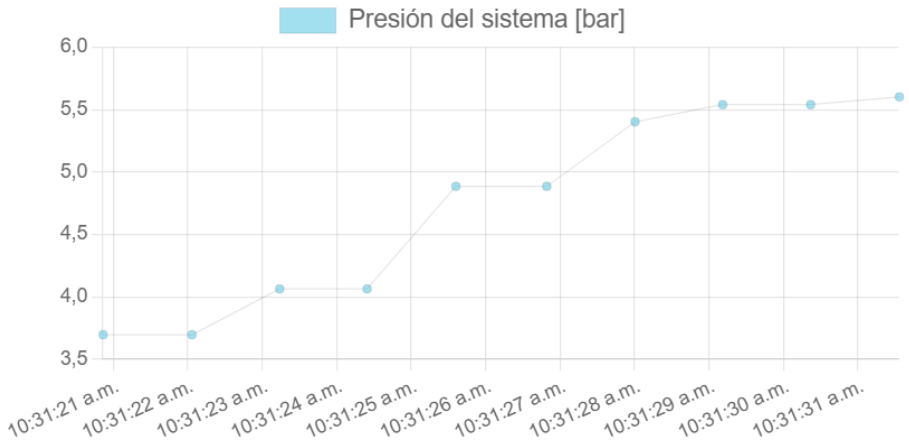
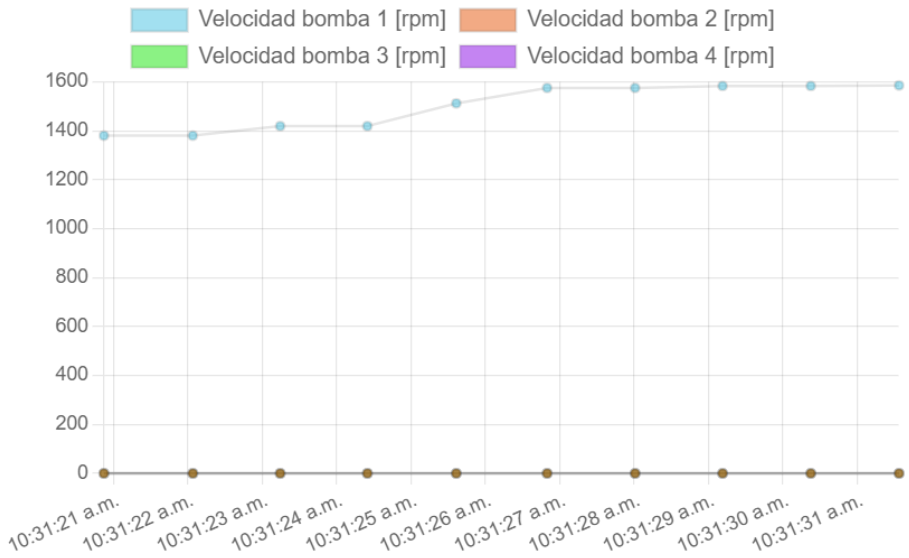


Figura 190

Evolución de las velocidades de las bombas en el escenario 1



En una segunda prueba, se configura al sistema con los parámetros de la Figura 191. En este caso, se utiliza un controlador PI con ganancia de 1 y tiempo de integración de 0.8 s. Además, se requiere una demanda de 175 l/min.

Una vez se da marcha, se observa en la Figura 192 que, en ese caso, tres bombas funcionan a su máxima capacidad y la cuarta bomba será la controlada para ajustar la referencia de 6 bares.

Figura 191

Escenario 2 – Configuración de los parámetros del sistema de control de bombas

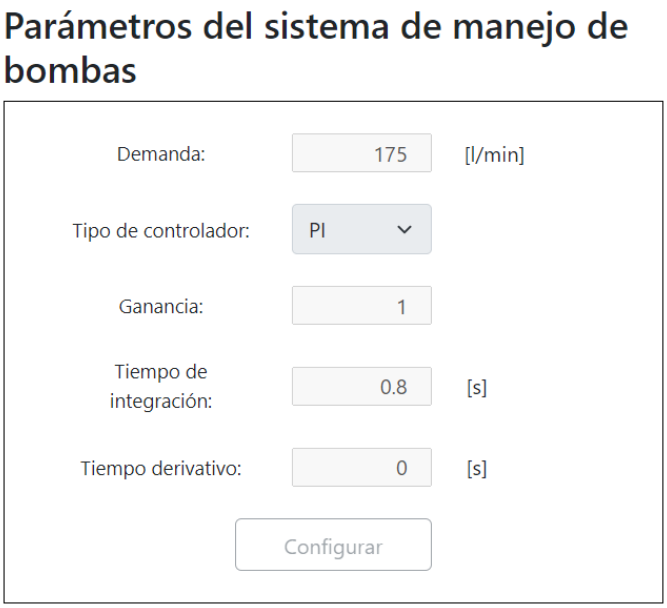
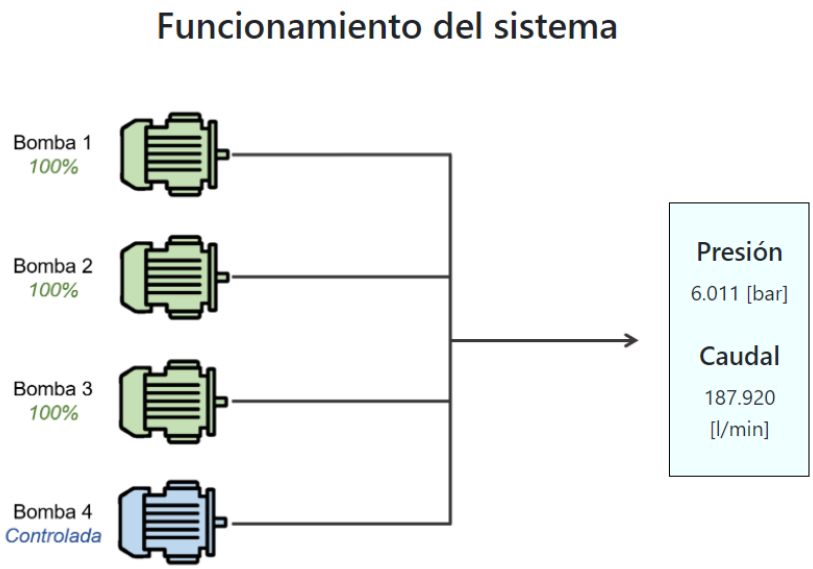


Figura 192

Funcionamiento del sistema para una demanda de 175 l/min



El crecimiento del caudal, presión y velocidades de las bombas se muestran en las Figuras Figura 193 a Figura 195. A diferencia del escenario anterior, se puede notar

claramente que la transición es más rápida, con lo cual, la asignación de los parámetros del controlador en el PLC se realiza correctamente.

Figura 193

Evolución del caudal del sistema en el escenario 2

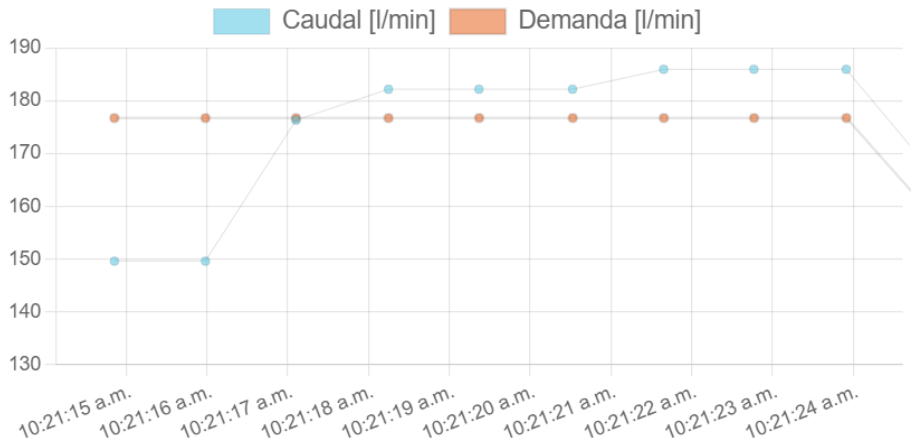


Figura 194

Evolución de la presión del sistema en el escenario 2

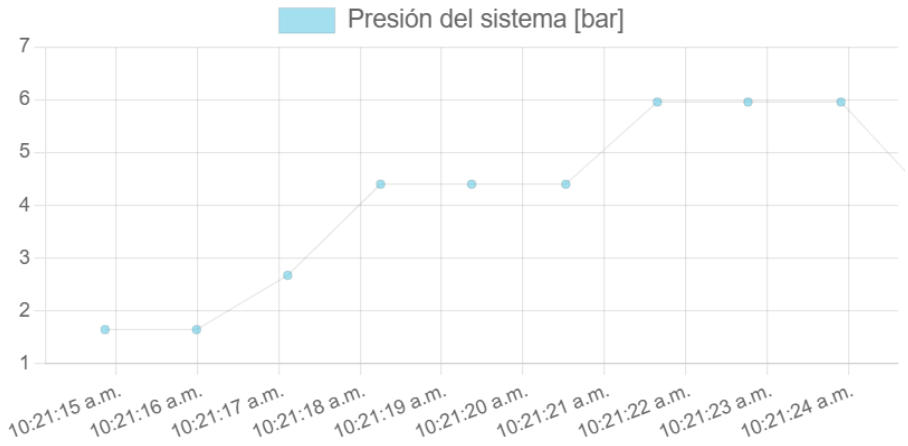
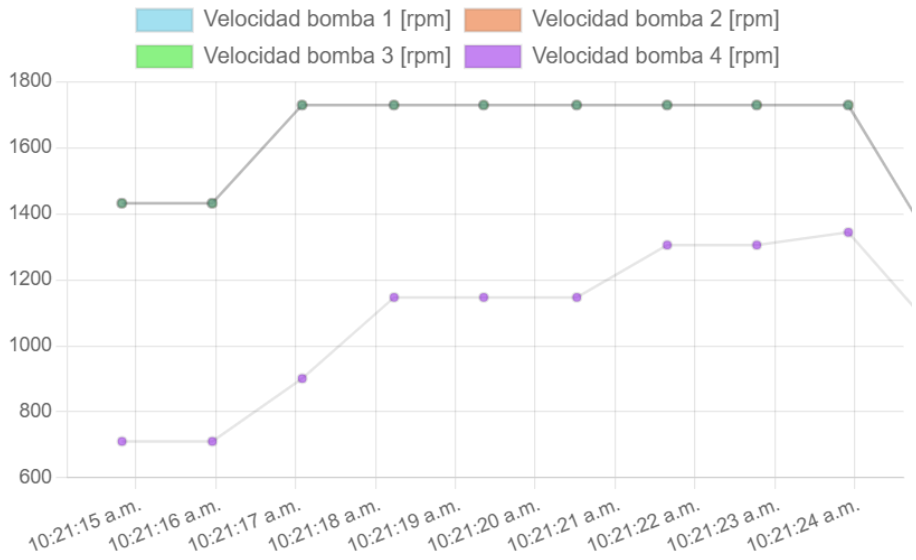


