Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

LABORATORIO DE CONTROL DE UN MOTOR INTEGRADO EN LMS

Alumno: Lucena Campos, Jesús
Tutor: Prof. D. Ildefonso Ruano Ruano
Cotutor: Prof. D. Elísabet Estévez Estévez
Dpto.: Ingeniería de Telecomunicación

Junio, 2022

Universidad de Jaén
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado
Curso 2021-2022

Don ILDEFONSO RUANO RUANO y Doña ELÍSABET ESTEVEZ ESTEVEZ, tutores del Proyecto Fin de Carrera titulado: LABORATORIO DE CONTROL DE UN MOTOR INTEGRADO EN LMS, que presenta JESÚS LUCENA CAMPOS, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Linares.

Linares, Junio de 2022

El alumno: Los tutores:

RUANO RUANO ILDEFONSO - 02889873S
Firmado digitalmente por RUANO RUANO ILDEFONSO - 02889873S
Fecha: 2022.06.20 09:35:00 +02'00'

Jesús Lucena Campos
DNI: 26970742E

Ildefonso Ruano Ruano

Elísabet Estévez Estévez

ÍNDICE DE CONTENIDOS

Índice de Contenidos i
Índice de Figuras ii
Índice de Tablas iii

1. INTRODUCCIÓN 1
2. OBJETIVOS 5
3. TECNOLOGÍAS UTILIZADAS 6
   3.1 HARDWARE 6
      3.1.1 Raspberry Pi 6
      3.1.2 Motor 10
      3.1.3 Módulo Puente H 12
      3.1.4 Panel iluminación 14
      3.1.5 Relé 15
      3.1.6 Cámara IP 16
      3.1.7 Fuente Alimentación 17
   3.2 SOFTWARE 18
      3.2.1 Remote Interoperability Protocol 19
      3.2.2 Easy JavaScript Simulations 24
      3.2.3 Sharable Content Object Reference Model 27
4. HERRAMIENTAS Y MÉTODOS 28
   4.1 SERVIDOR RIP 28
   4.2 CLIENTE 36
   4.3 SERVIDOR DE IMÁGENES 40
   4.4 SISTEMA DE CONTROL 42
   4.5 INTEGRACIÓN LMS 45
5. RESULTADOS 48
   5.1 IDENTIFICACIÓN DE LA DINÁMICA DEL SISTEMA 48
   5.2 CONTROL DE LA POSICIÓN DEL MOTOR 50
6. PRESUPUESTO 52
7. CONCLUSIONES Y TRABAJOS FUTUROS 53
8. ANEXOS 54
   8.1 MANUAL INSTALACIÓN Y CONFIGURACIÓN 54
   8.2 SEGURIDAD COMUNICACIONES 59
      8.2.1 Configuración segura del servidor RIP 59
      8.2.2 Configuración segura de la cámara IP 61
   8.3 CONFIGURACIÓN EXTERNA UJA 63
   8.4 ABREVIATURAS Y ACRÓNIMOS 65
9. REFERENCIAS BIBLIOGRÁFICAS 66

