UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de administración de prácticas remotas

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Manuel Alejandro Lara Huerta

ASESOR DE MATERIAL DIDÁCTICO

M.I. Yukihiro Minami Koyama

Ciudad Universitaria, Cd. Mx., 2024

AGRADECIMIENTOS

Este trabajo de titulación fue realizado gracias al apoyo del programa UNAM–DGAPA–PAPIIT IT102121 “Diseño de dispositivos mecatrónicos para apoyo en emergencias”.

Le agradezo a mi esposa por el apoyo en este largo proceso así como al profesor Yukihiro Minami Koyama por su seguimiento y apoyo para hacer esto posible.

CONTENIDO

CAPÍTULO 1
Introducción

1.1 Definición y objetivo

Laboratorio remoto FI es un sistema de software desarrollado en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, UNAM, que tiene el objetivo de permitir a profesores y alumnos del área de ciencias básicas la realización de prácticas de forma remota, con la finalidad de facilitar su desarrollo.

Se realiza a través de una página web donde los alumnos podrán acceder con sus datos, agendar y empezar alguna práctica, ver cámaras y datos de la práctica, así como instrucciones y controles donde podrán manipular los dispositivos para su desarrollo desde esta interfaz web. Se guardarán los datos y serán compartidos con los profesores que podrán utilizar esta información para sus registros.

Algunas materias que se tienen contempladas para el desarrollo de estas prácticas son: Mecánica, *Cinemática y Dinámica*, Electricidad y Magnetismo, Termodinámica y Ecuaciones Diferenciales. Se espera que más materias y prácticas se puedan incluir con el tiempo.

1.2 Descripción del sistema

Se pueden identificar diversos actores que utilizarán este, así como las funciones esperadas para cada uno:

- Alumnos
    - Ver y agendar práctica
        - Ver las prácticas que tienen asignadas por materia
        - Agendar una práctica en un día y una hora específica
        - Cambiar la hora y fecha de una práctica agendada
    - Realizar la práctica
        - Iniciar dicha práctica a la hora de una práctica agendada
        - Ver instrucciones e información importante para realizar la práctica
        - Ver cámaras y datos de sensores
        - Controles para manipular la práctica desde la interfaz
        - En caso de desconexión, volver a iniciar la práctica
        - Terminar la práctica y registrar la información relevante para revisión del profesor
- Profesores
    - Ver los grupos y los alumnos asignados
    - Ver el estado de las prácticas por cada alumno (terminada, sin empezar, entre otros)
- Desarrolladores de prácticas
    - Crear el control de una práctica, incluyendo cámaras, sensores, actuadores y pasos para realizarla.

Es importante que este sistema sea flexible, para que se pueda utilizar en una gran variedad de proyectos. Para esto, se identificaron algunos componentes compartidos y así diseñar el sistema para poder usarlo en varias situaciones.

Ya que sólo se permite el acceso a estas prácticas para una persona a la vez, además de la interfaz de la práctica es necesario crear una aplicación en la que alumnos como profesores puedan agendar espacios así como acceso a los datos recopilados en ellas, este control será de gran ayuda tanto para profesores como para alumnos.

CAPÍTULO 2
Antecedentes

2.1 Importancia de los laboratorios para la docencia

Se le puede denominar laboratorio para la docencia a un espacio diseñado para apoyar el proceso de enseñanza–aprendizaje, en particular de ciencias como la física y la química, mediante la utilización de recursos y tecnologías educativas. Brindan oportunidades para que los estudiantes interactúen directamente con el mundo material, o con datos extraídos del mundo material, usando las herramientas, técnicas de recopilación de datos, modelos y teorías de la ciencia [1].

El uso de laboratorios en la actividad experimental desempeña un papel crucial en el aprendizaje de la ingeniería. Estos laboratorios proporcionan un entorno equipado con recursos y tecnologías educativas específicas que, permiten a los estudiantes de ingeniería llevar a cabo experimentos prácticos y aplicar los conceptos teóricos en ambientes controlados. Es un lugar equipado con los medios y medidas de seguridad necesarias. También se utilizan para realizar investigaciones, experimentos, prácticas y trabajos científicos, técnicos o tecnológicos [2].

Los laboratorios para la docencia tienen los siguientes objetivos: mejorar el dominio de la materia en estudio; desarrollar el conocimiento científico; comprender la complejidad y la ambigüedad del trabajo empírico; desarrollar habilidades prácticas; comprender la naturaleza de la ciencia; cultivar el interés por la ciencia y el interés por aprender ciencia; desarrollar habilidades de trabajo en equipo, según Singer, S. R. Hilton, M. L. y Schweingruber, H. A. [3].

Entre las habilidades más importantes que se pueden adquirir en los laboratorios para la docencia, se pueden mencionar las siguientes: el empleo correcto de máquinas e instrumentos de medición; el manejo y procesamiento de los datos obtenidos; el desarrollo de destrezas de razonamiento lógico y organización; la construcción y comunicación de valores relativos a la naturaleza de la ciencia, con base en Barberá, O. Valdés, P. [4].

Además, las actividades de los estudiantes en los laboratorios para la docencia están enfocadas al desarrollo de habilidades sociales, como el del trabajo en equipo, de conciencia sensorial por medio del uso de los sentidos humanos para reunir información, hacer juicios sólidos al formular conclusiones sobre problemas del mundo real, y forma actitudes tales como la ética, propiciando el comportamiento con los más altos estándares éticos, incluida la presentación de información de manera objetiva y la interacción con la integridad [5].

Cuando las prácticas experimentales están contextualizadas, es decir, relacionadas con situaciones conocidas por los estudiantes, concebidas desde el planteamiento de un problema práctico, "… resultan ser atractivas, dinámicas, divertidas y motivadoras, puesto que permiten despertar el interés de los estudiantes hacia el aprendizaje de esta área de la ciencia en contextos de la vida diaria, a partir del planteamiento de un problema" [6].

2.2 Laboratorios remotos

En el Instituto Tecnológico de Massachusetts, MIT, se desarrolló el concepto iLabs en 1999, que consiste en un laboratorio automatizado cuyo equipo está conectado al servidor de dicho laboratorio, y que puede ser operado en forma remota, a través de la internet, por los usuarios autorizados. De esta manera se pretendió incrementar sustancialmente el número de usuarios de los servicios del laboratorio en todo el mundo y a compartir los recursos de infraestructura de este tipo con otras universidades.

Las asignaturas Estática, Cinemática y Dinámica, que se imparten en la División de Ciencias Básicas de la Facultad de Ingeniería, UNAM, cuentan con laboratorio no curricular, con objeto de complementar y mejorar el aprendizaje logrado por los alumnos en sus clases teóricas. Una gran cantidad de alumnos se inscribe semestre a semestre a dichos laboratorios, dada la buena opinión que tienen los profesores de esta actividad experimental, por lo que fomentan ampliamente entre sus estudiantes la asistencia a aquéllos. Pero por falta de infraestructura física y de equipamiento, los grupos están saturados, por lo que se requiere formar brigadas de cinco y hasta de seis alumnos, disminuyendo sustancialmente el impacto que puede generar esta actividad en el logro de un aprendizaje significativo [7].

Por la razón mencionada en el párrafo anterior, en el año de 2004 se intentó resolver esta problemática con la creación de un laboratorio en el cual se pudieran realizar las prácticas

de la asignatura Cinemática y Dinámica de forma remota a través de la internet, que facilitaran que un mayor número de estudiantes tuvieran acceso a él desde cualquier sitio que cuente con una conexión.

De esta forma, habría un mejor aprovechamiento de los recursos materiales con que cuenta la Universidad, convirtiéndose, a su vez, como un difusor de estos conocimientos a un sector más amplio de la población.

2.3 Semblanza de proyectos relacionados

2.3.1 Proyecto EN106204

En el año 2004 se inició el proyecto EN106204 denominado “Creación de un laboratorio remoto accedido por medio de la internet para la asignatura Cinemática y Dinámica” [8].

Se establecieron como objetivos: el diseño de prácticas de laboratorio de Cinemática y Dinámica que pudieran realizarse de forma remota a través de la internet; la implementación de dichas prácticas con el diseño y construcción de dispositivos mecatrónicos y sensores electrónicos requeridos; y el desarrollo del software para la operación del sistema con equipos de cómputo a través de la internet.

Además, uno de los objetivos más importantes que se busca en el desarrollo de proyectos de investigación y desarrollo tecnológico en instituciones de educación superior, es la formación de profesionistas de calidad que sean capaces de resolver problemas tecnológicos que tenga la sociedad. Para conseguirlo, se integra a alumnos de semestres avanzados, tanto para la realización de su servicio social como de su trabajo de titulación. Asimismo, se involucra a otros docentes, tanto para la generación de ideas en el desarrollo de los proyectos, la coordinación de las actividades de los alumnos, así como la asesoría y dirección de tesis. De esta manera, es posible capacitarlos para que en el futuro puedan encabezar sus propios proyectos de esta índole.

En este proyecto se lograron desarrollar prototipos mecatrónicos para la realización remota de cinco prácticas de la asignatura Cinemática y Dinámica:

- Movimiento rectilíneo uniformemente acelerado
- Análisis de la dinámica de un cuerpo empleando el método de trabajo y energía
- Tiro parabólico y caída libre
- Movimiento en medios con fricción fluida
- Cuerpos rígidos conectados con movimiento plano.

Durante el desarrollo de estos sistemas mecatrónicos se titularon nueve alumnos, quienes desarrollaron su trabajo en seis trabajos de tesis de licenciatura a lo largo de ocho años y se presentaron seis artículos en congresos nacionales.

2.3.2 Proyecto PE111218

En 2018 se inició el desarrollo del proyecto PE111218 “Diseño de prácticas de laboratorio para fortalecer el aprendizaje de conceptos matemáticos en Ciencias Básicas”, cuyo objetivo principal fue fortalecer el aprendizaje de conceptos matemáticos de algunas asignaturas de la División de Ciencias Básicas, a través del diseño y planteamiento de prácticas de laboratorio que describan experimentalmente fenómenos físicos, químicos o biológicos, así como el uso de software especializado, que permita orientar a los alumnos hacia la vinculación de la teoría con la práctica [9].

Las dos contribuciones más importantes relacionadas con este proyecto, que se lograron ofrecer en su desarrollo fueron:

- Prototipo para la realización remota de la práctica “Ley de enfriamiento de Newton”
- Sistema de realidad aumentada para la realización de la práctica “Aplicación de las ecuaciones diferenciales en circuitos eléctricos”.

Estas contribuciones permitieron titularse a dos alumnos con un trabajo de tesis de licenciatura y, además, se involucró a quince académicos, quienes idearon y propusieron varias prácticas, como la “Aplicación de las operaciones binarias en las compuertas lógicas de los circuitos electrónicos”, “Determinación de la constante de velocidad y orden de una reacción química mediante el análisis de ecuaciones diferenciales” y “Convolución y sus aplicaciones en efectos acústicos”. Adicionalmente, se elaboraron y presentaron cuatro artículos en dos congresos nacionales y dos internacionales.

2.3.3 Proyecto PE109021

La pandemia de COVID–19, en un inicio, provocó la suspensión de las actividades académicas en los primeros meses de 2019. Dada la grave situación sanitaria que se presentó, para la segunda mitad de dicho año la gran mayoría de las instituciones educativas, desde el nivel básico hasta el superior, decidieron iniciar las clases en línea a través de software de vídeo–conferencia y el uso de plataformas educativas.

En la Facultad de Ingeniería de la UNAM, se decidió emplear recursos de laboratorios virtuales tales como los simuladores, para dar atención al aprendizaje experimental de

asignaturas de física y química de los primeros semestres, aunque por la premura con la que se tomó esta decisión, sólo se logró una cobertura de menos de la mitad del contenido práctico de los programas de materia.

Algunos profesores de Análisis de Circuitos, Electrónica Básica y Circuitos Digitales propusieron a sus estudiantes la realización de prácticas en casa, dada la accesibilidad de los dispositivos a emplear, la mayoría económicos y que podían adquirirse por internet. Sin embargo, se percibió la necesidad de instrumentos de medición como el osciloscopio, y otros dispositivos como programadores universales, para conseguir mejores resultados.

Debido a esta situación, se tomó la decisión de buscar alguna solución a esta problemática, para lo cual se inició en el año 2021 el proyecto PE109021 denominado "Creación de material didáctico y dispositivos para la implementación de prácticas experimentales a distancia en la División de Ciencias Básicas", cuya finalidad fue la ideación de prácticas experimentales con posibilidad de realización remota sincrónica, y virtual o asincrónica en casa, con la generación de material didáctico y el diseño de dispositivos mecatrónicos que lo hicieran posible, de manera que permitieran la cobertura eficaz y completa de los objetivos de las asignaturas involucradas, en particular Electricidad y Magnetismo y Mecánica [10].