Figura 195

Evolución de las velocidades de las bombas en el escenario 2



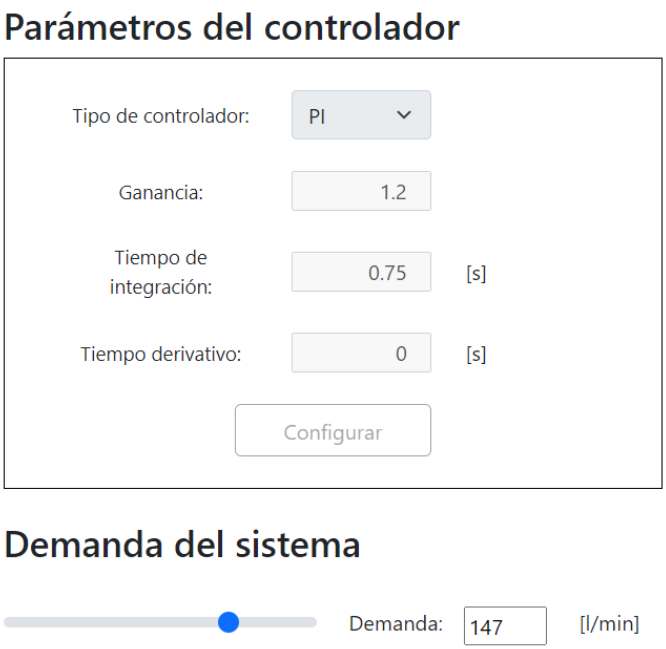
Práctica 3: Sistema con demanda variable

Se plantea un escenario cuyos resultados permitan identificar el comportamiento del sistema ante una modificación en la demanda.

La demanda inicial que se asigna para esta prueba es de 147 l/min. En cuanto al controlador del sistema, se selecciona un PI con constante proporcional de 1.2 y tiempo de integración de 0.75 s. La configuración de estos valores en la aplicación web se muestra en la Figura 196.

Figura 196

Configuración de los parámetros del controlador y demanda inicial del sistema en la práctica 3



Dado que se tiene una demanda de 147 l/min, se requiere de 2 bombas que funcionen al 100% de su capacidad, y de una tercera bomba que permita satisfacer la demanda restante. Sumado a esto, la presión de todo el sistema debe mantenerse en 6 bares independientemente de la demanda que se requiera.

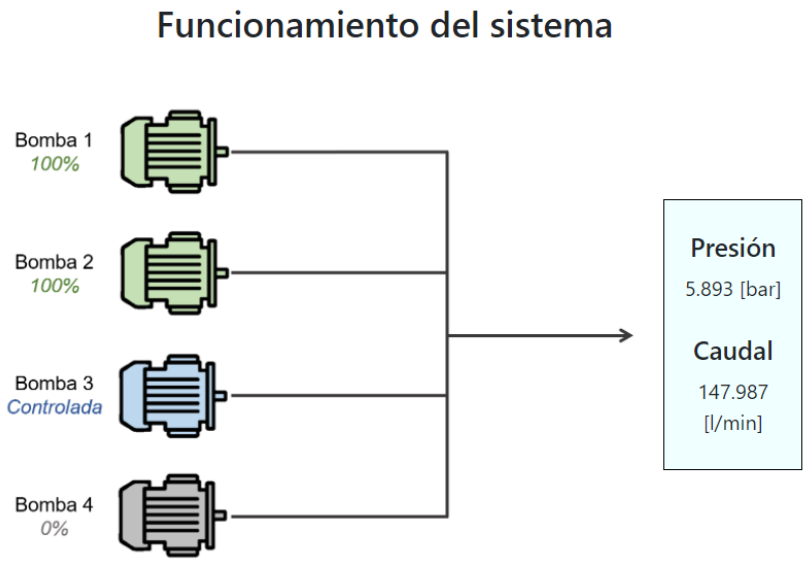
Dicho lo anterior, se da marcha al sistema desde la aplicación web, teniendo como primer resultado el video en tiempo real del funcionamiento de la maqueta. Como se puede ver en la Figura 197, se encuentran encendidos dos indicadores amarillos que representan las 2 bombas que funcionan al 100% de su capacidad. La referencia de frecuencia en el variador empieza a modificarse automáticamente como resultado de la señal de salida del controlador PI programado en el PLC Siemens S7-1200.

Figura 197

Transmisión en tiempo real de la maqueta para una demanda inicial de 147 l/min

Figura 198

Funcionamiento del sistema para una demanda inicial de 147 l/min



El segundo resultado obtenido en la interfaz web se observa en la Figura 198 y permite que el usuario entienda por completo el funcionamiento del sistema, ya que a diferencia de lo mostrado en la Figura 197, indica cual bomba tiene que ser controlada.

La presión del sistema llega a tomar un valor de 5.893 bares, que es muy cercano a 6 (el valor de presión que se requiere) y el caudal llega a ser de 147.987 l/min, que es el valor deseado en un principio.

Las gráficas mostradas en las Figuras Figura 199-Figura 200, indican la evolución de las variables de caudal y presión del sistema. Mientras que la Figura 201 señala las velocidades de las 4 bombas.

Figura 199

Caudal del sistema cuando se requiere una demanda de 147 l/min

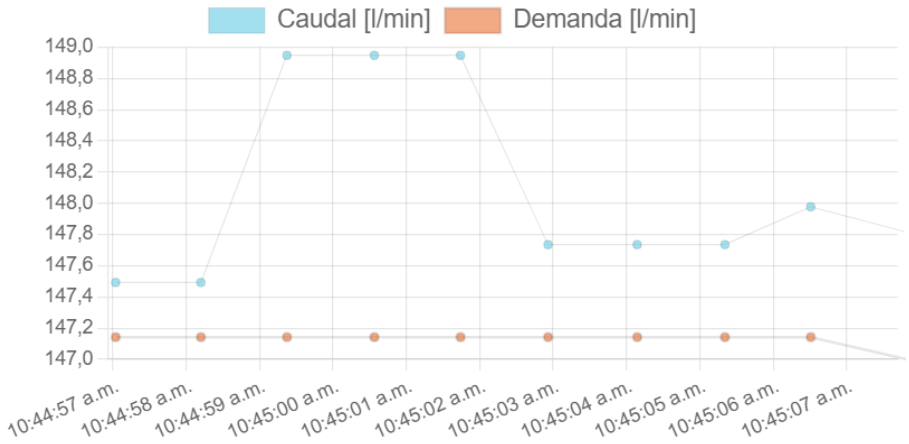


Figura 200

Presión del sistema para una demanda requerida de 147 l/min

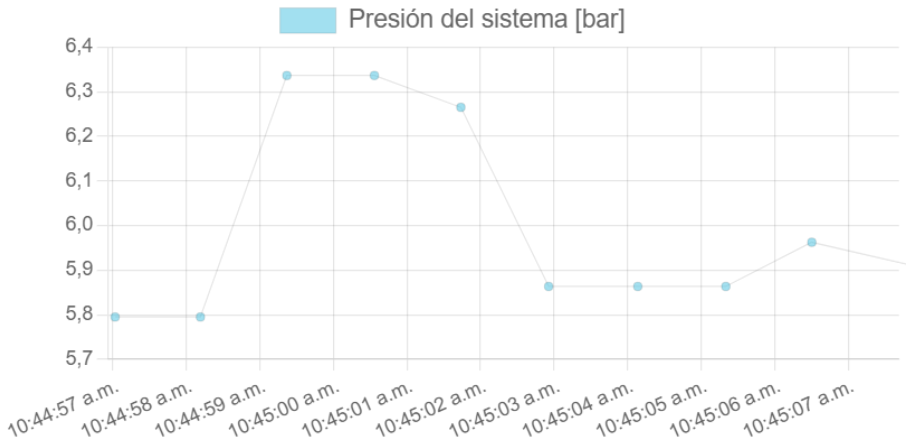
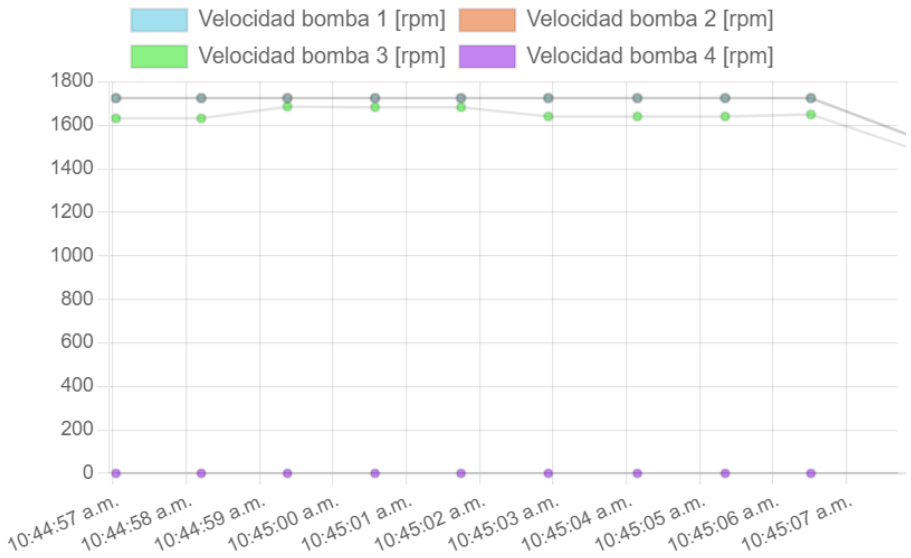


Figura 201

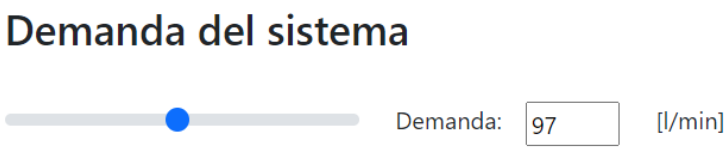
Velocidades de las bombas para una demanda requerida de 147 l/min



Después, se procede a modificar la demanda requerida a un valor de 97 l/min, como muestra la Figura 202. Con esta acción, se pretende evaluar la reacción del controlador ante las variaciones, y el comportamiento tanto del caudal como de la presión del sistema.