ÍNDICE DE FIGURAS

Figura 1-1.- El protocolo RIP es utilizado por un cliente RIP y un servidor RIP. 1
Figura 1-2.- Esquema general del sistema. 2
Figura 1-3.- Laboratorio de Control Procesos de la EPS de Jaén 3
Figura 1-4.- Armario laboratorio remotos (cerrado) 3
Figura 1-5.- Armario laboratorio remotos (abierto) 3
Figura 1-6.- Montaje laboratorio remoto 4
Figura 3-1.- Raspberry Pi 4 modelo B 7
Figura 3-2.- Componentes Raspberry Pi 4 modelo B 8
Figura 3-3.- Conexionado GPIO Raspberry PI 4 8
Figura 3-4.- Logotipo Raspberry Pi OS 9
Figura 3-5.- Escritorio Raspberry PI OS 9
Figura 3-6.- Motor POLOLU 10
Figura 3-7.- Engranajes interiores Motor 10
Figura 3-8.- Encoder 11
Figura 3-9.- Comparativa canales Encoder 11
Figura 3-10.- Controlador de Motor Dual 12
Figura 3-11.- Panel Led 14
Figura 3-12.- Modulo 4 Relés 15
Figura 3-13.- Cámara IP 16
Figura 3-14.- Fuente alimentación 17
Figura 3-15.- Esquema general comunicaciones 18
Figura 3-16.- RIP se basa en HTTP (POST y GET) y en el formato JSON-RPC. 19
Figura 3-17.- Representación de la correspondencia entre funciones y métodos. 20
Figura 3-18.- Vista arquitectónica de un servidor RIP 22
Figura 3-19.- Funciones internas y externas en clientes y servidores RIP 23
Figura 3-20.- Panel Modelo en EJS 25
Figura 3-21.- Panel Vista en EJS 26
Figura 4-1.- Conexiones sistema con pines GPIO 28
Figura 4-2.- Prueba de la respuesta del servidor RIP 29
Figura 4-3.- Árbol archivos servidor WEB 30
Figura 4-4.- Código archivo App.py 31
Figura 4-5.- Código archivo AppConfig.py 31
Figura 4-6.- Función SSE protocolo RIP. 32
Figura 4-7.- Función lectura pulsos encoders. 33
Figura 4-8.- Estados Encoder 33
Figura 4-9.- Función calculo velocidad y PID. 34
Figura 4-10.- Contenido archivo cert.pem 35
Figura 4-11.- Contenido archivo privkey.pem 35
Figura 4-12.- Elemento rip en EjsS 36
Figura 4-13.- Configuración rip en EjsS 37
Figura 4-14.- Configuración variables rip en EjsS 37
Figura 4-15.- Laboratorio remoto para el Control en posición del motor 38
Figura 4-16.- Página login DCS-4633EV 40
Figura 4-17.- Panel configuración video 41
Figura 4-18.- Sistema de Control en bucle abierto 42
Figura 4-19.- Sistema de Control en bucle cerrado 42
Figura 4-20.- Creación paquete SCORM 45
Figura 4-21.- Opciones paquete SCORM 45

Figura 4-22.- Activar edición SCORM 46
Figura 4-23.- Añadir actividad SCORM 46
Figura 4-24.- Selección paquete SCORM 46
Figura 4-25.- Creación y configuración paquete SCORM. 47
Figura 5-1.- Esquema interno para la identificación de la dinámica del motor. 48
Figura 5-2.- Práctica 1: Identificación de la dinámica del sistema 49
Figura 5-3.- Esquema interno para el control del sistema con un PID. 50
Figura 5-4.- Práctica 2: Control en posición del motor 51
Figura 8-1.- Archivo /etc/dhcpcd.conf 54
Figura 8-2.- Archivo /etc/rc.local 56
Figura 8-3.- Archivo inicio.sh 56
Figura 8-4.- Asistente D-Link 57
Figura 8-5.- Pantalla login D-Link 57
Figura 8-6.- Direccionamiento IP D-Link 57
Figura 8-7.- Generación certificado SSL autofirmado 59
Figura 8-8.- Código fuente archivo AppConfig.py 60
Figura 8-9.- Panel configuración HTTPS 61
Figura 8-10.- Certificado Cámara IP 61
Figura 8-11.- Advertencia Navegador conexión no segura 62
Figura 8-12.- Código de error navegador 62
Figura 8-13.- Terminal de Windows. 63
Figura 8-14.- Página Login router HUAWEI EG8245H 63
Figura 8-15.- Reglas habilitadas en Router 64
Figura 8-16.- Configuración regla 64

ÍNDICE DE TABLAS

Tabla 3-1.- Tabla comparativa modelos Raspberry PI. 7
Tabla 3-2.- Funciones cables Encoder 11
Tabla 3-3.- Tabla verdad L298 13
Tabla 3-4.- Correspondencia entre las funciones externas de RIP, los métodos HTTP y los puntos finales del servidor web de RIP 21
Tabla 4-1.- Tabla variables escritura ('writables' - SET). 32
Tabla 4-2.- Tabla variables lectura (’readables’ - GET). 32
Tabla 6-1.- Presupuesto materiales 52
Tabla 6-2.- Presupuesto mano de obra 52
Tabla 6-3.- Presupuesto total 52

Agradecimientos:

En primer lugar, quiero agradecer a Elísabet Estévez y Alonso Ruano,
tutores de este trabajo fin de grado, ya que sin su ayuda y apoyo no hubiera sido posible.