En este proyecto participaron poco más de quince académicos y unos diez alumnos, de los cuales dos de ellos están concluyendo su trabajo de tesis de licenciatura, quienes desarrollaron prototipos automatizados para la realización remota de dos prácticas:

- Fuerza de origen magnético sobre conductores
- Movimiento rectilíneo uniformemente variado

la primera de Electricidad y Magnetismo y la segunda de Mecánica.

Asimismo, al igual que el los proyectos antes mencionados, se presentaron cuatro ponencias, tres en congresos internacionales y uno en otro internacional, y se publicó un artículo en la revista digital de la ANFEI.

2.3.4 Proyecto IT102121

Finalmente, en el mismo año de inicio del proyecto anterior, 2021, se decidió también proponer el proyecto IT102121 denominado "Diseño de dispositivos mecatrónicos para apoyo en emergencias", con financiamiento de la UNAM, el cual fue aprobado.

Su objetivo principal fue la creación de dispositivos tecnológicos que sean útiles en situaciones emergentes como una pandemia o un desastre natural, dentro de las que se consideró la realización de prácticas de laboratorio experimentales de forma remota o

virtual, para mejorar el aprendizaje de asignaturas de física y química de los estudiantes de la Facultad de Ingeniería y de la Facultad de Estudios Superiores Aragón [11].

Por la razón anterior, se impulsó el desarrollo de este sistema de software, **Laboratorio remoto FI**, para facilitar a los estudiantes la realización de las diversas prácticas de realización remota con las que se cuenta, tal como se mencionó en el capítulo anterior.

Además de este material didáctico, en el proyecto citado se desarrollaron otros dos productos de índole experimental: un prototipo automatizado para la realización remota de la práctica Movimiento compuesto circular–parabólico de una partícula; propuesta de un osciloscopio accesible a estudiantes.

Hasta la fecha, en este proyecto se abordó la realización de cuatro trabajos de titulación y se han presentado cinco artículos en congresos nacionales e internacionales.

2.4 Reflexiones adicionales sobre los laboratorios para la docencia

Luego de tres semestres de impartir los cursos en línea, se pudo regresar a la modalidad presencial a partir de febrero de 2022, por lo que fue posible impartir los laboratorios para la docencia en las instalaciones de la Facultad. Sin embargo, la experiencia de aprendizaje práctico logrado con los simuladores hizo patente su utilidad para algunos casos específicos en los que la realidad se puede adaptar permitiendo simplificar el aprendizaje al destacar los conceptos importantes y evitar los detalles confusos.

Por consiguiente, se puede mencionar que existen situaciones excepcionales en las que las prácticas virtuales y las que pueden realizarse en casa son una valiosa alternativa para satisfacer el aprendizaje experimental, al permitir el contacto fáctico en el ámbito de la educación a distancia.

Además, se puede asegurar que tanto los recursos digitales como los virtuales para la realización de actividades prácticas en la enseñanza de la ingeniería son un complemento enriquecedor para la formación de los futuros profesionistas de esta disciplina.

CAPÍTULO 3
Diseño

Para lograr el sistema propuesto, se necesitan dos sistemas independientes que se puedan comunicar entre sí, uno que será la página web donde alumnos y profesores podrán acceder y realizar las prácticas, y otro para que la parte física de las prácticas pueda comunicar la información hacia la interfaz del alumno. Al primer sistema se le denominará `webapp` y al otro `labserver`.

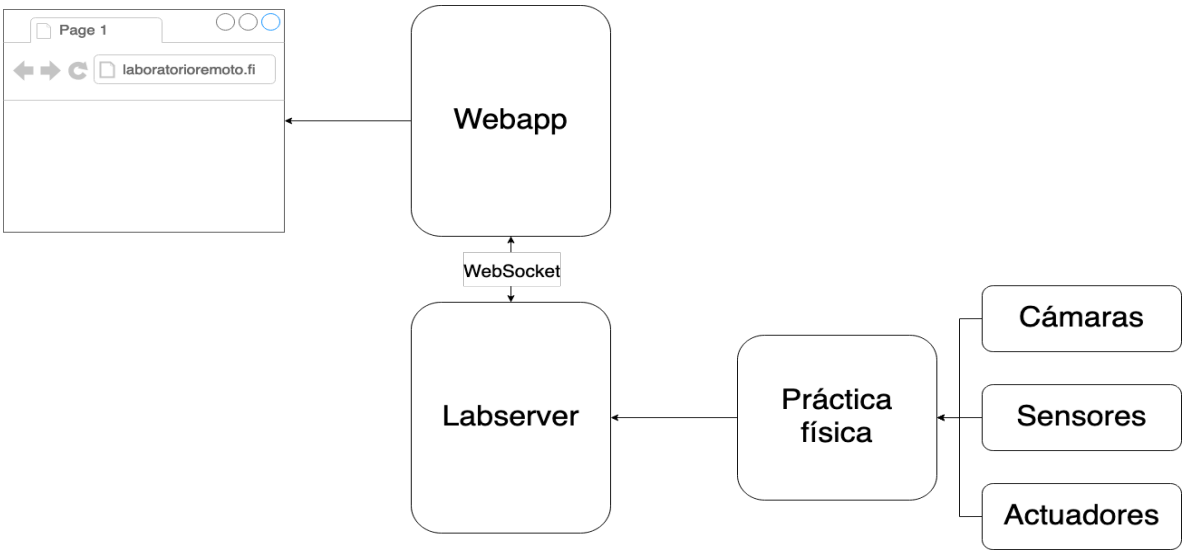


Figura 1 Arquitectura del sistema propuesto.

La comunicación de ambos sistemas deberá ser en tiempo real, el protocolo elegido para este caso es el websocket, más particularmente con la biblioteca *socket.io*. La arquitectura, en términos generales, es como la que se muestra en la figura 1.
Ya que se ha definido la estructura general, se puede hablar a detalle de cada uno de los elementos.

3.1 Webapp

Para facilidad su uso, se decidió desarrollar la `webapp`, o servidor de la aplicación web, con el framework llamado *NextJS*, que permite desarrollar aplicaciones web incluyendo el servidor de una manera muy sencilla. Utiliza el lenguaje *Javascript*, así como otros archivos de bibliotecas importantes tales como *ReactJS* y *MUI* (Material UI), que ayudarán a generar los componentes de interfaz. Para la base de datos se decidió usar *MongoDB* por la flexibilidad al generar la estructura de datos. En la siguiente lista se presentan los elementos más importantes utilizados en este proyecto:

- NextJS: generación de aplicaciones web, que incluye un servidor en nodejs en el que se pueden generar puntos de acceso a una API
- React: modularización de componentes
- MUI: elementos de interfaz (botones, listas, texto, etc.)
- SocketIO: extiende los websockets con otras funcionalidades como reconexión automática
- MongoDB: base de datos de tipo archivos
- Next-Auth: agrega autenticación y seguridad en una aplicación de *NextJS.*

A continuación, se detalla la implementación de las diversas funcionalidades en la plataforma.

3.1.1 Inicio de sesión y grupos

Para dar de alta a los alumnos y profesores, en este momento se realiza de forma manual en la base de datos; la información está en la colección *users*. Al momento de iniciar sesión, se solicita el id del usuario y su contraseña, los cuales están indicados en esta colección (esto no es lo más adecuado y se espera en un futuro generar un sistema para almacenar contraseñas). Para el manejo de las sesiones, se utiliza una biblioteca *Next-Auth* que permite mantener la sesión en el navegador una vez verificadas las credenciales.

Se usa el mismo sistema para iniciar sesión de alumnos y profesores; para los alumnos el id es el número de cuenta, pero en el caso de los profesores, ya que no se cuenta con un número de cuenta, se puede usar cualquier otro identificador como su dirección de correo electrónico o su nombre. La ventana para iniciar sesión se puede observar en la figura 2.

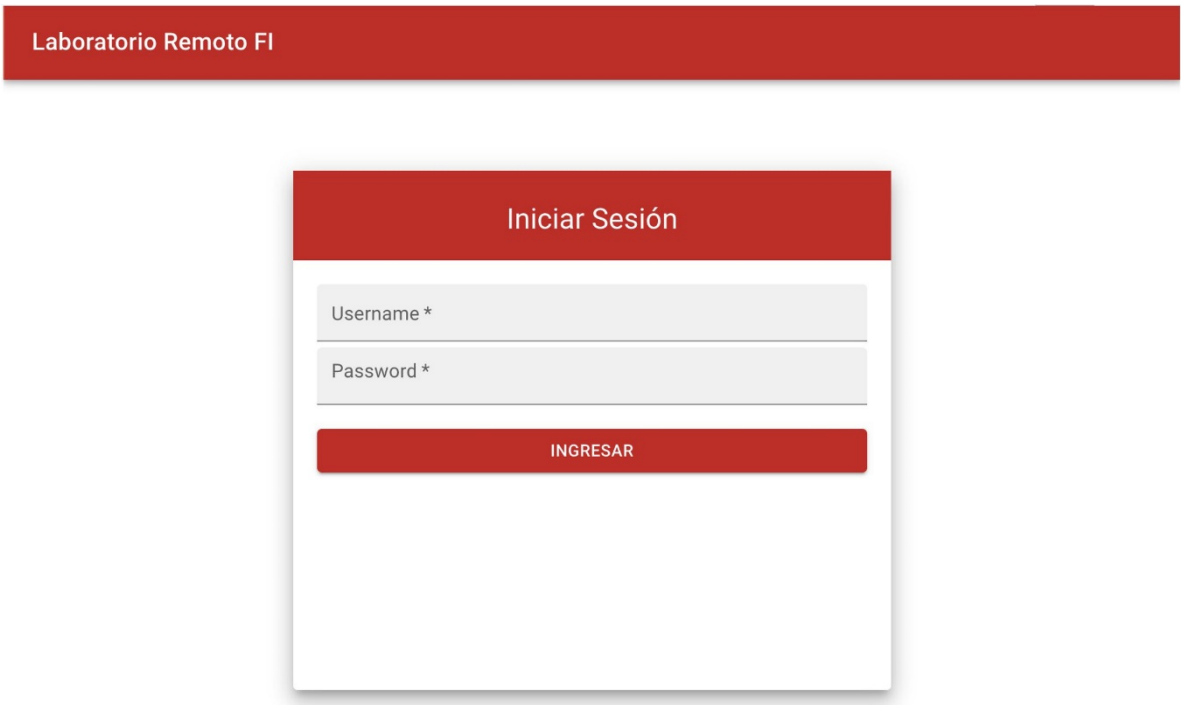


Figura 2 Ventana para iniciar sesión

Una vez con la sesión iniciada se podrá ver el nombre de la persona, así como la fecha y hora actual, tal como se muestra en la figura 3. Se podrá ver el botón para cerrar sesión en la esquina superior derecha. También se puede cerrar una sesión borrando las cookies de la página.

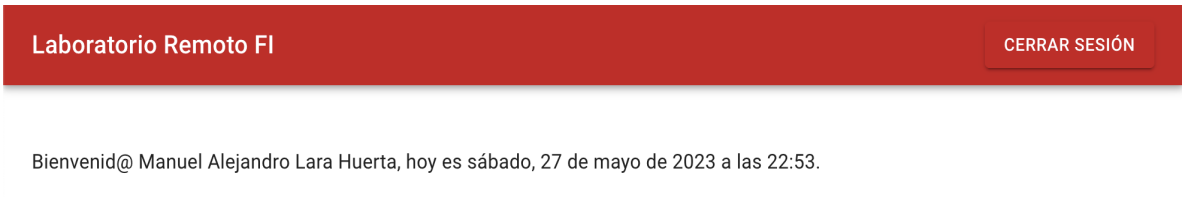


Figura 3 Información desplegada después de iniciar sesión

3.1.2 Lista de materias y prácticas

Una vez que un alumno inicia sesión correctamente, puede ver sus materias y prácticas asignadas. Cada alumno tiene la información de los grupos a los que pertenece en el campo `groupsIds`, que es una lista con los identificadores de cada grupo; estos identificadores se construyen con tres partes: primero el semestre al que pertenece, el código de la materia y al final el número de grupo, que permite tener un identificador único por cada materia de cada grupo de cada semestre. Ej: `2021-2_1500_2`.

El sistema verifica todos los grupos asignados para el alumno que tiene la sesión iniciada y los muestra en pantalla, posteriormente revisa cada práctica asignada a cada materia, para esto se extrae el código de la materia del `groupId`, y en la tabla de `subjects` se encuentra la materia. En este registro también se incluye el campo `practicesIds` el cuál es una lista de las prácticas asignadas a dicha materia.

Una vez que se cuenta con los identificadores de las prácticas, se puede obtener la información del nombre desde la colección de `practices`, de esta manera permite saber el nombre para mostrar por cada práctica. Para saber el estado de cada práctica, se hace la búsqueda en la colección de `schedules`, que cuenta con el registro de todas las reservaciones para realizar una práctica, por lo que se filtra por medio de `practiceId`, `studentId` y `subjectId`, que debe regresar sólo un registro por cada práctica, y de este registro se obtiene el estado, si es que ya está terminada, empezada, sin empezar etc. Existen muchos estados diferentes para el `Schedule`; se pueden revisar a detalle en la sección de especificaciones. Algunos de estos estados se guardan en la base de datos, pero otros tienen que ser calculados dependiendo de la fecha actual y de las fechas de la práctica. En la figura 4 está la imagen de la ventana en la que se muestra las materias, prácticas y su estado, asignadas al alumno.