Figura 202

Modificación de la demanda en la aplicación web a 97 l/min



La Figura 203 muestra la transmisión en tiempo real de la maqueta, donde se puede ver que ahora el sistema únicamente funciona con una bomba al 100% de su capacidad.

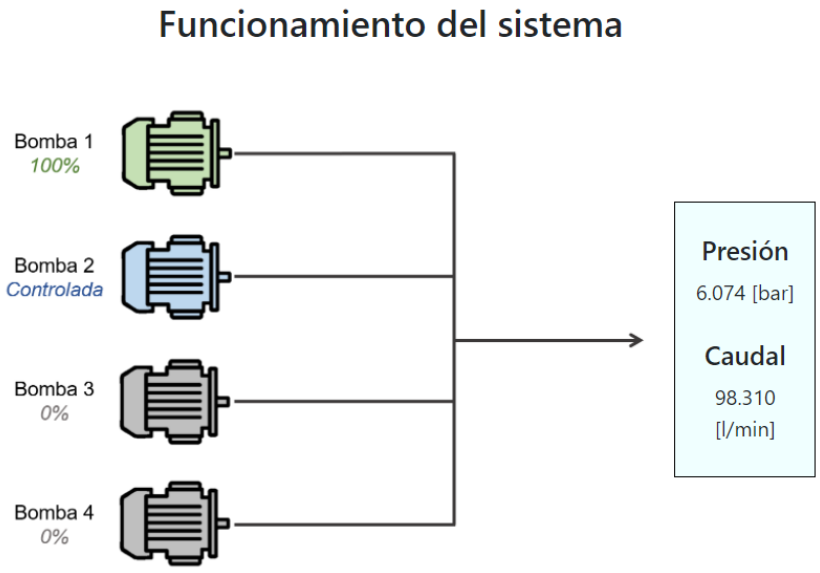
Figura 203

Transmisión en tiempo real de la maqueta para una demanda de 97 l/min

Transcurrido un tiempo, se comprueba que la presión del sistema se mantiene en un valor de 6.074 bares, y el caudal llega a 98.310 l/min, tal y como muestra la Figura 204.

Figura 204

Funcionamiento del sistema cuando la demanda se modifica a 97 l/min



Los indicadores numéricos de la Figura 204 permiten constatar que el sistema funciona correctamente, sin embargo, la visualización de las variables del sistema mediante gráficas temporales, permite analizar de mejor forma el comportamiento del sistema ante la modificación de demanda que se realiza.

La Figura 205 muestra como el caudal del sistema se reduce para poder satisfacer la demanda de 97 l/min. Esta reacción toma un tiempo de 5 segundos, el cual resulta ser óptimo para un sistema de estas condiciones.

Al mismo tiempo, ante la modificación de demanda, el controlador reacciona y, después de 7 segundos aproximadamente, el sistema se mantiene con una presión cercana a los 6 bares, como muestra la Figura 206. Por lo tanto, se puede decir que el sistema efectivamente controla la presión, independientemente de la demanda que se requiera.

Finalmente, como se observa en la Figura 207, la velocidad de la bomba 3 toma un valor de 0 rpm, dado que ahora, únicamente la bomba 1 funciona al 100% de su capacidad, y la bomba 2 es la controlada para satisfacer la demanda deseada.

Figura 205

Caudal del sistema cuando se modifica la demanda a 97 l/min

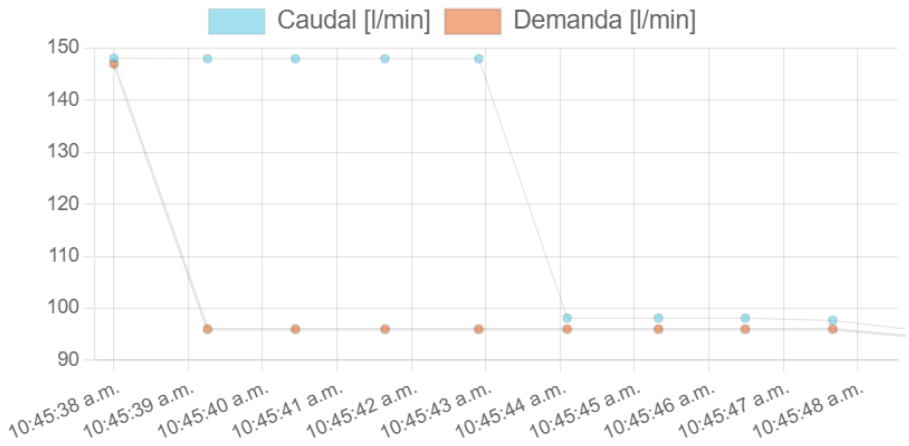


Figura 206

Presión del sistema cuando se modifica la demanda a 97 l/min

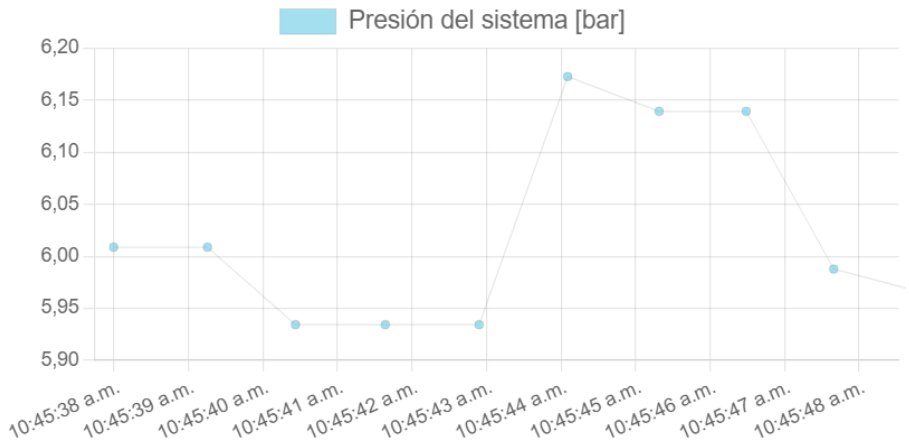
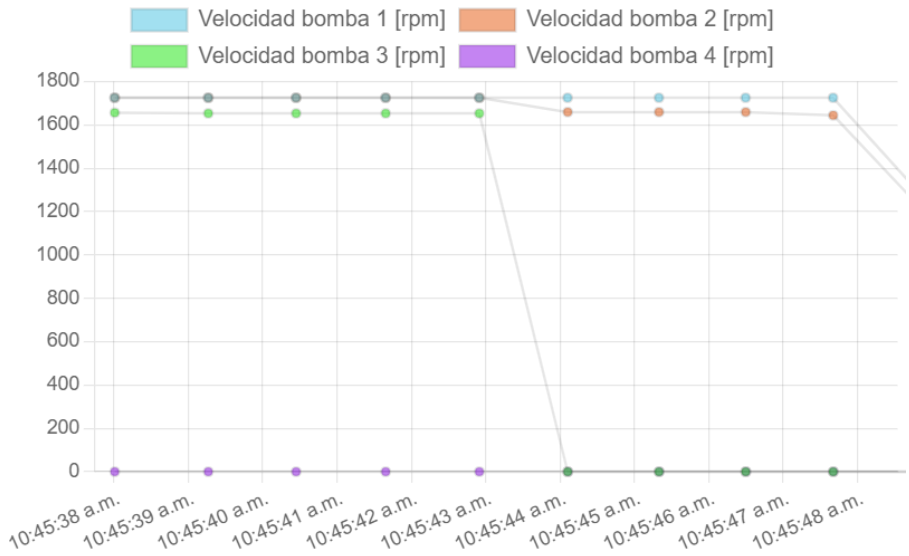


Figura 207

Velocidades de las bombas cuando se modifica la demanda a 97 l/min



Evaluación del laboratorio remoto en términos de aprendizaje

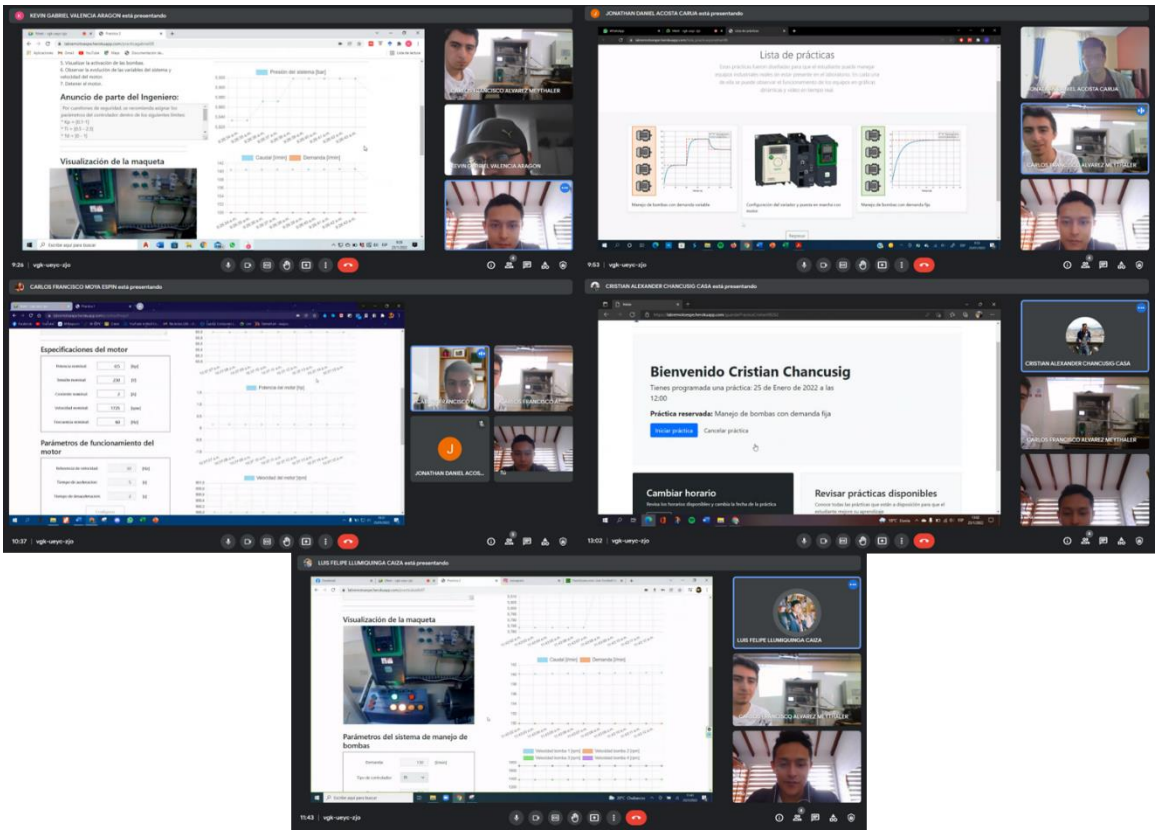
Uno de los principales objetivos de la implementación del laboratorio remoto, es el aprendizaje de los usuarios en cuanto a la configuración de un variador de frecuencia y la sintonización de controladores PI/PID aplicados a un sistema de bombas.