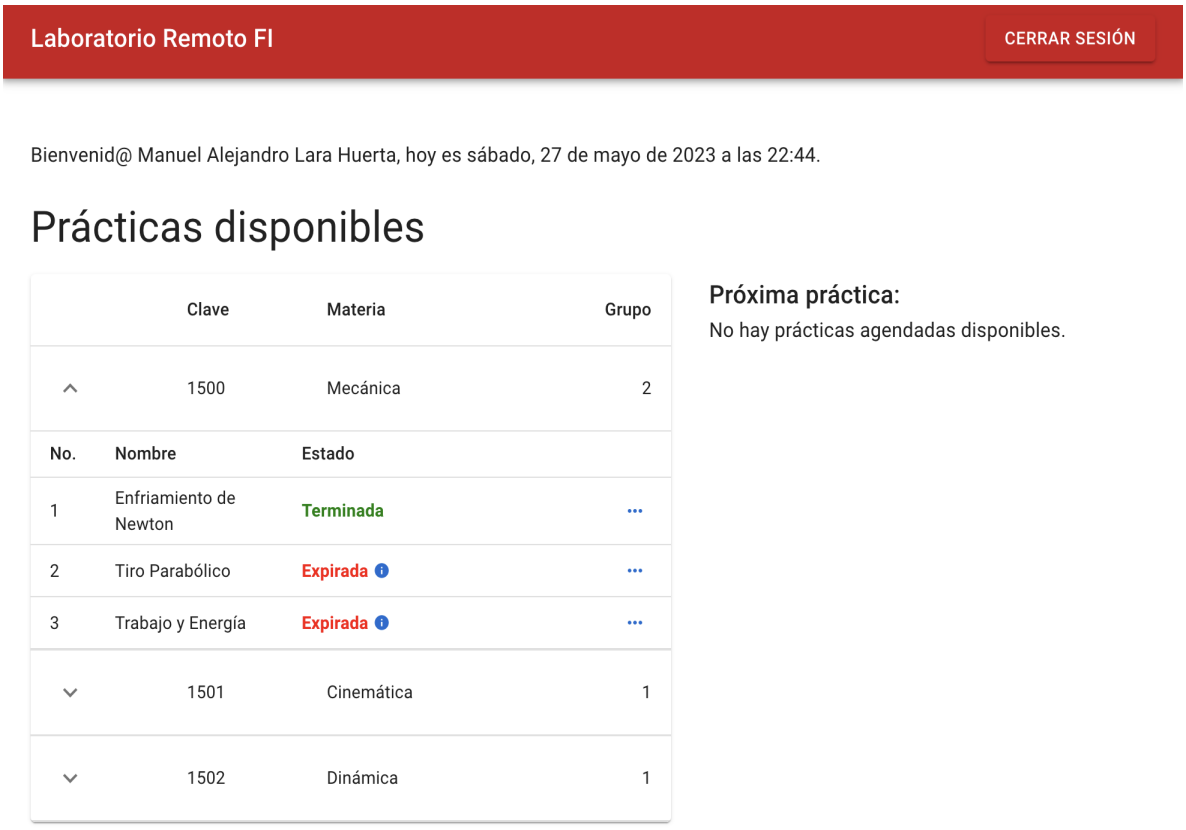


Figura 4 Ventana que muestra las materias, prácticas y su estado, asignadas al alumno.

Existe un botón de detalles de cada práctica donde se puede observar la información de las fechas de inicio y fin de la práctica, la duración, y si es que ya se agendó la hora y fecha de ésta, tal como se muestra en la figura 5.

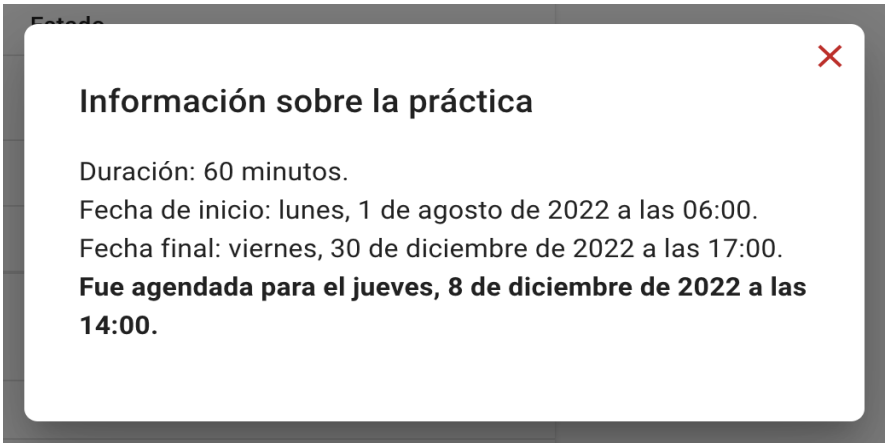


Figura 5 Información de las fechas de inicio y fin de la práctica, incluyendo la hora

3.1.3 Agendar una práctica

Para poder agendar una práctica, primero se debe estar dentro de las fechas establecidas en la información de la práctica; de ser así, al ver la lista de prácticas se verá el estado *Agendar*, tal como se muestra en la figura 6. Al dar *click* en el botón de detalles, se podrán ver las acciones disponibles. En los únicos casos que no se podrá agendar la práctica son cuando ya esté terminada o que ya haya pasado el tiempo para su realización; en los demás casos, se podrá ya sea agendar o reagendar.

Figura 6 Ventana que muestra la posibilidad de agendar una práctica

En la figura 7 se muestra la ventana en la que se proporciona la información sobre la práctica y el botón a oprimir para agendarla.

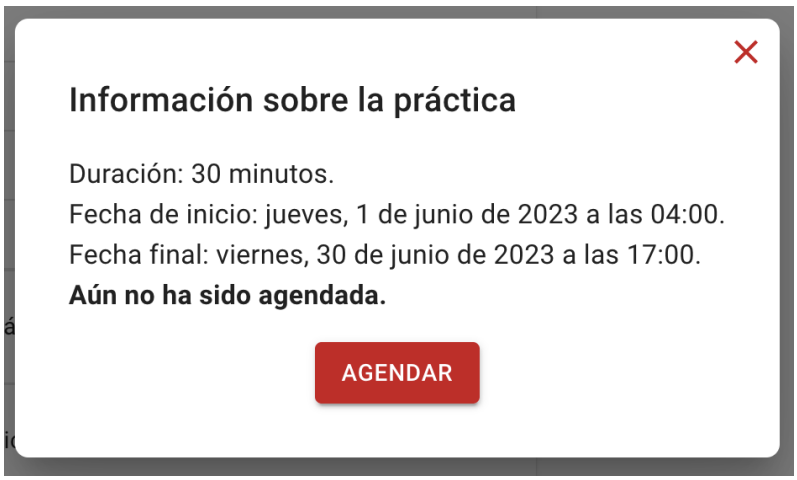


Figura 7 Ventana en la que se puede agendar una práctica

Aquí se puede seleccionar la fecha y horario disponible que se requiera en una ventana como la que se puede observar en la figura 8. Los horarios que ya no estén disponibles simplemente no aparecerán en esta lista. Al confirmar que se desea agendar dicho horario, se hará una llamada al servidor para confirmar que sigue disponible y asignarlo al alumno que tiene la sesión iniciada. Al hacer esto, se agrega un registro dentro de la colección de `schedules` con el id del estudiante, de la práctica, materia y horario asignado. El horario se guarda como la fecha y hora en formato *epoch*, que es utilizado para bloquear el horario a los demás alumnos. También en este registro se añaden las acciones, una vez iniciada la práctica.

Figura 8 Ventana para reservar la fecha y hora de la práctica

Si se desea cambiar el horario, se puede hacer mientras no haya pasado el tiempo límite para realizar la práctica. Solo hay que volver a la sección de detalle y seleccionar la opción de *reagendar*.

3.1.4 Inicio de una práctica

Cuando se tiene agendada una práctica y el horario ya ha empezado al acceder a la página, se podrá ver activado el botón que permite iniciar la práctica, tal como se muestra en la figura 9.

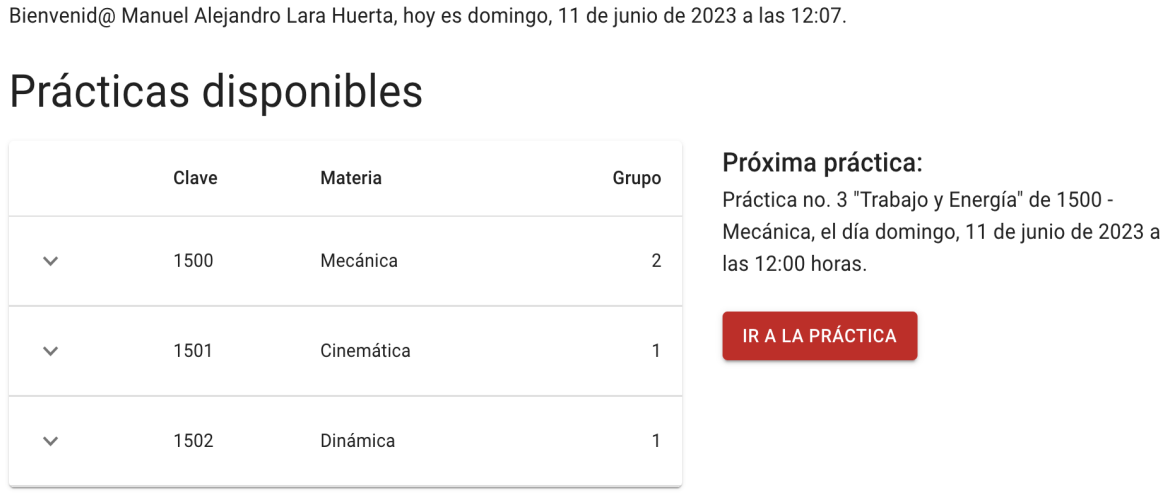


Figura 9 Botón para iniciar la práctica activada

Al dar click en el botón para iniciar, la interfaz se conectará al servidor y puerto especificados en el registro de la práctica en el campo *raspIP*, se tratará de iniciar una conexión con el servidor de *Socket.io*; existe una ligera medida de seguridad donde se manda un usuario y una contraseña, en el futuro ésta podrá ser actualizada. El `labserver` es el que recibe esta solicitud. En caso de que exista algún error, éste se enviará a la interfaz o se mostrará en la terminal del servidor. Si la conexión es exitosa y la configuración inicial se realiza de manera correcta, el servidor envía toda la información de la práctica a la interfaz; esta información está descrita en un archivo llamado `metadata.yml` (se hará una descripción más detallada de esto en la sección del `labserver`). Al final de este proceso se podrá ver la información de la práctica y un botón para empezarla, tal como se muestra en la figura 10.

El flujo de las prácticas está definido por *páginas*; cada página incluye instrucciones, datos y acciones. Las instrucciones simplemente se incluyen en un texto, donde se indica qué es lo que se debe realizar; los datos son información que viene de sensores desde la práctica; las acciones son aquéllas disponibles en cada página.

Figura 10 Información de la práctica y el botón activado para empezarla

La ventana en la que aparecen las *Instrucciones* se muestra en la figura 11.

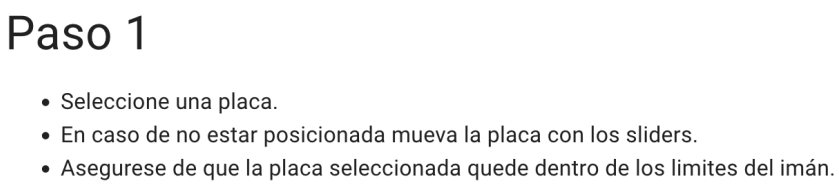


Figura 11 Ventana de instrucciones

De forma similar, la ventana de *Acciones* se muestra en la figura 12.

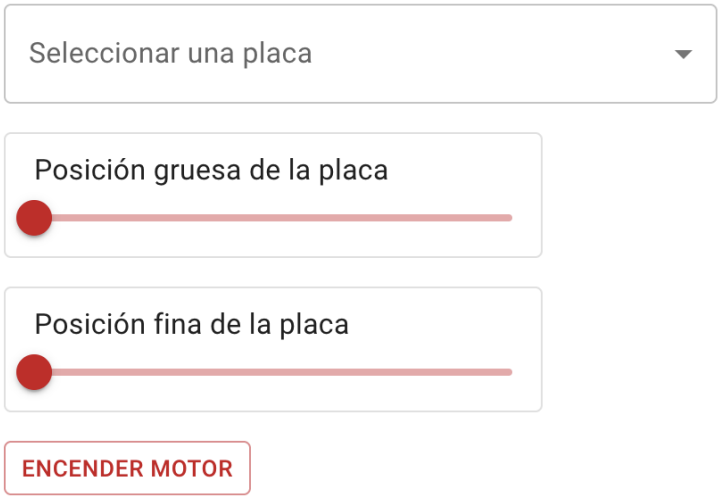
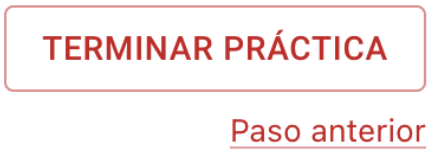


Figura 12 Ventana de acciones

Finalmente, en la figura 13 se puede observar la configuración de la ventana de *Datos*.