Por esta razón, se selecciona cinco estudiantes de la carrera de Ingeniería Electrónica, Automatización y Control de la Universidad de las Fuerzas Armadas ESPE,

para que pongan a prueba todas las funcionalidades del laboratorio remoto y experimenten la realización de las tres prácticas que se proponen en el presente trabajo.

Figura 208

Estudiantes de Ingeniería Electrónica, Automatización y Control, haciendo uso del laboratorio remoto



Como se muestra en la Figura 208, con cada uno de los estudiantes se realiza una reunión virtual mediante Google Meet para darles una introducción respecto al uso de la aplicación web y las acciones que debe realizar: crear una cuenta, revisar las prácticas disponibles, agendar una práctica en el horario que se haya ingresado y realizar cada una de las prácticas siguiendo las indicaciones mostradas. Luego, cada uno ingresa a la URL para comenzar a probar el laboratorio remoto. Mientras el estudiante desarrolla las prácticas, se responde cualquier duda que tenga.

Una vez que el estudiante termina con el desarrollo de todas las prácticas, responde un formulario con 15 preguntas. Los resultados obtenidos posterior a las pruebas con los cinco estudiantes se muestran en las Figuras Figura 209 a Figura 223.

Figura 209

Resultados de la pregunta 1 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

Figura 210

Resultados de la pregunta 2 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

Figura 211

Resultados de la pregunta 5 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

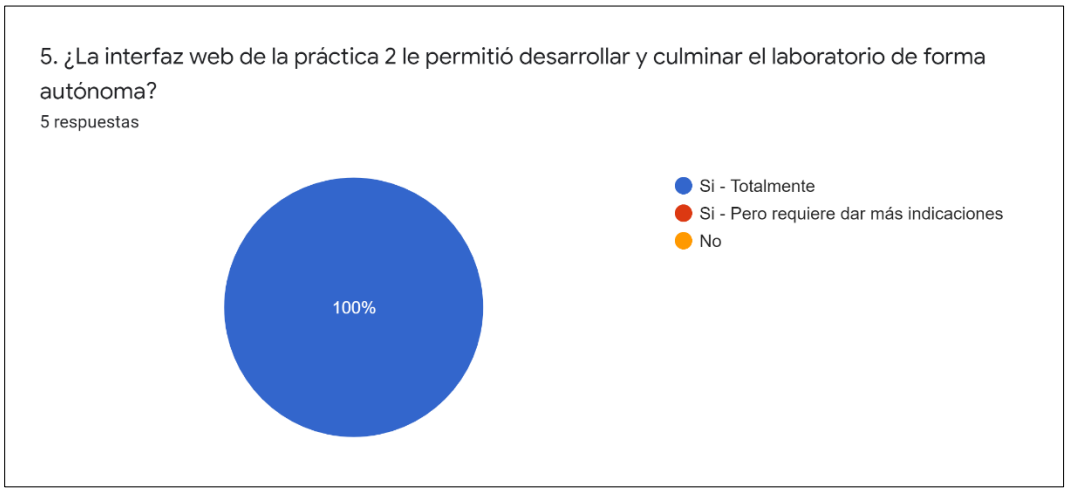


Figura 212

Resultados de la pregunta 8 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

Los resultados de las preguntas 1, 2, 5 y 8, demuestran que la aplicación web fue diseñada de forma apropiada para el manejo de un estudiante, ya que el 100% de los encuestados no tuvo dificultad en el registro de usuario, reserva de horario, ni en el desarrollo de alguna de las prácticas del laboratorio remoto.

Figura 213

Resultados de la pregunta 3 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

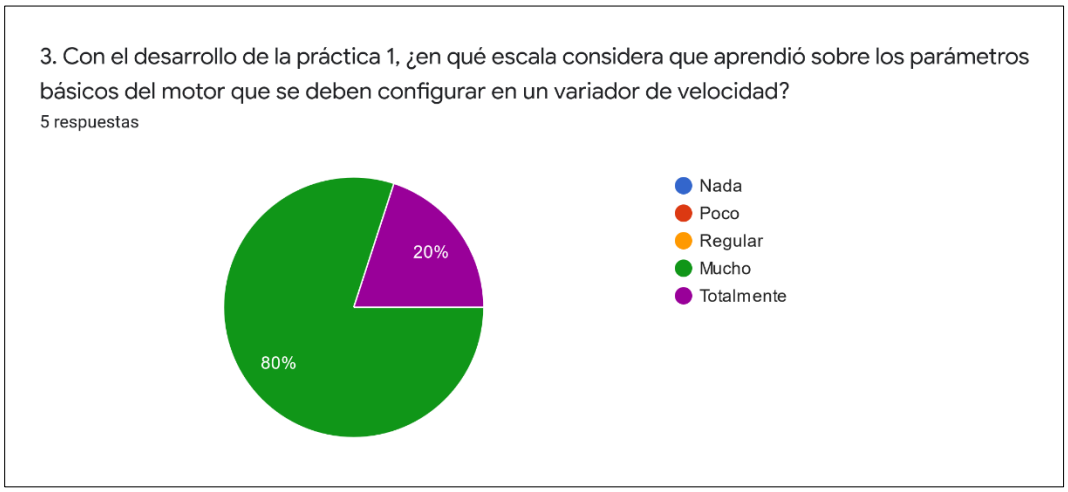
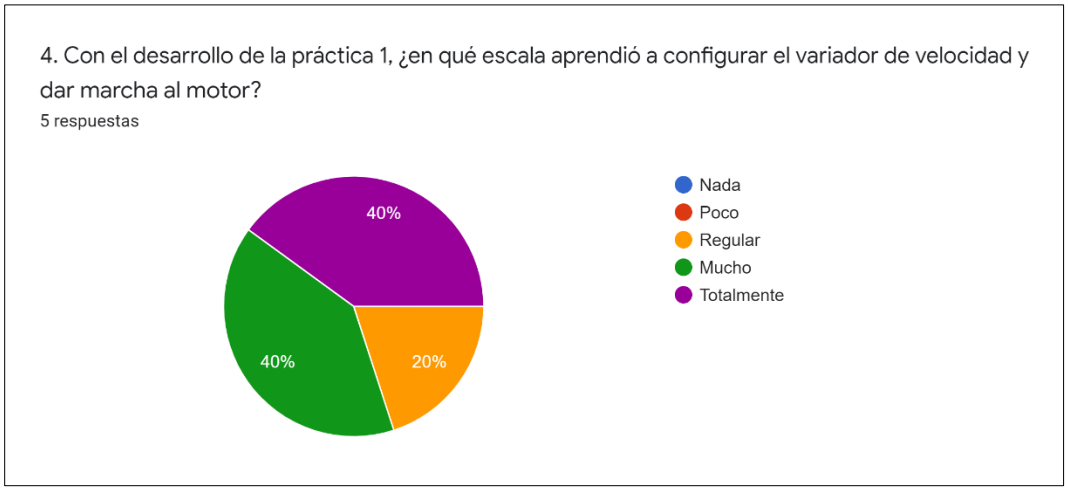


Figura 214

Resultados de la pregunta 4 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



Las preguntas 3 y 4 están enfocadas a conocer si se alcanza los resultados de aprendizaje que se busca con el desarrollo de la práctica 1. En la primera se cuestiona sobre los parámetros básicos del motor que se deben configurar en el variador, obteniendo como resultado que el 80% considera que aprendió mucho, y el 20% captó

totalmente. Por otro lado, en cuanto a la configuración del variador para una puesta en marcha simple, 40% cree que aprendió totalmente, otro 40% aprendió mucho, y el 20% restante considera un aprendizaje regular. De forma general, la práctica aporta al aprendizaje de los dos ámbitos. Los resultados del primero muestran que los estudiantes reconocieron las características del motor que deben ser configurados en el variador antes de poner en marcha; mientras que, en los resultados de la pregunta 4 resalta el 20% (1 estudiante) que no sintió haber aprendido mucho en cuanto a la configuración del variador para una puesta en marcha. En la retroalimentación brindada por el estudiante, expresa que no está explicada la relación entre la referencia de frecuencia y la velocidad del motor deseada.

Figura 215

Resultados de la pregunta 6 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

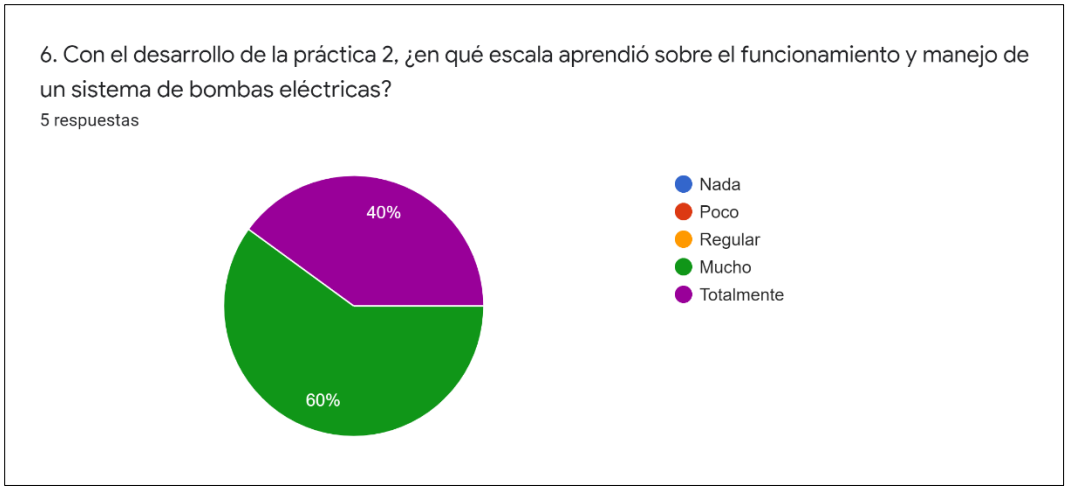
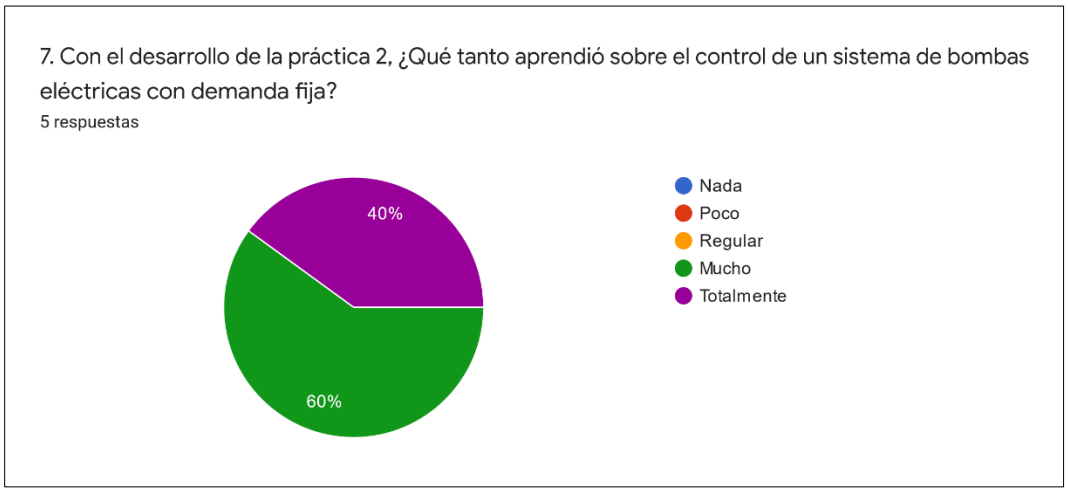


Figura 216

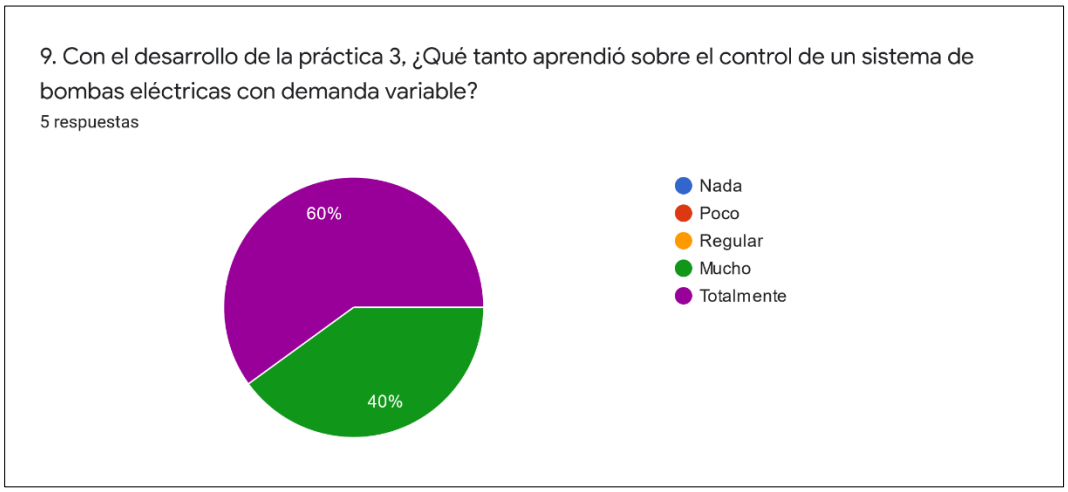
Resultados de la pregunta 7 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



Las preguntas 6 y 7, reflejan que 3 estudiantes aprendieron mucho sobre el control de bombas eléctricas con demanda fija, mientras que los otros 2 sobrantes, aprendieron totalmente. Estos resultados, permiten constatar que la práctica 2 tiene los componentes necesarios para que el estudiante pueda tener un aprendizaje óptimo en cuanto al funcionamiento de un sistema de bombas en el que la presión sea controlada y la demanda requerida sea fija. Es decir, con la realización de esta práctica, se considera que los estudiantes conocen sobre la temática y pueden manejar a la distancia un sistema de estas características.

Figura 217

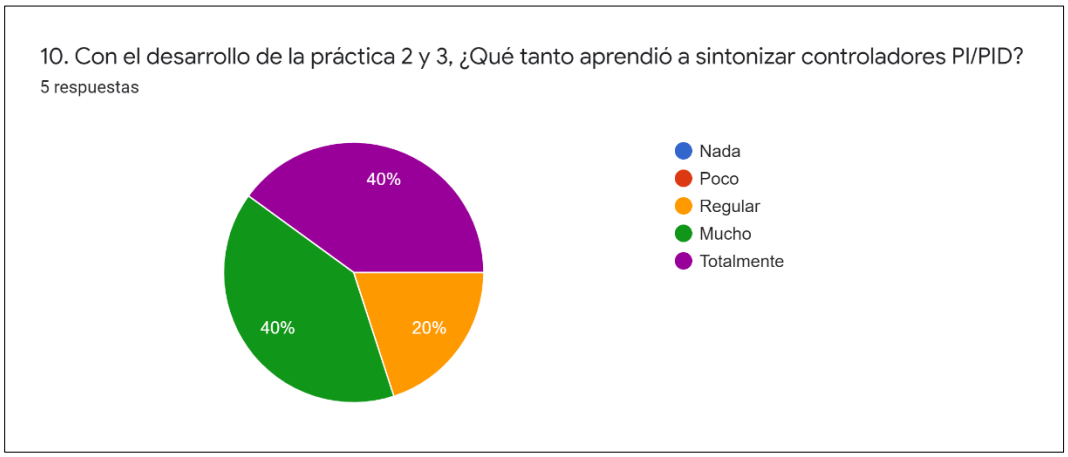
Resultados de la pregunta 9 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



Los resultados de la pregunta 9 (60% totalmente, 40% mucho) demuestran que la práctica 3 permiten al estudiante lograr el aprendizaje sobre el control de un sistema de bombas eléctricas con demanda variable, seguramente apoyada con la experiencia tenida previamente en la práctica 2. Los estudiantes manifestaron que la opción de modificar la demanda permitía determinar mejor el rendimiento del controlador y, con ello, sintonizar parámetros más adecuados para el sistema con demanda variable.

Figura 218

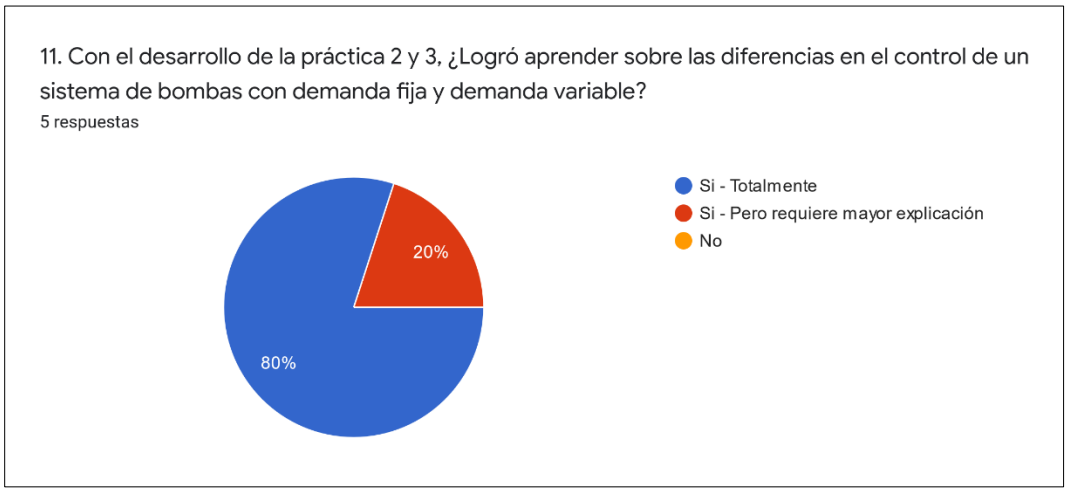
Resultados de la pregunta 10 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



El gráfico resultante de la pregunta 10, indica que 1 estudiante aprendió a sintonizar regularmente los controladores PI/PID, 2 estudiantes aprendieron mucho, y los 2 restantes totalmente. Se puede considerar entonces, que los estudiantes son capaces de sintonizar controladores aplicados a un pequeño sistema de bombas, no obstante, se espera que estos resultados mejoren cuando los estudiantes utilicen más tiempo el laboratorio remoto, debido a que el aprendizaje en sintonización de controladores depende de la puesta en marcha del sistema con diferentes parámetros de ajuste.

Figura 219

Resultados de la pregunta 11 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



En la pregunta 11 se cuestiona respecto a si el estudiante reconoció las diferencias en el control del sistema con demanda fija y demanda variable. El 80% aseguró que si, mientras que el 20% (1 estudiante) necesitó mayor explicación para notar las diferencias. Con esta pregunta, se estudia la capacidad de autoaprendizaje que se puede tener con el laboratorio remoto, a pesar de que con la mayoría habría sido efectiva, es necesario mantener un canal de comunicación con el instructor para aclarar esta y otras dudas que tengan los estudiantes.