Figura 13 Ventana de datos

Cada acción que realice el alumno será registrada en la base de datos de la práctica, para que pueda ser revisada por el profesor. Una vez que el alumno ha terminado los pasos de una página, se podrá ir a la siguiente página dando click en el botón *Siguiente Paso*. De esta manera se avanza a través de la práctica hasta que se llega a la última página donde el texto cambia a *Terminar la práctica*. Una vez que se termina la práctica, se regresa automáticamente a la pantalla inicial. Toda la información de la práctica es guardada y, además, cambia el estado a *Terminada*. La ventana en la que se tiene el botón para declarar terminada la práctica se puede observar en la figura 14.



Paso anterior

Figura 14 Botón para declarar la terminación de la práctica

3.2 Labserver

En esta sección se explicará en detalle cómo es que se configura y desarrolla una práctica con el `labserver`, o Servidor de prácticas.

El servidor está desarrollado con *NodeJS*, utilizando la biblioteca *ExpressJS* como base. La opción recomendada es utilizar una tarjeta de desarrollo Raspberry; sin embargo, cualquier computadora capaz de ejecutar *NodeJS* puede funcionar.

La comunicación con la interfaz se realiza con *SocketIO* a través del cual se reciben comandos y se envía la información en tiempo real. La conexión con el hardware de la práctica puede ser de dos maneras diferentes, una es utilizando los pines de entrada/salida de la Raspberry. Esta opción puede no funcionar en todos los casos, si es que se necesita procesamiento que solo un microcontrolador puede hacer; en esos casos la recomendación es conectar algún otro microcontrolador especializado (una tarjeta Arduino, por ejemplo) y conectarlo a la Raspberry a través del puerto serial.

Las cámaras web funcionan de una manera diferente al resto de los sensores; para enviar el vídeo se debe utilizar algún servidor que pueda hacer stream de vídeo a través de *HTTP*, no es necesario usar uno en específico, pero en esta guía se utilizará *mjpeg_server*. En la figura 15 se muestra la arquitectura del Labserver.

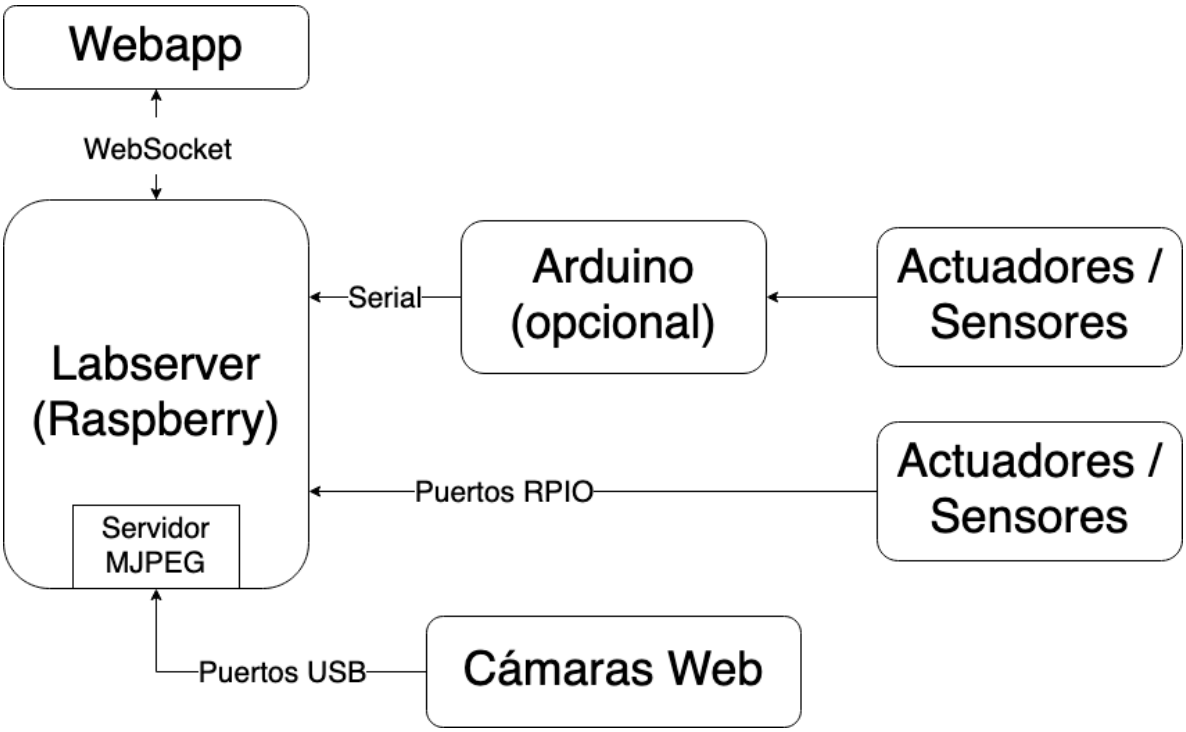


Figura 15 Diagrama de bloques de la arquitectura del Labserver

Para configurar la práctica hay dos archivos que hay que modificar, el primero es `metadata.yml` el cual tiene la información general de la práctica, acciones, datos, páginas

y algunos otros datos. Además de este, se requiere modificar uno llamado `practice.js` que es el que se conectará con el hardware; dentro de éste se definirá toda la lógica de la práctica. Al iniciar el servidor, se va a leer el archivo `metadata.yml`, y esta información se usa para construir la interfaz y mandarle toda la información.

3.2.1 Archivo metadata

El objetivo de este archivo es definir la información general; se utiliza un tipo de formato *Yaml* ya que es fácil de leer. A continuación, se listan todos los campos que se pueden definir en este archivo:

- name: nombre de la práctica
- objective: descripción de la práctica, puede ser un texto largo.
- actions: lista de acciones disponibles
- data: lista de datos disponibles
- videos: lista de videos disponibles
- pages: lista de páginas.

3.2.2 Acciones

Las acciones son todas las *interacciones* que el usuario puede tener desde la interfaz, por ejemplo, botones que activen o desactiven algo, un componente deslizable para elegir un valor, una caja para seleccionar alguna opción, entre otros, ilustrados en la figura 16.

Figura 16 Ejemplos de componentes para realizar algunas acciones

Los componentes soportados actualmente son los siguientes:

- button: un botón simple que puede tener un texto y ejecuta una única acción al ser presionado

- toggle: un botón que puede tener dos estados *activado* o *desactivado*, puede tener un texto
- slider: un componente deslizable que puede seleccionar un valor que está entre un mínimo y un máximo
- selector: un componente para seleccionar una opción de una lista.

El detalle de los campos para cada una de las acciones se puede encontrar en la sección de especificaciones. Este es un ejemplo de este campo:

```
actions:
  corrienteOn:
      name: Encender corriente
  selectPlaca:
      name: Seleccionar una placa
      type: selector
      values:
      placaA: Placa A
      placaB: Placa B
  corrienteMode:
      name: Ajustar corriente
      type: slider
      minValue: 0
      maxValue: 255
      step: 1
```

3.2.3 Datos

Los datos son información que se presentará en la interfaz en tiempo real: simplemente se tendrá el `id` del campo y el nombre con el que se mostrará. Hay casos en los que se pueden recibir números, pero no siempre esto es lo que se quiere mostrar, por ejemplo, si se requiere verificar si un motor está o no encendido, o si un dispositivo está en cierta posición. Por esta razón, conviene tener la capacidad no solo de recibir un dato y mostrarlo sino también enlazarlo a una *etiqueta* y que de esta manera sea más clara la información proporcionada, lo cual se define en el campo `labels`, como se puede observar en la figura 17.

Figura 17 Información proporcionada por la interfaz: nombre de campo y valor

A continuación se muestra un ejemplo de cómo presentar información sencilla y también información que esté mapeada a ciertas etiquetas:

```
data:
  opticalPositionD:
      name: Limite seguro derecho
      labels:
      0: Dentro del sistema
      1: Fuera del sistema
  pwmLedValue:
      name: Valor de PWM del led
      units: grados
  sensorDistance:
      name: Distancia al sensor
      units: cm
```

3.2.4 Vídeos

Para las cámaras web, en realidad el servidor `labserver` no es el que se encarga de mandar esta información, debido a que el procesamiento para codificar y enviar por internet el vídeo es muy complejo, por lo que es mejor utilizar herramientas especializadas.

Se decidió utilizar servidores que se encarguen de enviar el vídeo por *HTTP* usando el formato de imagen, lo que permite cambiar de manera sencilla la calidad del vídeo para adaptarlo a los requerimientos de red. Existen muchos proyectos que se pueden utilizar y que funcionan con diversos sistemas operativos que, por lo general, se encuentran con un nombre similar a *mjpeg_server*.

Una vez que se elija uno de estos servidores y se tenga funcionando, será importante conocer la *ip* y el puerto de este servidor, ya que de esta manera se podra acceder al vídeo. Una vez que se tengan estos datos, se puede declarar esta información en la sección de vídeos y tener un `label` para cada cámara web. En seguida se muestra la forma como se establecen estos datos.

```
videos:
  temperatureVideo:
      name: Monitor de temperatura
      url: http://159.223.156.70:8080?action=stream
      width: 640
      height: 480
```

3.2.5 Páginas

Las páginas son la parte que une todos los elementos anteriores, es una lista de elementos donde se incluye primero las instrucciones de la página, después los identificadores de las acciones disponibles, los identificadores de los datos y los identificadores de los vídeos. Es importante notar que en las secciones pasadas, cada elemento cuenta con un código identificador y este es el que se usa para agregarlos a cada página, tal como se muestra a continuación.

```
pages:
  - instructions:
      - Seleccione una placa.
      - En caso de no estar posicionada mueva la placa con los sliders.
      dataIds:
      - mecanicaPositionD
      - mecanicaPositionI
      actionIds:
      - selectPlaca
      - positionPlacaG
      - positionPlacaF
      videoIds:
      - placaVideo
```

```
  - instructions:
      - Activar la corriente
      - Ver el monitor de corriente y anotar los datos
      dataIds:
      - currentState
      - currentValue
      actionIds:
      - toggleCurrent
```

3.2.6 Archivo de “practice.js”

La función principal de este archivo es realizar toda la lógica dentro de los microcontroladores para leer sensores, procesar la información y regresar los datos hacia la interfaz. Esto lo realiza en conjunto con la configuración del archivo `metadata.yml`, donde se definen los identificadores de las acciones, sensore y datos; estos identificadores son usados en este archivo “practice.js” para enviar a la interfaz los datos y ejecutar las acciones. Esto es, por ejemplo, si se define una acción para encender una resistencia, se le llama `encenderResistencia` y se puede tener un dato que indique la temperatura del agua que se denominará `temperaturaAgua`, la configuración dentro de `metadata.yml` sería:

```
(metadata.yml)
actions:
  encenderResistencia:
      name: Encender corriente
      type: button
data:
  temperaturaAgua:
      name: Temperatura del agua
      units: grados
```

La función del archivo “`practice.js`“ es tomar estos identificadores de acciones y datos, para las acciones definir la lógica que se debe realizar y para los datos colocar el valor que debe ser enviado.

Por ejemplo, si se desea que cuando la acción de `encenderResistencia` sea activada, se encienda una resistencia, se lea el valor de la temperatura de un sensor y se envíe este dato, en grados centígrados, para lograrlo se requiere que dentro del archivo se encuentre una clase llamada `ExamplePractice`, que es la que contiene todos los métodos necesarios para interactuar con la interfaz; los métodos que se proponen son los siguientes:

- constructor()
- command()
- init().

El primer método que se debe conocer es el constructor de la clase, en este se pueden inicializar las variables que se desea enviar a la interfaz, para lo cual solo hay que definirlas con el mismo identificador. También dentro de este constructor se tendrá una función que se estará ejecutando en un bucle, con la cual se pueden leer sensores y efectuar acciones dependiendo de las lecturas de estos sensores, por ejemplo, si se requiere parar un motor por que se encendió un sensor de fin de carrera.