Figura 220

Resultados de la pregunta 12 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

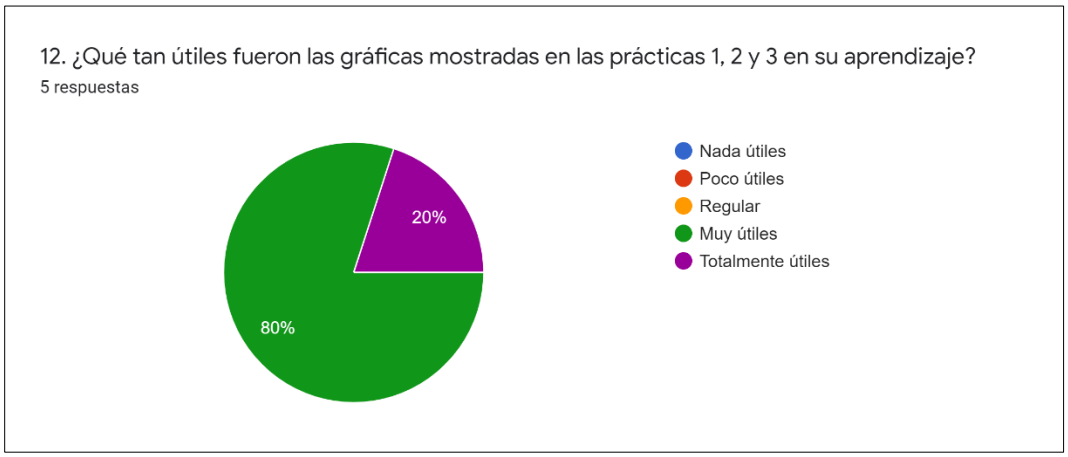
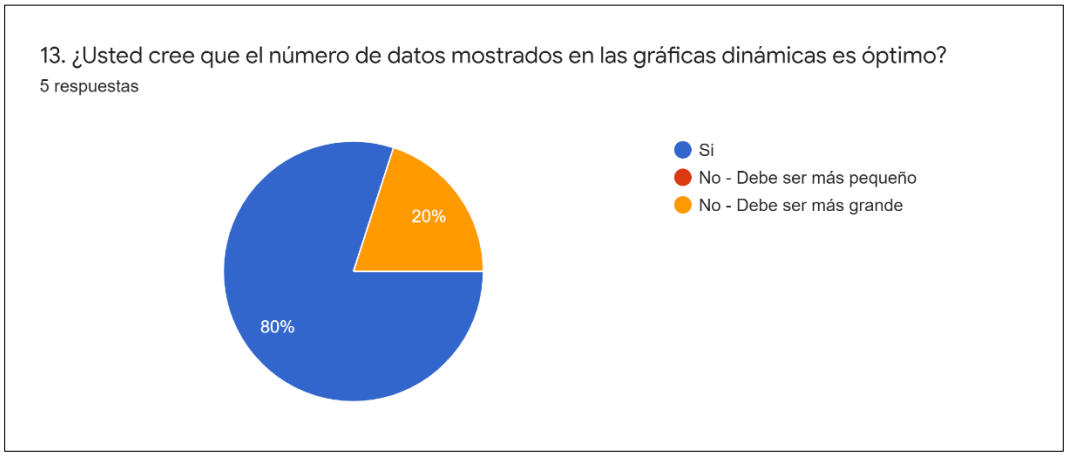


Figura 221

Resultados de la pregunta 13 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



Las respuestas a la pregunta 12 demuestran que la inclusión de gráficas de las variables de funcionamiento, tanto del variador de frecuencia como del sistema de bombas, facilita al estudiante en cuanto al aprendizaje de la práctica, ya que al mostrar datos en tiempo real expresados gráficamente, se puede visualizar el comportamiento del sistema ante la configuración de un determinado controlador con sus respectivos

parámetros de ajuste. Sumado a esto, las gráficas que se muestran en el laboratorio remoto son dinámicas, y acorde a lo respondido en la pregunta 13, 4 de los 5 encuestados concuerdan que el número de datos mostrados en las gráficas son suficientes para entender el comportamiento del motor o sistema de bombas; no obstante, 1 encuestado considera que se requiere mostrar más datos. Cabe indicar que, la cantidad de datos a mostrar en las gráficas es un valor que se puede modificar en el desarrollo de la aplicación, y ciertamente con este cambio la experiencia de laboratorio remoto mejoraría el entendimiento de las prácticas por parte de los usuarios.

Figura 222

Resultados de la pregunta 14 del cuestionario realizado a los estudiantes que desarrollaron las prácticas

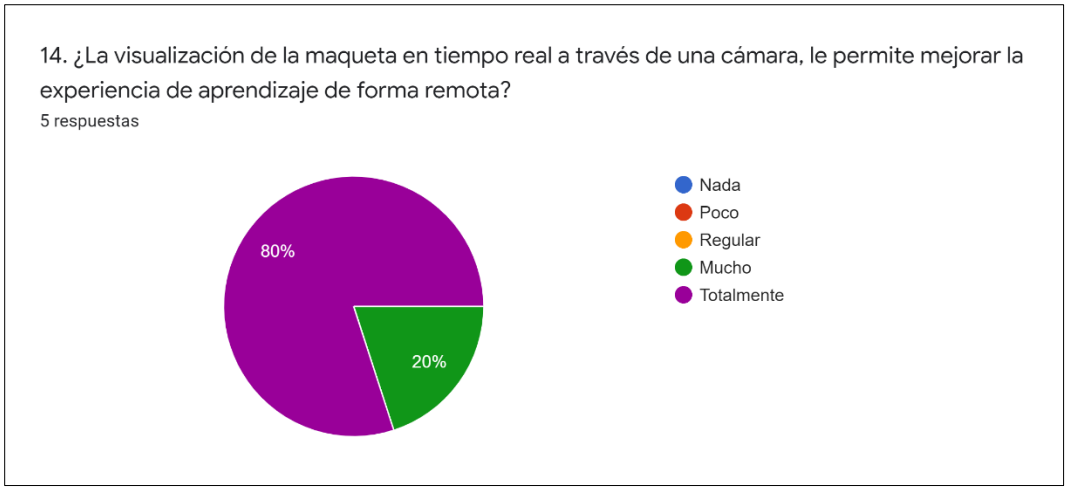
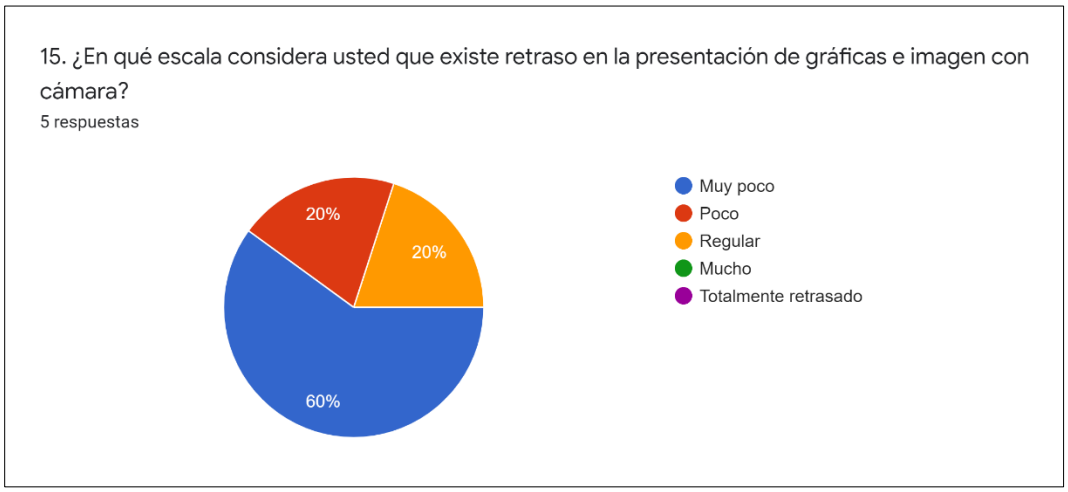


Figura 223

Resultados de la pregunta 15 del cuestionario realizado a los estudiantes que desarrollaron las prácticas



Finalmente, las preguntas 14 y 15 están relacionadas con la transmisión de imagen de la maqueta en la aplicación web. En la primera se cuestiona si la capacidad de visualizar la maqueta durante la práctica mejora la experiencia de aprendizaje. De acuerdo a los resultados (80% totalmente, 20% mucho), este factor si representaría una mejora en la experiencia de aprendizaje con el laboratorio remoto. En cuanto a la segunda pregunta, se busca determinar si existió algún retraso en la presentación de los datos en las gráficas o en la transmisión de video. La mayoría (80%) considera que fue poco o muy poco el retraso, mientras que el 20% cree que el retraso fue regular. Este último considera que el tiempo de muestreo en las gráficas podría ser más corto, sin embargo, dado que la transmisión de datos de la maqueta hacia el servidor siempre es mayor a un segundo, reducir el tiempo de actualización en las gráficas no mejoraría considerablemente la visualización de los resultados.

Capítulo VI. Conclusiones y Recomendaciones

Conclusiones

En este trabajo se desarrolla un laboratorio remoto para que los estudiantes de la carrera de Ingeniería en Electrónica, Automatización y Control, y futuras generaciones, aprendan sobre el manejo de un variador de velocidad y la sintonización de controladores PID aplicados a un sistema de bombeo, mediante el manejo de equipos industriales reales. El laboratorio remoto cuenta con tres prácticas: configuración del VFD y puesta en marcha del motor, control de bombas con demanda fija y control de bombas con demanda variable.

Para el desarrollo de las tres prácticas se elabora una maqueta en el laboratorio de Robótica y PLCs que cuenta con los siguientes equipos: PLC Siemens S7-1200, variador de frecuencia Schneider Electric Altivar ATV630U15M3, panel de operador Schneider Electric Magelis HMI STU855 y motor Baldor CM3538 1/2 hp. Además, dispone de una fuente de alimentación de 24 V DC, switch industrial con 5 puertos, y botones e indicadores industriales. El PLC contiene toda la programación para simular el sistema de bombeo y manejar el variador en el desarrollo de cada una de las prácticas, mientras que el panel de operador controla el funcionamiento de manera local.