A continuación se muestra el código del método constructor().

```
constructor() {
  this.status = "not ready";

  // Variables de ayuda
  // Los valores que queramos mandar a la interfaz
  // deben tener el mismo identificador que en metadata.yml
  this.temperaturaAgua = 0;

  // lectura de variables y lógica, similar al loop de arduino
  setInterval(() => {
      // Leemos el sensor y esto se enviará a la interfaz
      this.temperaturaAgua = rpio.read(PIN_SENSOR_TEMPERATURA);

      // Podemos leer otros sensores y hacer algo dependiendo
      let finalCarrera = rpio.read(pin_FCD);
      if (finalCarrera) {
      stopMotor();
      }
  }, 10);
}
```

Otra sección muy importante es `command()`, en la que se puede definir el comportamiento que se requiere para cada acción que se tenga establecida, por ejemplo, qué es lo que debe suceder cuando se presiona el botón para encender la resistencia. Esto está dentro de una estructura `switch` con el identificador de cada acción.

También se pueden establecer restricciones a las acciones, dependiendo del estado de ciertos sensores, por ejemplo, no realizar ninguna acción mientras el botón de emergencia esté activado. Los errores o mensajes que se deseen enviar a la interfaz también se pueden configurar aquí. En seguida se muestra un ejemplo de la función command().

```
command(command, value) {
  if (this.emergencyButtonState) {
      return {
        status: "error",
        message:
          "No se puede recibir ningún comando mientras el botón de
           emergencia esté presionado",
      };
  }

  // Agregar acciones
  switch (command) {
      case "encenderResistencia":
        this.encenderResistencia();
        break;
      default:
        return {
          status: "error",
          message: `No se reconoce el comando ${command}`,
        };
  }

  return { status: "success" };
}
```

La última función es `init()`, esta se va a ejecutar cada vez que se inicie una práctica. Permite llevar la práctica a un estado *inicial* antes de que el alumno pueda empezar. Esto se determina con una variable llamada `this.status`: cuando esta variable se coloca con el valor `"ready",` quiere decir que la práctica está lista para iniciar. Su código se muestra a continuación.

```
init() {
  // Checar el estatus de las cosas y si no está en el
  // 'estado inicial' cambio el estado de la práctica
  // a 'initializing' y mando los comandos para que esté en
  // el estado inicial

  this.off();
  this.setPWMLedValue(0);

  let someSensorValue = readSomeSensor();
```

```
  while(someSensorValue !== 0) {
      someSensorValue = readSomeSensor();
  }

  // Cuando el sensor es 0 podemos iniciar la práctica
  this.status = "ready";
}
```

Hay dos formas en las que se pueden leer sensores y encender pines para interactuar con la práctica: una es utilizar los pines *rpio* de la Raspberry, lo que se puede lograr con cualquier archivo de biblioteca que esté disponible para *NodeJS*. La recomendada en este caso es *node-rpio* con la cual se pueden leer y escribir datos tanto digitales como analógicos. En caso de que esto no se pudiera efectuar y fuera necesario un microcontrolador más específico, es posible utilizarlo y conectarlo con la tarjeta Raspberry a través del puerto serial, con el empleo de archivos de bibliotecas muy sencillos como *serialport*.

3.3 Hacer funcionar el PWM

El proyecto que se está utilizando para habilitar el uso de los pines de entrada y salida se puede encontrar en node-rpio. Este proyecto soporta el uso de PWM en los pines que tienen habilitada esta funcionalidad por hardware.

3.3.1 Deshabilitar audio

En este momento, hay un problema con el audio y el pwm, por lo que primero hay que deshabilitar éste en el archivo `/boot/config.txt`. Esto se puede hacer con la edición del archivo como root.

```
sudo nano /boot/config.txt
```

Y se convierte en comentario la línea:

```
# dtparam=audio=on
```

3.3.2 Incluir el archivo de bilbioteca y configuración inicial

Para poder utilizar el PWM es necesario que el archivo de biblioteca se configure para utilizar el sistema `/mem` en lugar del configurado por defecto `/gpiomem`. Esto se puede lograr con las siguientes líneas:

```
var rpio = require("rpio");
rpio.init({ gpiomem: false });
```

Este cambio requiere que el programa sea ejecutado como usuario `root`.

```
sudo yarn dev
```

3.3.3 Habilitar el pin y mandar el dato

Para poder utilizar el pin como PWM es necesario configurarlo así con la función `rpio.open` de la siguiente manera:

```
rpio.open(13, rpio.PWM);
```

Y para colocar el pin en un valor en particular:

```
// rpio.pwmSetData(pin, value);
rpio.pwmSetData(13, 1024);
```

3.3.4 Otras configuraciones de PWM

También existen otros dos valores que se pueden configurar, el divisor de frecuencia del PWM y el rango total, de la siguiente manera:

```
rpio.pwmSetClockDivider(8);
rpio.pwmSetRange(13, 1024);
```

CAPÍTULO 4
Especificaciones

4.1 Tipos de acciones

4.1.1 Botón

Campo	Descripción	Ejemplo
type	Tipo de componente, para los botones esto es opcional	button
name	Nombre que se mostrará del botón	Encender motor

4.1.2 Toggle

Campo	Descripción	Ejemplo
type	Tipo de componente	toggle
name	Nombre que se mostrará del botón	Encender motor

4.1.3 Slider

Campo	Descripción	Ejemplo
type	Tipo de componente	slider
name	Nombre que se mostrará del botón	Encender motor
minValue	Valor minimo que se puede seleccionar	0
maxValue	Valor máximo que se puede seleccionar	255
step	Cantidad de incrementos por cada _paso_	1

4.1.4 Selector

Campo	Descripción	Ejemplo
type	Tipo de componente	selector
name	Nombre que se mostrará del botón	Encender motor
values	Lista con las opciones disponibles, el id de cada opción será el valor enviado al servidor	- firstOption: Placa A - otherOption: Placa B

4.2 Reglas para determinar los estados de los “schedules”

4.2.1 Valores que se guardan en la base de datos

Valor posibles en base de datos	Cuándo se asigna valor
FINISHED	Al terminar de manera adecuada la práctica
STARTED	Al iniciar una práctica
SCHEDULED	Cuando se agenda un horario para realizar la práctica

4.2.2 Valores que se muestran en la interfaz

Valor en base de datos	Condición	Valor mostrado en pantalla	Alerta
FINISHED	Cualquier condición	Terminado	
STARTED	currDate < scheduleTimestamp + practiceTimeframe	Empezada	
STARTED	currDate > scheduleTimestamp + practiceTimeframe	Empezada	¡Es necesario reagendar!
SCHEDULED	currDate < scheduleTimestamp	Agendada	
SCHEDULED	currDate > scheduleTimestamp && currDate < practiceEndDate	Reagendar	¡Es necesario reagendar!
undefined	currDate < practiceEndDate && currDate >= practiceStartDate - reserveTime	Agendar	
*	currDate > practiceEndDate	Expirada	Ya no es posible agendar
*	currDate < practiceStartDate - reserveTime	No disponible	

4.3 Bases de datos

4.3.1 users

Tabla para guardar la información de los usuarios, tanto alumnos como profesores, para los alumnos se utiliza el id para iniciar sesión (el número de cuenta), y para los profesores se utiliza el correo.

Campo	Tipo	Descripción	Ejemplo
id	string	Identificador del usuario, en el caso de alumnos es el número de cuenta, en el caso de los profesores puede ser algo diferente.	304316636
type	string	Tipo de usuario. Puede ser `student` o `professor`.	student
email	string	Correo electrónico.	micorreo@correo.com
password	string	Contraseña para entrar al sistema. (Esta forma de almacenar contraseñas no es seguro)	mYP4sword
name	string	Nombre del usuario.	Juan Gonzalez Gonzalez
groupdIds	array<string>	Un array con los ids de los grupos. Los identificadores se conforman por: {año}-{semestre}_{código de materia}_{número de grupo}	2021-2_1500_2

4.3.2 subjects

Tabla para guardar información de las materias.

Campo	Tipo	Descripción	Ejemplo
id	string	Código de la materia.	1500
name	string	Nombre de la materia.	Cinemática y dinámica
practicesIds	array <string>	Array con los códigos identificadores de las prácticas asignadas a esta materia.	caida_libre

4.3.3 practices

Tabla para guardar la información de las prácticas.

Campo	Tipo	Descripción	Ejemplo
id	string	Código de la práctica.	caida_libre
name	string	Nombre de la práctica.	Caída libre
raspIp	string	IP de la raspberry para conectarse a la práctica. Debe incluir el puerto del servidor de socket.io para iniciar la conexión.	135.145.0.0:8000
startDate	double	Tiempo en formato unix (epoch) de la fecha de inicio de la práctica. La hora indicada aquí será la hora del día que la práctica empieza cada día.	1659330000000
endDate	double	Tiempo en formato unix (epoch) de la fecha de termino de la práctica. La hora indicada aquí será la hora del día que la práctica dejará de estar disponible.	1669957200000
timeFrame	int	Minutos requeridos para terminar la práctica.	60

4.3.4 schedules

Tabla para guardar cada reservación para realizar una práctica, guarda el estado de esa reservación, la práctica, hora, usuario que la va a realizar y toda la información de eventos una vez realizada la práctica.

Campo	Tipo	Descripción	Ejemplo
practiceId	string	Código de la práctica.	caida_libre
studentId	string	Número de cuenta del alumno que va a realizar la práctica.	304316636
raspIp	string	IP de la raspberry para conectarse a la práctica. Debe incluir el puerto del servidor de socket.io para iniciar la conexión.	135.145.0.0:8000
subjectId	string	Identificador de la materia de la práctica. (Esto es importante porque el sistema puede	1500

		tener la misma práctica asignada a diferentes materias)	
status	string	Estado de la reservación de la práctica. Puede ser 'FINISHED', 'STARTED', 'STARTED_EXPIRED', 'SCHEDULED', 'READY_TO_START', 'READY_TO_SCHEDULE', 'SCHEDULE_EXPIRED', 'NO_LONGER_AVAILABLE', 'NOT_YET_AVAILABLE'	FINISHED
timestamp	double	Tiempo en formato unix (epoch) de la fecha y hora de la reservación.	1659330000000
log	array<{date, message}>	Array con los elementos de eventos que sucedieron en la práctica, cada objeto tiene `date`: Fecha y hora del evento, y `message`: Mensaje describiendo el evento.	[{date: \<Date object\>, message: "Se inició la práctica"}]

4.3.5 groups

Tabla para identificar un grupo y los alumnos que forman parte de él. Con esta información es como un profesor puede identificar todos los alumnos que están asignados a sus grupos.

Campo	Tipo	Descripción	Ejemplo
subjectId	string	Código de la materia.	1500
groupNumber	string	Número del grupo.	1
studentsIds	array <string>	Array con los números de cuenta de los alumnos que pertenecen al grupo.	[304316636, ...]

CAPÍTULO 5
Pruebas y resultados

El sistema está siendo utilizado para la implementación de la interfaz en algunas prácticas, esto ayuda a que los alumnos que están desarrollando dichas prácticas no se preocupen por desarrollar una interfaz y solo en como procesar y conectar los datos.

Una de estas prácticas es para las materias de “Mecánica” y “Cinemática y Dinámica”, que incluye las prácticas de “Planos inclinados” y “Movimiento rectilineo uniformemente variado”, se presentan los detalles de la implementación a continuación.

La práctica original mueve un elemento con peso a lo largo de una plataforma inclinada con sensores que permiten medir la velocidad de este objeto dependiendo de la inclinación del plano, se muestra una ilustración de esta en la figura 18.

Figura 18 Implementación actual de la práctida de “Planos inclinados”

El primer paso para que esta práctica funcione de forma remota es que los sensores, acciones y datos tienen que poder ser manejados por intenet además de sistemas automáticos de seguridad, esto implica cambios en los elementos utilizados por ejemplo el uso de sensores para determinar la posición de la base sobre el riel, cámaras para determinar el ángulo de la plataforma entre otros.

Es importante mencionar que el diseño de una práctica remota necesita tener un balance adecuado entre sistemas digitales y automáticos con sistemas manuales ya que el aprendizaje requiere participación activa de los alumnos lo cual no se logra si todo el sistema es automático. En el caso particular de esta práctica el posicionamiento del ángulo de la plataforma es algo que se deja de forma manual para tal propósito.

Los cambios anteriormente mencionados pueden verse en el prototipo ilustrado en la figura 19.

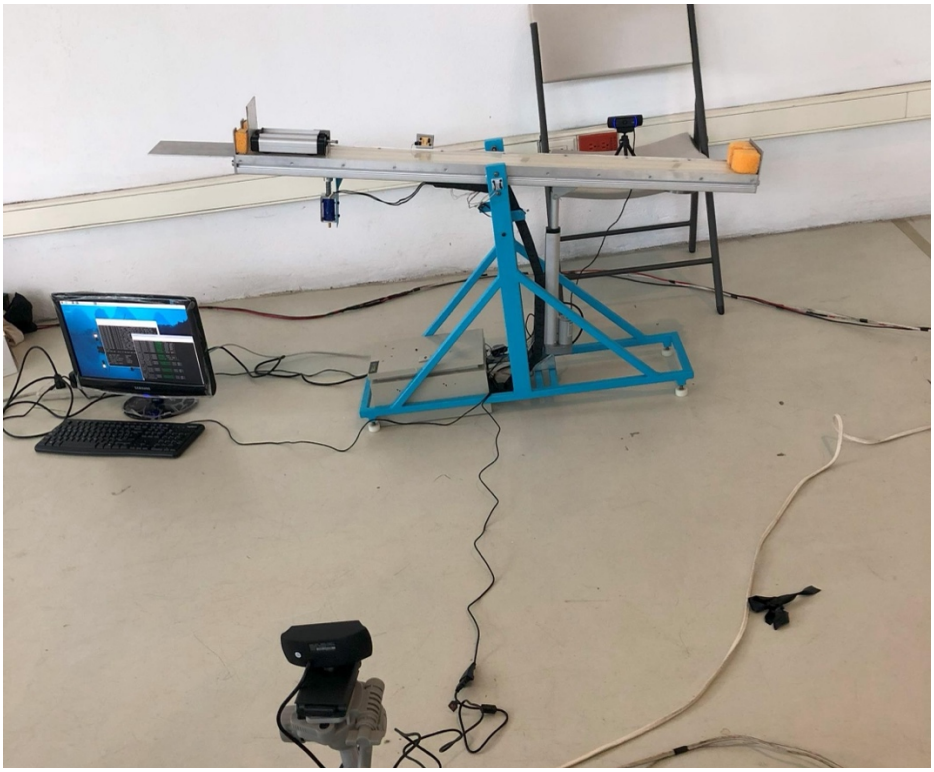


Figura 19 Prototipo para la práctica remota de "Planos inclinados"

Dentro de los procesos del diseño de la práctica se determinan cuales son las “acciones“ dentro del sistema (mover plataforma, liberar carro, etc), algunas de estas acciones pueden ser internas y otras para que el alumno las pueda ejecutar. En esta práctica el desarrollador definió las siguientes acciones:

- Girar la rampa a la derecha y a la izquierda
- Cambiar la velocidad de giro
- Activar solenoide
- Medir ángulo actual
- Lector de rango de movimiento
- Interruptor de emergencia

Todo esto es primero definido en el archivo `metadata.yml`, se puede observar el archivo de esta práctica a continuación, el cuál no solo incluye las acciones sino también los sensores, datos, cámaras y páginas con las instrucciones.

```
(metadata.yaml)

name: Practica ejemplo
objective: Agregar objetivos de la práctica.
actions:
```

```
  pwmSwap:
    name: PWM grueso / PWM fino
    type: toggle
  turnLeft:
    name: giro a la izquierda
  turnRight:
    name: giro a la derecha
  lockPosition:
    name: Fijar posición como horizontal (posición 0 grados)
    type: toggle
  lockTarget:
    name: Fijar ángulo de inclinación
    type: toggle
  powerSolenoid:
    name: Soltar objeto
  turnOff:
    name: Desactivar todos los dispositivos
  resetPosition:
    name: Reiniciar lanzamiento
data:
  rawAngleData:
    name: ángulo crudo
    units: grados
  newAngleData:
    name: ángulo actual
    units: grados
  motionSensor:
    name: Sonar
  solenoidInfo:
    name: Solenoide info
    labels:
      0: apagado
      1: activo
  voltageSensor:
    name: Sensor de voltaje
    labels:
      0: apagado
      1: encendido
videos:
  mainVideo:
    name: Camara principal
    url: http://192.168.1.79:8080?action=stream
    width: 640
    height: 480
  detailVideo:
```

```
    name: Camara de detalle
    url: http://192.168.1.79:8080?action=stream
    width: 640
    height: 480
pages:
  - instructions:
      - Usa los controles y la cámara para coloca la rampa en
posición horizontal
      - Fija la posición actual como horizontalcon el botón
correspondiente
      - Al terminar avanza al siguiente paso
    dataIds:
      - rawAngleData
      - newAngleData
    actionIds:
      - pwmSwap
      - turnRight
      - turnLeft
      - lockPosition
      - turnOff
    videoIds:
      - detailVideo
  - instructions:
      - Coloca la rampa en el ángulo pedido en la práctica
      - Fija el ángulo actual con el botón correspondiente
      - Al terminar avanza al siguiente paso
    dataIds:
      - newAngleData
    actionIds:
      - pwmSwap
      - turnRight
      - turnLeft
      - lockTarget
      - turnOff
    videoIds:
      - mainVideo
  - instructions:
      - Realiza los lanzamientos
      - Presiona el botón "Soltar objeto" para realizar un
lanzamiento
      - Presiona el botón "Reiniciar lanzamiento" para regresar el
carro a la posición inicial
      - Realiza 10 lanzamientos exitosos
    dataIds:
      - newAngleData
```

```
  actionIds:
    - powerSolenoid
    - resetPosition
    - turnOff
  videoIds:
    - mainVideo
```

Una vez que toda esta información está definida el siguiente paso es escribir el código que se va a ejecutar una vez que la interfaz mande y reciba los comandos. Este está definido en el archivo practice.js aunque puede apoyarse de otros archivos de biblioteca. Para esta práctica este archivo puede verse en el apendice A2.

El resultado de esta implementación en la interfaz se puede observar en las figuras 20, 21 y 22 que representan los pasos 1, 2 y 3 de la práctica.

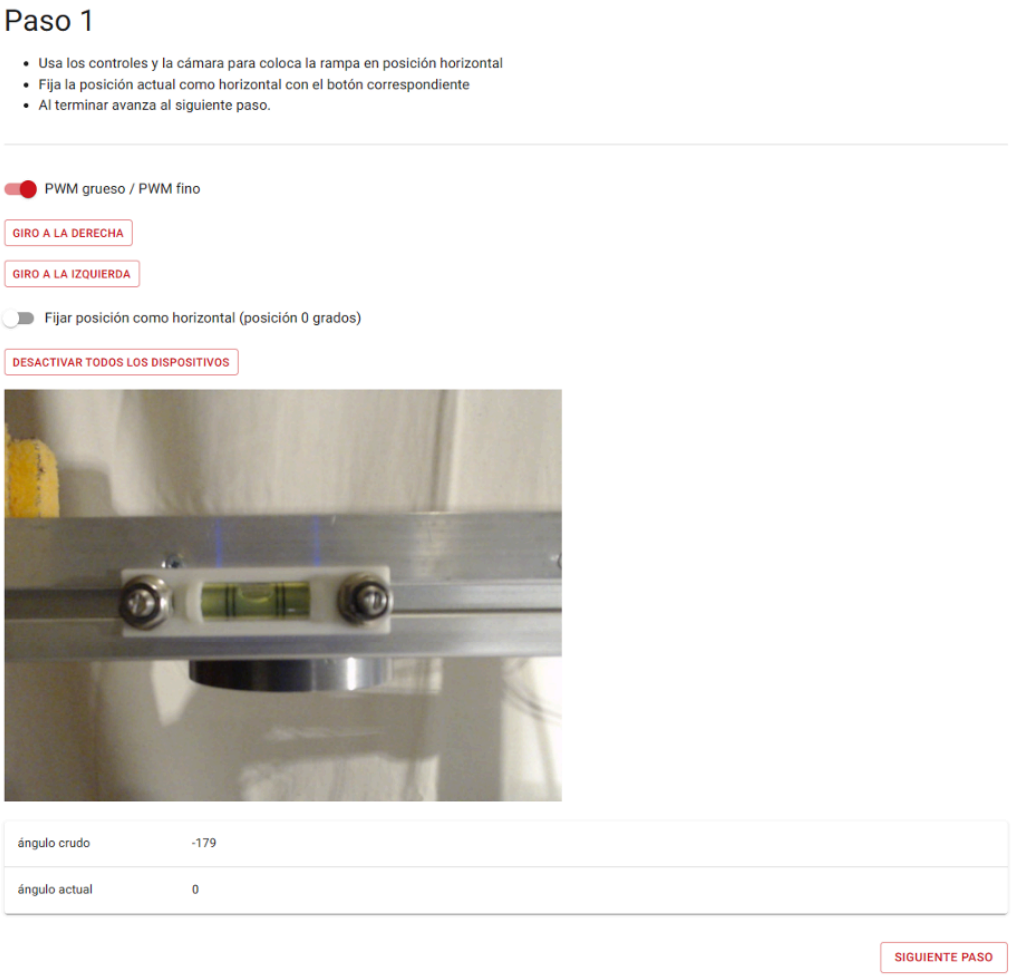


Figura 20 Paso 1 de la práctica de "Planos inclinados"

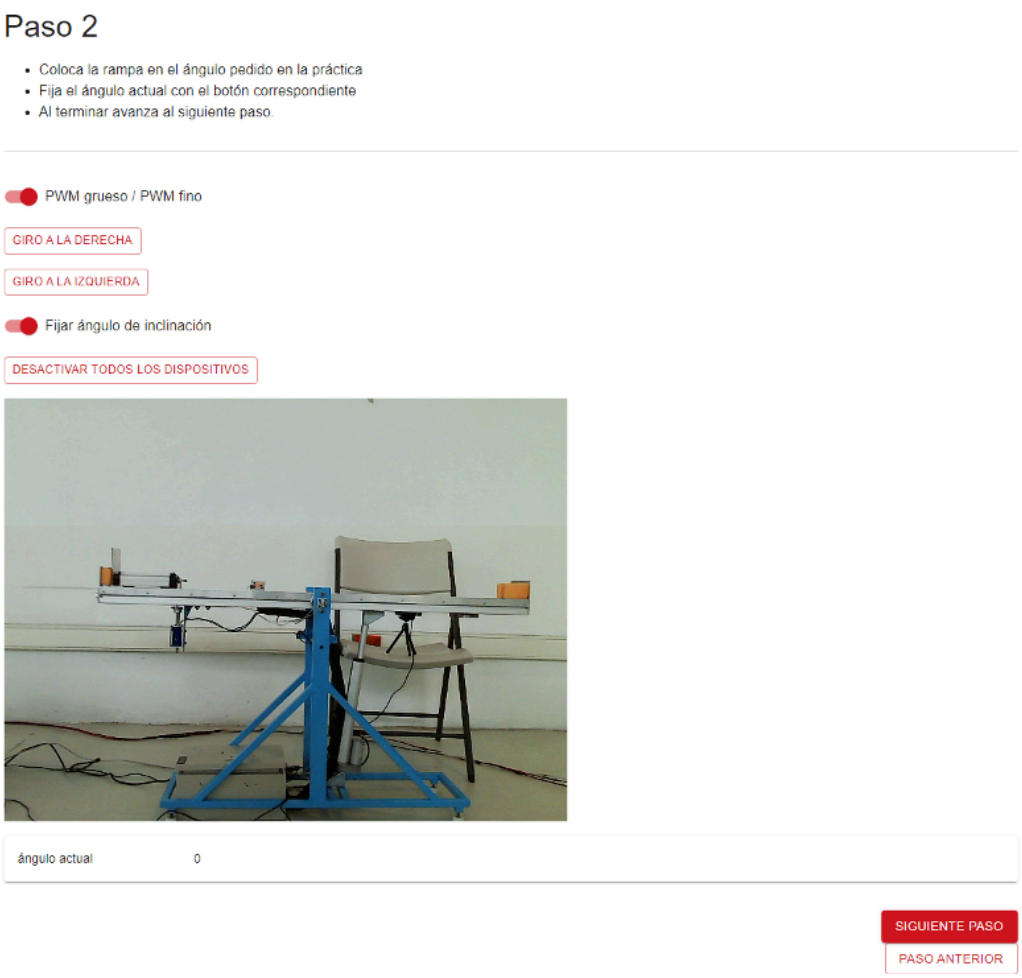


Figura 21 Paso 2 de la práctica de "Planos inclinados"