El enlace de comunicación entre el variador de frecuencia y el PLC se establece mediante el protocolo MODBUS TCP/IP, siendo este último el que se comporta como cliente para leer y escribir en los registros del variador. Mientras que, la comunicación entre el PLC y el panel de operador se realiza a través de comunicación S7, por medio de la interfaz Ethernet.

Se desarrolla la aplicación mediante Django que, para el presente trabajo, permite la integración de Python, HTML, JavaScript y CSS como principales lenguajes de programación. Además, se utiliza la plataforma de servicios Heroku como tecnología

de software para el almacenamiento de la aplicación en la nube y que, en definitiva, permite el acceso al laboratorio remoto desde cualquier lugar en donde se encuentre el usuario.

El enlace de comunicación para el acceso remoto se realiza a través de la tarjeta Raspberry PI 4. Fueron tres las directrices que se consideraron para el diseño, y que se implementaron mediante funcionalidades de software, tales como: SNAP 7, para la adquisición y envío de datos al PLC; peticiones GET y POST, para el acceso y envío de datos a la aplicación web, y WebRTC para la transmisión de video en las interfaces. La integración de todos estos recursos, permitieron que el enlace remoto garantice un flujo de datos óptimo y continuo desde la maqueta hacia la aplicación web, y viceversa.

El 100% de las pruebas realizadas enfocadas en la transmisión de datos, demuestran que la conectividad del laboratorio remoto, es totalmente óptima; dado que, si existe alguna modificación de parámetros desde la aplicación web, el enlace remoto accede a esta información y establece conexión con el PLC, que es el componente que maneja todo el funcionamiento de la maqueta. Adicional a esto, el 95% de los ciclos de envío y recepción de datos de la maqueta al servidor de la aplicación web, oscila entre 1.8 y 2.286 segundos, lo que indica que, la conectividad además de ser continua, es rápida; ya que son tiempos relativamente pequeños y no repercuten en el desarrollo de las prácticas de laboratorio por parte de los usuarios. Estos dos parámetros, permiten concluir que el laboratorio remoto en términos de conectividad es fiable.

Acorde a la evaluación que se realizó a los estudiantes de la carrera de Ingeniería Electrónica, Automatización y Control, se determinó que el 100% de las personas encuestadas pudieron realizar el registro, autenticación, agendamiento y desarrollo de las prácticas de forma autónoma; lo que quiere decir, que el diseño de todas las interfaces de la aplicación web fueron bien estructuradas y diseñadas para

que el usuario pueda ejecutarlas de forma remota y sin la necesidad de acudir a un laboratorio de forma presencial.

En relación a la enseñanza, los resultados de las encuestas indican que los estudiantes alcanzan los objetivos de aprendizaje de cada una de las prácticas, aunque en las prácticas 2 y 3 se debería disponer de mayor tiempo para ensayar la sintonización de controladores PID, y tener un medio de comunicación con el instructor durante la práctica para solventar las dudas que tenga el estudiante.

La visualización de gráficas en tiempo real, y la transmisión de video de la maqueta son dos recursos que los estudiantes consideran de gran utilidad para el desarrollo de las prácticas, dado que reflejan los resultados de manera inmediata y mejora la experiencia de manejo de equipos industriales reales. Por otro lado, ampliar el tiempo que se muestran los datos en las gráficas, mejoraría el reconocimiento de resultados de las prácticas.

Las ventajas del laboratorio remoto en términos de implementación radican en su desarrollado con lenguajes de programación libres (Python, JavaScript, HTML) y recursos en la nube gratuitos (Heroku – Redis, Postgres) que, a su vez, permiten alcanzar tiempo de respuesta cortos y fiabilidad de transmisión. Además, evita la necesidad de instalar alguna aplicación o extensión que pueda ser incompatible con el dispositivo o sistema operativo que dispone el estudiante; siendo los únicos requisitos, tener cualquier navegador instalado y contar con conexión a internet.

Recomendaciones

Antes del inicio de cada práctica, se recomienda que el instructor que se encuentre en el laboratorio de Robótica y PLC de la Universidad de las Fuerzas Armadas ESPE matriz Sangolquí, configure la maqueta a las condiciones iniciales para evitar posibles daños en sus componentes.

Reiniciar la ejecución del *script* de Python en la Raspberry Pi 4, con el objetivo de refrescar la página web para transmitir el video de la webcam, de esa forma se evita que pueda mantenerse el enlace punto a punto con algún ordenador, y produzca problemas en la aparición de la imagen en la aplicación web.

Los estudiantes deberían revisar las especificaciones técnicas del variador de frecuencia y del motor trifásico implementados en la maqueta, previo a la utilización del laboratorio remoto. De esta manera, se previene que se configuren parámetros fuera de los límites normales de funcionamiento de cada componente y se evitan posibles daños.

Trabajos futuros

Adaptar más equipos industriales a la maqueta, e idear más prácticas para el laboratorio remoto, a fin de crear un recurso de enseñanza que permita una mejor formación en los estudiantes de la carrera de Ingeniería en Electrónica, Automatización y Control, y futuras generaciones.

Añadir la transmisión de sonido en tiempo real de la maqueta hacia la aplicación web, para mejorar la experiencia del estudiante con el manejo de equipos industriales.

Referencias

3CX. (2021). *¿Qué es WebRTC?* https://www.3cx.es/voip-sip/que-es-webrtc/

ABB. (2021). *Qué es un variador de frecuencia: Definición, cómo funciona, características y ventajas*. https://new.abb.com/drives/es/que-es-un-variador

Abrams, J. (26 de mayo de 2020). *Los laboratorios remotos revolucionan el aprendizaje desde casa*. https://theconversation.com/los-laboratorios-remotos-revolucionan-el-aprendizaje-desde-casa-137101

Alvarado, D., & Sánchez, J. (2011). Laboratorio remoto para prácticas virtuales de automatización con el PLC SIMATIC S7-1200. Cuenca, Ecuador: Universidad del Azuay.

Arbeláez, O., Medina, F., & Chaves, J. (2011). Herramientas para el desarrollo rápido de aplicaciones web. *Scientia et Technica Año XVII, 1*(47), 254-258. https://doi.org/10.22517/23447214.511

Arora, N. (29 de diciembre de 2019). *Django Models*. GeekforGeeks: https://www.geeksforgeeks.org/django-models/

Arrieta, R., & Piedra, O. (2015). Laboratorio virtual y remoto para el desarrollo de prácticas de Control de Procesos. Ecuador, Sangolquí: Repositorio Universidad de las Fuerzas Armadas.

Atlassian Bitbucket. (2022). *Qué es Git*. https://www.atlassian.com/es/git/tutorials/what-is-git

Aula21. (2020). *Qué es un HMI: para qué sirve la Interfaz Hombre-Máquina*. https://www.cursosaula21.com/que-es-un-hmi/

Autycom. (2 de octubre de 2019). *¿Qué es un sistema HMI?* https://www.autycom.com/que-es-un-sistema-hmi/

Ayllon, N. (10 de febrero de 2021). *PROFINET Explained*. PI North America: https://us.profinet.com/profinet-explained/

Bailey, C. (2022). *Making HTTP Requests With Python*. https://realpython.com/courses/python-requests/

Belloso, R. (2021). Capítulo II. Marco Teórico. http://virtual.urbe.edu/tesispub/0107218/cap02.pdf

Bocancho Sanchez, I. E. (2021). *Camara Web Agiler AGI 4188*. https://www.ecuatech.com/shop/product/cwagiler4188-agiler-agi4188-camara-web-agiler-agi-4188-1080p-full-hd-microfono-incluido-1920x1080-usb-2-0-ideal-clases-en-linea-y-zoom-8045?category=31

Borges, E. (17 de marzo de 2019). *Servidor Base de Datos*. https://blog.infranetworking.com/servidor-base-de-datos/

Carrera, V. (11 de abril de 2017). *Una nueva herramienta al alcance de los estudiantes de la MAD*. Universidad Técnica Particular de Loja: https://noticias.utpl.edu.ec/una-nueva-herramienta-al-alcance-de-los-estudiantes-de-la-mad

Casado, C. (2021). *Las aplicaciones web y las bases de datos*. http://multimedia.uoc.edu/blogs/fem/es/las-aplicaciones-web-y-las-bases-de-datos/

Circuit.pk. (2021). *110v Ac Indicator Light AD1622DS White Colour*. http://circuit.pk/product/3018

Cloudflare. (2021). *What do client side and server side mean?* | *Client side vs. server side*. https://www.cloudflare.com/es-es/learning/serverless/glossary/client-side-vs-server-side/

CLR. (s.f.). *Motores monofásicos, bifásicos y trifásicos: todo lo que necesitas saber*. https://clr.es/blog/es/motores-monofasicos-bifasicos-trifasicos/

Cruz, F. D., Díaz-Granados, M., Zerpa, S., & Giménez, D. (2010). Web-LABAI: Laboratorio Remoto de Automatización Industrial. *Revista Iberoamericana ded Automática e Informática Industrial, 7*(1), 101-106. 10.4995/RIAI.2010.01.10

Daley, B. (26 de mayo de 2020). *Los laboratorios remotos revolucionan el aprendizaje desde casa*. The Conversation: https://theconversation.com/los-laboratorios-remotos-revolucionan-el-aprendizaje-desde-casa-137101

Electrónica I+D. (2021). *Suiche pulsador NC 22mm Rojo*. https://www.didacticaselectronicas.com/index.php/suiches-y-conectores/suiches/pulsadores/suiche-pulsador-nc-22mm-rojo-suiches-switches-botones-pulsadores-moment%C3%A1neos-pulsadores-industriales-normalmente-cerrados-de-22mm-rojo-detail

Gargil, S.L. (26 de agosto de 2021). *Cálculo de la potencia de una bomba hidraulica*. https://gargil.es/calculo-de-la-potencia-de-una-bomba-hidraulica/

Gavilema Casa, S., & Yánez Garzón, V. (2020). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO EN CONTROL DE MOVIMIENTO PARA EL DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE.* Sangolquí: Universidad de Las Fuerzas Armadas ESPE.