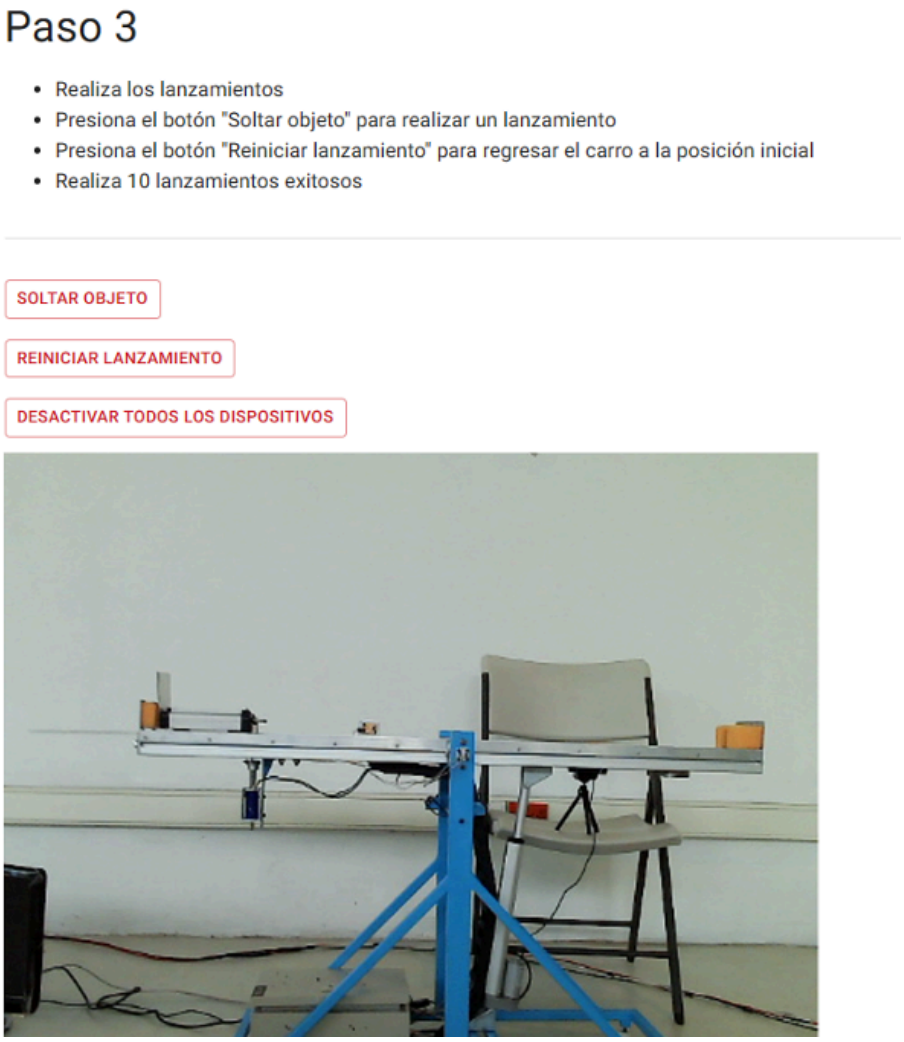


Figura 22 Paso 3 de la práctica de "Planos inclinados"

Esta práctica se encuentra avanzada para su terminación recibiendo comentarios positivos, este sistema ayuda a crear una base para conectar a internet cualquier sistema robótico pudiendo ser reutilizado por varios proyectos compartiendo soluciones y estrategias, algunos de los puntos positivos que se compartieron son:

- La facilidad de uso del archivo yaml para crear la interfaz solo con texto
- La clara división entre la práctica y la interfaz
- El poder usar la interfaz desde cualquier dispositivo sin necesidad de conectarse a la práctica directamente
- La interfaz minimalista

Sin embargo, aún hay áreas que mejorar para hacer este proceso más sencillo, las cuales se comentarán en la siguiente sección.

CAPÍTULO 6
Conclusiones y trabajo futuro

6.1 Conclusiones

El sistema de software Laboratorio remoto FI, creado con el desarrollo de este trabajo, cumplió con los principales objetivos establecidos en su planteamiento, de ofrecer una herramienta que facilitara la realización de prácticas experimentales de algunas asignaturas de la División de Ciencias Básicas, de forma remota a través de la internet, por parte de las y los estudiantes de la Facultad.

Este software pudo aplicarse con éxito para las pruebas piloto de realización remota de tres prácticas, una de Termodinámica conocida como la de “Ley de enfriamiento de Newton”, otra de Electricidad y Magnetismo con título “Fuerza de origen magnético sobre conductores”, y la tercera de Mecánica relacionada con “Planos inclinados”.

Aunque se ha logrado mucho en esta primera fase, aún hay piezas importantes que realizar, pero una vez superadas, el impacto que este proyecto puede tener es grande, al proporcionar herramientas de desarrollo, de manera que el proyecto de laboratorios remotos sea una realidad y con ello coadyuvar en el desarrollo académico de los alumnos de la Facultad, así como abrir espacios de innovación.

Este proyecto tiene un objetivo bastante ambicioso ya que son muchas piezas que tienen que trabajar en conjunto de manera correcta. Lo que se ha logrado hasta ahora es un buen fundamento sobre el cuál seguir construyendo. Como se mencionó en la sección anterior, se han recibido comentarios muy buenos sobre el diseño modular, siendo la separación de la interfaz y el sistema de prácticas uno de ellos, ya que permite el enfocarse solo en lo relevante. Para el caso del desarrollo de prácticas, no es necesario

hacer ninguna modificación al código de la interfaz y solo construirla con el archivo yaml, el cuál es la conexión entre interfaz y práctica.

Sin embargo, al ser este proyecto bastante amplio no ha sido posible terminar todas las funcionalidades en esta primera fase; se hablará de las diferentes áreas que aún hace falta trabajar y cuáles son los comentarios que se han hecho que facilitaran su uso en próximos desarrollos.

6.2 Trabajo futuro

6.2.1 Interfaz

Aunque se ha presentado cómo funciona la interfaz en general, ésta aún no está en un estado productivo, esto quiere decir que funciona en el desarrollo pero es necesario instalar esta en un servidor para que los alumnos puedan acceder a ella, en este sentido los puntos que aún hacen falta son:

Requerimientos obligatorios

- Instalar la aplicación en un servidor de la Facultad y asignarle una una URL fija para que los alumnos puedan acceder a ella.
- Los datos en este momento son solo de prueba por lo que es necesario crear la base de datos con información real de profesores y alumnos. Parte de esta propuesta incluye automatizar la construcción de datos de alumnos, profesores y grupos al conectarse directamente a esta información en otros servidores, esto es deseable aunque no completamente necesario. Otra opción es generar una interfaz interna para dar de alta esta información de una manera más interactiva.
- Definir las IP finales de las prácticas que serán incluidas y confirmar que se pueden conectar de manera adecuada.

Requerimientos opcionales

- Mejorar el estilo de la interfaz
  - Tipografía, tamaño de letra y colores
  - Acomodo de los elementos de las prácticas, el posicionamiento de botones, acciones, tablas.
- Dar la capacidad de cambiar el acomodo de los elementos de la práctica desde el archivo yaml. Esto ha sido comentado, pero resulta ser un cambio más complejo que los otros, ya que habría que definir un API de acomodo de elementos, pero si se logra podría dar mucha flexibilidad a los desarrolladores de prácticas para acomodar mejor su interfaz especifica.
- Mejorar el sistema de mensajes y errores.

6.2.2 Labserver

Del lado del labserver, el principal problema que se ha visto es del lado de la documentación, cómo saber de manera más clara cómo resolver ciertos problemas y guias más completas en cómo usar el sistema, ya que son varias piezas trabajando en conjunto es fácil no entender cómo funciona todo. Para resolver esto, se necesita una buena guía; existe una guia en proceso pero aún no está completa, puede encontrarse en el repositorio de documentación dentro del proyecto de github en la siguiente liga https://github.com/LaboratorioRemotoFi/docs.

De manera opcional se podría desarrollar una versión de labserver en Python, ya que la mayoría de los alumnos está más familiarizado con este lenguaje y hay más archivos de biblioteca especializados de robótica en Python. La parte compleja de esto es que los websocket y funciones asincronas son más complejas en Python, pero sin duda se puede hacer.

APÉNDICES

A.1 Código del programa

- Código de la aplicación `webapp` se puede encontrar en el siguiente repositorio de github https://github.com/LaboratorioRemotoFi/webapp
- El código de la aplicación `labserver` se puede encontrar en el siguiente repositorio de github https://github.com/LaboratorioRemotoFi/labserver

A.2 Código de la práctica “Planos inclinados”

```
(practice.js)

// Importar bibliotecas de conexión con la práctica física o
serial para arduino
var rpio = require("rpio");
// Necesario para poder usar PWM, también el programa debe ser
ejecutado como root
rpio.init({ gpiomem: false });

// Comunicación serial
const { SerialPort } = require("serialport");
const { ReadlineParser } = require("@serialport/parser-readline");

const port = new SerialPort({
  path: "/dev/ttyUSB0", //ttyUSB0 nano ttyACM0 uno
  baudRate: 115200,
```

```
});

const parser = port.pipe(new ReadlineParser({ delimiter:
"\r\n" }));

var serialData = [];
parser.on("data", (data) => {
  serialData.push(data);
});

// MECANICA
const PWM_PIN = 32; // pin salida actuador lineal, señal pwm
const SOL_PIN = 37; // pin salida solenoide
const OPT_PIN = 16; // pin sensor optico
const DIR_PIN = 33; // pin salida actuador lineal, sentido de giro
const VOL_PIN = 35; // pin de lectura de voltaje
const CAR_PIN = 12; // pin sensor optico carro dinamico
const ARR_PIN = 11; // pin de sensor optico para la llegada del
carro dinámico

// Configuración de entradas y salidas
// M3CANICA
rpio.open(PWM_PIN, rpio.PWM);
rpio.open(SOL_PIN, rpio.OUTPUT, rpio.LOW);
rpio.open(DIR_PIN, rpio.OUTPUT, rpio.LOW);
rpio.open(OPT_PIN, rpio.INPUT);
rpio.open(VOL_PIN, rpio.INPUT);
rpio.open(CAR_PIN, rpio.INPUT);
rpio.open(ARR_PIN, rpio.INPUT);

// Configuración de pwm
rpio.pwmSetClockDivider(8);
rpio.pwmSetRange(PWM_PIN, 1024);

class ExamplePractice {
  constructor() {
    this.status = "not ready";

    // estados iniciales
    rpio.pwmSetData(PWM_PIN, 0);
    //    rpio.write(SOL_PIN,rpio.LOW);

    // Variables metadata
    this.rawAngleData = 0; // angulo crudo, mostrado en interfaz
    this.newAngleData = 0; // angulo calibrado, mostrado en
```

```
interfaz

    // Variables de sensores
    this.readOpticalAngle = 0; // lectura sensor óptico de
herradura
    this.readVoltageSensor = 0; // lectura sensor de voltaje
    this.readOpticalDropReset = 0; // lectura sensor ópticomedio,
de salida y retorno del carro dinámico
    this.readOpticalArrival = 0; // lectura sensor óptic de
llegada de carro dinámico

    // Variables toggle
    this.pwmSwapTypeStatus = false; // cambio de señal PW
    this.lockPositionStatus = false; // bloquear posición actual
como horizontal o ángulo cero
    this.lockTargetStatus = false; // bloqueo de angulo de
inclinacion

    // por revisar
    this.endCycleFixAngle = false; // termina el ciclo para
corregir el ángulo
    this.retractSolenoidStatus = false; // retraer solenoide
    this.extendSolenoidStatus = false;

    // Variables de retorno de carro dinámico
    this.restartLaunchTurnLeft = false; // giro a la izquierda
durante el reposicionamiento del carro
    this.restartLaunchTurnRight = false; // giro a la derecha
durante el reposicionamiento del carro
    this.restartLaunchSolenoid = false; // activa el solenoide
durante el reposicionamiento del carro

    // CVariables para corregir el ángulo
    this.fixAngleStatus = false; // si es necesario corregir el
ángulo de lanzamiento
    this.fixAngleSide = false; // si es "false" se mueve a la
izquierda, si es "true" a la derecha
    this.fixAngleAfterDrop = false; // si es corrección después de
un lanzamiento

    // Variables de calibrado
    this.auxRawAngleData = 0; // Angulo crudo, medido desde sensor
    this.lockAngleData = 0; // Angulo calibrado, angulo que marca
0 grados u horizontal
    this.currentAngleData = 0; // Angulo corregido, angulo
```

```
calculado despues de calibrar

    // Variables de retorno de carro
    this.targetAngleData = 777; // Angulo Objetivo, angulo de
inclinación inicial
    this.fixSide = 0;

    // Variables para ciclos
    this.fixAngle = null; // Ciclo de correccion de angulo

    // Mensajes
    this.mssgRightLimit = false;
    this.mssgLeftLimit = false;
    this.mssgSolBtn = false;
    this.mssgResBtn = false;
    this.mssgTargetAngle = false;

    // Bloqueo de botones
    this.btnSolInUse = false;
    this.btnResInUse = false;

    // lectura de variables y logica, similar al loop de arduino
    setInterval(() => {
      // Agregar verificacion de gpio y acciones acordes

      // Lectura de sensores
      // Sensor óptico de herradura
      this.readOpticalAngle = this.readOpticalAngleFunction();
      // Sensor óptico medio
      this.readOpticalDropReset =
this.readOpticalDropResetFunction();
      // Sensor óptico llegada
      this.readOpticalArrival = this.readOpticalArrivalFunction();

      // Lectura de datos sensor magnético
      while (serialData.length > 0) {
        if (this.lockPositionStatus === false) {
          this.auxRawAngleData = Number(serialData.shift());
          this.rawAngleData = -1 * (this.auxRawAngleData * 1);
        } else if (this.lockPositionStatus === true) {
          this.lockAngleData = this.auxRawAngleData * 1;
          this.currentAngleData = Number(serialData.shift());
          this.newAngleData =
            -1 * (this.lockAngleData - this.currentAngleData * 1);
          // console.log(this.newAngleData);
```

```
        }
      }

      // Funcion de retorno, giro a la derecha y cambio
      if (this.restartLaunchTurnRight === true) {
        if (this.newAngleData <= -4) {
          rpio.pwmSetData(PWM_PIN, 0);
          this.restartLaunchTurnRight = false;

          // Funcion de retorno, giro a la izquierda
          setTimeout(() => {
            this.restartLaunchTurnLeft = true;
            rpio.write(DIR_PIN, rpio.LOW);
            rpio.pwmSetData(PWM_PIN, 384); // 384
          }, 4000);
        }
      }

      // Funcion de retorno, giro a la izquierda
      if (this.restartLaunchTurnLeft === true) {
        if (this.newAngleData >= this.targetAngleData) {
          rpio.pwmSetData(PWM_PIN, 0);
          this.restartLaunchTurnLeft = false;

          // Corrección de ángulo, después del retorno
          setTimeout(() => {
            if (this.newAngleData != this.targetAngleData) {
              this.fixAngleStatus = true;
              if (this.newAngleData > this.targetAngleData) {
                this.fixAngleFunction(true);
              } else if (this.newAngleData < this.targetAngleData)
{
                this.fixAngleFunction(false);
              }
            } else {
              this.btnResInUse = false;
            }
          }, 2000);
        }
      }

      // Corrigiendo angulo
      if (this.fixAngleStatus === true) {
        if (this.newAngleData == this.targetAngleData) {
          clearInterval(this.fixAngle);
```

```
        // Un giro extra de la suerte
        console.log("giro extra");
        rpio.write(DIR_PIN, rpio.HIGH);
        rpio.pwmSetData(PWM_PIN, 200);
        setTimeout(() => {
          rpio.pwmSetData(PWM_PIN, 0);
        }, 200);

        this.fixAngleStatus = false;
        this.btnResInUse = false;
      }
    }

    // Activación dl solenoide durante el retorno
    if (
      this.readOpticalDropReset === 0 &&
      this.restartLaunchSolenoid == true
    ) {
      this.retractSolenoidArrivalFunction(3000);
      this.restartLaunchSolenoid = false;
    }

    // Desactivacion del solenoide durante el lanzamiento
    if (
      this.readOpticalDropReset === 0 &&
      this.retractSolenoidStatus === true
    ) {
      rpio.write(SOL_PIN, rpio.LOW);
      this.retractSolenoidStatus = false;
    }
  }, 10);
}

// Funciones M3CANICA

// Funcion de reposicionamiento de carro
// Giro a la derecha
restartLaunchFunction() {
  this.btnResInUse = true; // Boton en uso, se activa bloqueo
  this.restartLaunchTurnRight = true;
  this.restartLaunchSolenoid = true;

  setTimeout(() => {
    rpio.write(DIR_PIN, rpio.HIGH);
```

```
      rpio.pwmSetData(PWM_PIN, 384); // 256
    }, 1000);
  }

  // Funciones para ajustar el angulo
  fixAngleFunction(side) {
    if (side) {
      console.log("giro a la derecha");
      rpio.write(DIR_PIN, rpio.HIGH);
    } else if (!side) {
      console.log("giro a la izquierda");
      rpio.write(DIR_PIN, rpio.LOW);
    }
    this.fixAngle = setInterval(this.turnRightStepByStepFunction,
300);
  }

  turnRightStepByStepFunction() {
    rpio.pwmSetData(PWM_PIN, 256); //256
    setTimeout(() => {
      rpio.pwmSetData(PWM_PIN, 0);
    }, 100);
  }

  // Funcion de ayuda, eliminar al eliminar boton de desactivar
los dispositivos
  stopAllCyclesFunction() {
    clearInterval(this.fixAngle);
  }

  // Lectura de  sensores
  // Sensor optico de herradura
  readOpticalAngleFunction() {
    return rpio.read(OPT_PIN);
  }

  // Sensor optico de paso, medio
  // Se activa durante el paso del carro dinamico
  readOpticalDropResetFunction() {
    return rpio.read(CAR_PIN);
  }

  // Sensor optico de paso, llegada
  // Se activa cuando el carro dinamico llega a la posicion
inicial
```

```
readOpticalArrivalFunction() {
  return rpio.read(ARR_PIN);
}

// Giro a la izquierda
turnLeftPositionFunction() {
  if (this.readOpticalAngle == 1) {
    rpio.write(DIR_PIN, rpio.LOW);
    if (this.pwmSwapTypeStatus === true) {
      rpio.pwmSetData(PWM_PIN, 384); // 256
      setTimeout(() => {
        rpio.pwmSetData(PWM_PIN, 0);
      }, 250);
    } else if (this.pwmSwapTypeStatus === false) {
      rpio.pwmSetData(PWM_PIN, 512);
      setTimeout(() => {
        rpio.pwmSetData(PWM_PIN, 0);
      }, 750); //500
    }
  } else {
    // Lanzar alerta
    this.mssgLeftLimit = true;
    rpio.write(DIR_PIN, rpio.HIGH);
    rpio.pwmSetData(PWM_PIN, 384); // 256
    setTimeout(() => {
      rpio.pwmSetData(PWM_PIN, 0);
    }, 750);
  }
}

// Giro a la derecha
turnRightPositionFunction() {
  if (this.readOpticalAngle == 1) {
    rpio.write(DIR_PIN, rpio.HIGH);
    if (this.pwmSwapTypeStatus === true) {
      rpio.pwmSetData(PWM_PIN, 384); // 256
      setTimeout(() => {
        rpio.pwmSetData(PWM_PIN, 0);
      }, 250);
    } else if (this.pwmSwapTypeStatus === false) {
      rpio.pwmSetData(PWM_PIN, 512);
      setTimeout(() => {
        rpio.pwmSetData(PWM_PIN, 0);
      }, 750); //500
    }
```

```
    } else {
      // Lanzar alerta
      this.mssgRightLimit = true;
      rpio.write(DIR_PIN, rpio.LOW);
      rpio.pwmSetData(PWM_PIN, 384); // 256
      setTimeout(() => {
        rpio.pwmSetData(PWM_PIN, 0);
      }, 750);
    }
}

// Funcion para energizar el solenoide, para lanzamientos
retractSolenoidDropFunction(delay) {
  rpio.write(SOL_PIN, rpio.HIGH);
  this.retractSolenoidStatus = true;
  setTimeout(() => {
    rpio.write(SOL_PIN, rpio.LOW);
  }, delay);
}

// Funcion para energizar el solenoide, para retornos
retractSolenoidArrivalFunction(delay) {
  rpio.write(SOL_PIN, rpio.HIGH);
  setTimeout(() => {
    if (this.readOpticalArrival == 0) {
      rpio.write(SOL_PIN, rpio.LOW);
    } else {
      setTimeout(() => {
        rpio.write(SOL_PIN, rpio.LOW);
      }, 1000);
    }
  }, delay);
}

// Guarda el ángulo actual como el ángulo objetivo
lockTargetAngleFunction() {
  this.targetAngleData = this.newAngleData;
  if (this.targetAngleData === 777) {
    this.mssgTargetAngle = true;
  }
  console.log("Angulo guardado", this.targetAngleData);
}

// Funcion de ayuda y pruebas
turnEverythingOffFunction() {
```

```
    rpio.write(SOL_PIN, rpio.LOW);
    rpio.pwmSetData(PWM_PIN, 0);
    this.stopAllCyclesFunction();
  }

  // Se definen las acciones para cada comando
  command(command, value) {
    if (this.status === "initializing") {
      return {
        status: "error",
        message:
          "Regresando a valores iniciales, espera a que esté lista
la práctica",
      };
    }

    if (this.mssgRightLimit === true) {
      this.mssgRightLimit = false;
      return {
        status: "error",
        message: "L mite derecho alcanzado",
      };
    }

    if (this.mssgLeftLimit === true) {
      this.mssgLeftLimit = false;
      return {
        status: "error",
        message: "L mite izquierdo alcanzado",
      };
    }

    if (this.mssgResBtn === true) {
      this.mssgResBtn = false;
      return {
        status: "error",
        message: "Operacion en desarrollo",
      };
    }

    if (this.mssgSolBtn === true) {
      this.mssgSolBtn = false;
      return {
        status: "error",
        message: "Opción no disponible durante reposicionamiento",
```

```
      };
    }

    if (this.mssgTargetAngle === true) {
      this.mssgTargetAngle = false;
      return {
        status: "error",
        message: "Error al fijar el ángulo de inclinación, intenta
de nuevo",
      };
    }

    // Agregar acciones
    switch (command) {
      case "pwmSwap":
        this.pwmSwapTypeStatus = value;
        break;
      case "turnLeft":
        this.turnLeftPositionFunction();
        break;
      case "turnRight":
        this.turnRightPositionFunction();
        break;
      case "lockPosition":
        this.lockPositionStatus = value;
        break;
      case "powerSolenoid":
        if (this.btnResInUse === false) {
          this.retractSolenoidDropFunction(1500);
        } else if (this.btnResInUse === true) {
          this.mssgSolBtn = true;
        }
        break;
      case "resetPosition":
        if (this.btnResInUse === false) {
          this.restartLaunchFunction();
        } else if (this.btnResInUse === true) {
          this.mssgResBtn = true;
        }
        break;
      case "turnOff":
        this.turnEverythingOffFunction();
        break;
      case "lockTarget":
        this.lockTargetStatus = value;
```

```
        if (this.lockTargetStatus === true) {
          this.lockTargetAngleFunction();
        }
        break;
      default:
        return {
          status: "error",
          message: `No se reconoce el comando ${command}`,
        };
    }

    return { status: "success" };
  }

  // Se ejecuta cuando la interfaz se conecta por primera vez
  init() {
    // Checar el estatus de las cosas y si no está en el 'estado
inicial' cambio el
    // estado de la práctica a 'initializing' y mando los comandos
para que esté en
    // el estado inicial

    // Estados inciales

    this.status = "ready";
  }
}

export default ExamplePractice;
```

REFERENCIAS

[1] De Jong, T., Linn, M. C. & Zacharia, Z. C. “Physical and virtual laboratories in science and engineering education”. Science, 340, pp. 305–308. 2013. Disponible en junio de 2023 de https://www.science.org/doi/10.1126/science.1230579.

[2] Wyman, B. (2022). *Definición de un laboratorio escolar.* Website Filosofía.co. Disponible en junio de 2023 de Definicion de un laboratorio escolar - Filosofía (filosofia.co).

[3] Singer, S. R., Hilton, M. L. & Schweingruber, H. A. “America’s Lab Report. Investigations in High School Science”. The National Academies Press, pp. 1–11. 2006. Retrieved from http://elibrary.pcu.edu.ph:9000/digi/NA02/2005/11311.pdf.

[4] Barberá, O. & Valdés, P. “El trabajo práctico en la enseñanza de las ciencias: una revisión”. Enseñanza de las ciencias, 14 (3), pp. 365–379. 1996. Retrieved from https://ddd.uab.cat/pub/edlc/02124521v14n3/02124521v14n3p365.pdf.

[5] Feisel, L. D. & Rosa, A. J. “The Role of the Laboratory in Undergraduate Engineering Education”. Journal of Engineering Education, 94, pp. 121–130. 2005. Retrieved from “https://onlinelibrary.wiley.com/doi/epdf/10.1002/j.2168-9830.2005.tb00833.x.

[6] Gutiérrez–Mosquera, A. & Barajas–Perea, D. S. “Uso de productos cotidianos en las prácticas de laboratorio de química orgánica: una estrategia metodológica basada en la investigación dirigida” (Use of Daily Products for Organic Chemistry Laboratory Practices: A Methodological Strategy Base don Directed Research). Revista científica 44 (2), pp. 189–201. 2022. DOI: https://doi.org/10.14483/23448350.18616.

[7] Minami–Koyama, Y., Peñuelas–Rivas, U. M., Savage–Carmona, J. y Castañeda–Cedeño, S. “Laboratorio remoto accedido por internet para la asignatura Cinemática

y Dinámica", artículo publicado en el III Simposio la investigación y desarrollo en la Facultad de Ingeniería". UNAM, Facultad de Ingeniería, noviembre de 2007.

[8] Savage–Carmona, J. y Minami–Koyama, Y. "Creación de un laboratorio remoto accedido por medio de la internet para la asignatura Cinemática y Dinámica". Protocolo del proyecto UNAM-DGAPA-PAPIME EN106204. 2003.

[9] Minami–Koyama, Y. y Arenas–González, A. "Diseño de prácticas de laboratorio para fortalecer el aprendizaje de conceptos matemáticos en ciencias básicas". Protocolo del proyecto UNAM–DGAPA–PAPIME PE111218. 2017.

[10] Minami–Koyama, Y. y Gámez–Leal, R. "Creación de material didáctico y dispositivos para la implementación de prácticas experimentales a distancia en la División de Ciencias Básicas". Protocolo del proyecto UNAM–DGAPA–PAPIME PE109021. 2020.

[11] Minami–Koyama, Y. y Serrano–Miranda, H. G. "Diseño de dispositivos mecatrónicos para apoyo en emergencias". Protocolo del proyecto UNAM–DGAPA–PAPIIT IT102121. 2020.