Gómez, P. (noviembre de 2003). *DeviceNet, un Bus de Campo maduro y consolidado en el mercado de automatización*. Editora Microbyte: http://www.emb.cl/electroindustria/articulo.mvc?xid=117&ni=devicenet-un-bus-de-campo-maduro-y-consolidado-en-el-mercado-de-automatizacion

Hidráulica Hidraoil. (2021). *Dimensionamiento de la bomba hidráulica en función del cilindro hidráulico*. https://hidraulicahidraoil.es/articulos/dimensionamiento-de-la-bomba-hidraulica-en-funcion-del-cilindro-hidraulico/

Ibanez, L. (21 de abril de 2018). *Development: Difference between the http requests POST and GET*. https://lazaroibanez.com/difference-between-the-http-requests-post-and-get-3b4ed40164c1

ifm electronic. (2021). *Principios de medición de los sensores de caudal y caudalímetros de ifm*. https://www.ifm.com/es/es/shared/technologies/stromungssensoren/technologie/principios-de-medicion-de-los-sensores-de-caudal-y-caudalimetros-de-ifm

Jiménez, L., Puerto, R., Reinoso, R., Fernández, Ó., & Ñeco, R. (2010). Recolab: Laboratorio Remoto de Control Utilizando Matlab y Simulink. *Revista Iberoamericana de Automática e Informática industrial, 2*(2), 64-72.

Llordachs, F. (2021). *¿Qué son los servicios en la nube? Tipos y ejemplos*. https://clinic-cloud.com/blog/servicios-en-la-nube-tipos-ejemplos/

Logicbus. (s.f.). *Switches Ethernet Industriales*. https://www.logicbus.com.mx/switch-ethernet-industrial.php

López-Ortiz, S., Balseca-Sampedro, O., Moreno-Romero, T., Bravo-López, E., & Pino-Pilco, P. (2020). Diseño y evaluación de un laboratorio remoto para la enseñanza de diseño de circuitos electrohidráulicos. *Revista Científica Dominio de las Ciencias, 6*(3), 399-424. http://dx.doi.org/10.23857/dc.v6i3.1291

Mar-Cornelio, O., Santana-Ching, I., & González-Gulín, J. (2019). Sistema de Laboratorios Remotos para la práctica de Ingeniería de Control. *Revista científica, 36*(3), 356-366. https://doi.org/10.14483/23448350.14893

MDN Web Docs. (2022). *Tutorial Django Parte 2: Creación del esqueleto del sitio web*. https://developer.mozilla.org/es/docs/Learn/Server-side/Django/skeleton_website

MDN Web Docs. (s.f.). *Introducción a Django*. https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

Miru, G. (14 de marzo de 2017). *The Siemens S7 Communication - Part 1 General Structure*. http://gmiru.com/article/s7comm/

MODICON Inc. (1996). *Modicon Modbus Protocol Reference Guide.* North Andover.

Mostia, W. (04 de enero de 2019). *Introduction to Modbus*. https://www.controlglobal.com/articles/2019/introduction-to-modbus/

Mougharbel, I., Hajj, A. E., & Riman, C. (2006). Remote Lab Experiments Models: A Comparative Study. *International Journal of Engineering Education, 22*(4), 849-857.

Mozilla Corporation. (2022). *WebRTC connectivity*. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity

Pérez-López,, E., Vázquez Atochero, A., & Cambero Rivero, S. (2021). Educación a distancia en tiempos de COVID-19: Análisis desde la perspectiva de los

estudiantes universitarios. *Revista Iberoamericana de Educación a Distancia, 24*(1), 330-342.

PI North America. (2021). *Time Sensitive Networking (TSN)*. https://us.profinet.com/digital/tsn/

Platzi. (2015). *PostgreSQL: qué es, cómo funciona y cuáles son sus ventajas*. https://platzi.com/blog/que-es-postgresql/

PROFIBUS Nutzerorganisation e.V. (2016). *PROFIBUS System Description.* Karlsruhe.

Quisnancela, W., & Iza, V. (2016). Implementación de un sistema de supervisión vía internet para prácticas en línea en el laboratorio de PLC's de la Universidad de las Fuerzas Armadas ESPE extensión Latacunga. Ecuador, Latacunga: Repositorio Universidad de las Fuerzas Armadas.

Raspberry Pi Foundation. (2021). *Raspberry Pi 4*. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Raspberry Pi Foundation. (2021). *Raspberry Pi 4 Tech Specs*. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Real Time Automation, Inc. (2021). *A little background Ethernet/IP*. https://www.rtautomation.com/technologies/ethernetip/

Real Time Automation, Inc. (2021). *An Introduction to Profibus*. https://www.rtautomation.com/technologies/profibus/

Real Time Automation, Inc. (2021). *An Introduction to Profinet IO*. https://www.rtautomation.com/technologies/profinet-io/

Red Hat, Inc. (2021). *What is a Raspberry Pi?*. Opensource: https://opensource.com/resources/raspberry-pi

Revilla-Cuesta, V., Skaf, M., Varona, J., & Ortega-López, V. (2021). The Outbreak of the COVID-19 Pandemic and its Social Impact on Education: Were Engineering

Teachers Ready to Teach Online? *International Journal of Environmental Research and Public Health, 18*(4), 2127.

Riman, C., Hajj, A. E., & Mougharbel, I. (2011). A Remote Lab Experiments Improved Model. *International Journal of Online Engineering (iJOE), 7*(1), 37-39. 10.3991/ijoe.v7i1.1460

Robledano, Á. (24 de septiembre de 2019). *Qué es MySQL: Características y ventajas*. https://openwebinars.net/blog/que-es-mysql/

Romero, E. (s.f.). *Heroku – Una plataforma para la creación de aplicaciones*. https://estebanromero.com/herramientas-emprender-desarrollar-proyectos/heroku-una-plataforma-para-la-creacion-de-aplicaciones/

Rómmel, F. (2021). *SQLite: La Base de Datos Embebida*. https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida

Rosencrance, L. (septiembre de 2021). *Remote Access*. TechTarget: https://searchsecurity.techtarget.com/definition/remote-access

Rossouw, F. (20 de abril de 2020). *Introduction to WebRTC*. https://medium.com/dvt-engineering/introduction-to-webrtc-cad0c6900b8e

Saavedra, J. (2016). Diseño de un servidor web utilizando hardware libre. Piura, Perú: Universidad Nacional de Piura.

Schneider Electric. (15 de diciembre de 2021). *Ficha técnica Magelis STU 5,7" Color touch 22mm - HMISTU855*. https://www.se.com/ar/es/product/download-pdf/HMISTU855

Schneider Electric. (15 de diciembre de 2021). *Ficha técnica Magelis STU módulo posterior - HMIS5T*. https://www.se.com/ar/es/product/download-pdf/HMIS5T

Schneider Electric. (15 de diciembre de 2021). *Ficha técnica Switch Ethernet TCSESU053FN0*. https://www.se.com/cl/es/product/download-pdf/TCSESU053FN0

Schneider Electric. (2021). *Siemens PROFIBUS/MPI S7 Protocol & Siemens Industrial Ethernet S7 protocol*. https://igss.schneider-electric.com/plc-scada-driver-8/

Schneider Electric. (2021). *Variador de velocidad Altivar Process ATV600 ATV630 1.5kW/2 hp 200...240 V IP21/UL tipo 1*. https://www.se.com/co/es/product/ATV630U15M3/variador-de-velocidad-altivar-process-atv600-atv630-1-5kw-2-hp-200-240-v-ip21-ul-tipo-1/

Shacklett, M. (octubre de 2021). *WebRTC (Web Real-Time Communications)*. TechTarget: https://www.techtarget.com/searchunifiedcommunications/definition/WebRTC-Web-Real-Time-Communications

Siemens. (5 de diciembre de 2019). *¿Qué características especiales, ventajas y propiedades ofrece el protocolo S7?*. https://support.industry.siemens.com/cs/document/26483647/%C2%BFqu%C3%A9-caracter%C3%ADsticas-especiales-ventajas-y-propiedades-ofrece-el-protocolo-s7-?dti=0&lc=es-WW

Siemens. (16 de enero de 2021). *Data sheet 6ES7212-1BE40-0XB0.* Automation24: https://media.automation24.com/datasheet/se/6ES72121BE400XB0_en.pdf

Siemens. (2021). *SIMATIC S7-1200 CPUs*. https://new.siemens.com/in/en/products/automation/systems/industrial/plc/s7-1200.html

Universidad Rey Juan Carlos. (s.f.). *Instalación de Django*. https://gsyc.urjc.es/grex/cursosweb/guion.html

Upadhyay, I. (23 de diciembre de 2020). *What is a Web Server? An Overview*. Jigsaw: https://www.jigsawacademy.com/blogs/cyber-security/what-is-a-web-server/

Valarezo, M., Honores, J., Gómez, A., & Vinces, L. (2018). Comparación de tendencias tecnológicas en aplicaciones web. *3C Tecnología. Glosas de Innovación*

aplicadas a la pyme, 7(3), 28-49. http://dx.doi.org/10.17993/3ctecno.2018.v7n3e27.28-49/

Válvulas Arco. (24 de octubre de 2019). *Presión del agua: ¿Cuál es el máximo y el mínimo que se debe alcanzar?*. https://blog.valvulasarco.com/presi%C3%B3n-del-agua-cual-es-el-maximo-minimo-que-se-debe-alcanzar

Vargas, J., Cuero, J., & Torres, C. (2020). Laboratorios Remotos e IOT una oportunidad para la formación en ciencias e ingeniería en tiempos del COVID-19: Caso de Estudio en Ingeniería de Control. *Revista Espacios, 41*(42), 188-198. 10.48082/espacios-a20v41n42p16

Villafañe, A., Brumovsky, A., & Aveleyra, E. (2019). Sistema de Laboratorios Remotos basado en una plataforma Raspberry/Arduino. *I Congreso Internacional de Ingeniería Aplicada a la Innovación y Educación de ISTEC.* Buenos Aires, Argentina.

W.W. Grainger, Inc. (2021). *Baldor Electric General Purpose Motor*. https://www.grainger.com/product/BALDOR-ELECTRIC-General-Purpose-Motor-29UX64

Weg. (2008). *DeviceNet Communication Manual.*

Anexos