También mostrar mi gratitud infinita hacia mis padres,
por todo el esfuerzo, paciencia y confianza depositados en mí.

A Inma, mi compañera de vida, ya que gran parte de este TFG se lo debo a ella,
que me ha ayudado a poder invertir todo el tiempo que he necesitado.

Pero sobre todo a mi hijo Hugo, por el cual encontraba la motivación para seguir.
Y para el cual espero sirva de ejemplo.

RESUMEN

El presente Trabajo Fin de Grado (TFG) tiene como objetivo el diseño y creación de un laboratorio remoto docente para el control de un motor apoyado en la metodología del proyecto UNILabs (University Network of Interactive Laboratories). Las comunicaciones cliente-servidor están basadas en el protocolo RIP (Remote Interoperability Protocol) para comunicar desde el cliente web las acciones requeridas. Para el desarrollo de los clientes se ha utilizado la herramienta Easy Java/JavaScript Simulations (EJsS), gracias al cual, se ha desarrollado un entorno gráfico intuitivo para el usuario final, donde se pueden observar resultados gráficos en tiempo real conforme se cambian valores de control del sistema. Adicionalmente, los programas cliente realizan otra conexión paralela con una cámara IP para poder ofrecer en la interfaz gráfica del estudiante imágenes de la planta.

Se ha desarrollado una planta física que consiste principalmente en un motor de corriente continua y un sistema de iluminación del mismo, donde se ha trabajado en el diseño y montaje del sistema físico y el servidor, así como su conexionado y programación de comunicaciones con los clientes. Se ha utilizado una Raspberry Pi (Single Board Computer) para configurar el programa servidor y de control, por su parte los alumnos usarán en cliente web creado con EjsS para conectarse remotamente. También se han propuesto unas prácticas enfocadas a alumnos de asignaturas de control donde en primer lugar podrán trabajar la identificación del sistema (en este caso un motor de corriente continua) y posteriormente aplicar sus conocimientos para la sintonía de un controlador en posición del mismo.

Adicionalmente, se ha estudiado el protocolo SCORM (Shareable Content Object Reference Model) y se han integrado los programas cliente como módulos SCORM que pueden ser usados en plataformas de aprendizaje o LMS (Learning Management System). Estos módulos han sido probados en PLATEA, el LMS institucional de la Universidad de Jaén.

ABSTRACT

The objective of this Final Degree Project (TFG) is the design and creation of a remote teaching laboratory for the control of a motor supported by the methodology of the UNILabs project (University Network of Interactive Laboratories). Client-server communications are based on the RIP protocol (Remote Interoperability Protocol) to communicate the required actions from the web client. For the development of the clients, the Easy Java/JavaScript Simulations (EJsS) tool has been used, thanks to which an intuitive graphical environment has been developed for the end user, where graphical results can be observed in real time as values are changed. system control. Additionally, the client programs make another parallel connection with an IP camera to be able to offer images of the plant in the student's graphical interface.

A physical plant has been developed that consists mainly of a DC motor and its lighting system, where work has been done on the design and assembly of the physical system and the server, as well as its connection and communication programming with clients. A Raspberry Pi (Single Board Computer) has been used to configure the server and control program, while the students will use a web client created with EjsS to connect remotely. Some practices have also been proposed focused on students of control subjects where they will first be able to work on the identification of the system (in this case a direct current motor) and later apply their knowledge to tune a controller in its position.

Additionally, the SCORM (Shareable Content Object Reference Model) protocol has been studied and client programs have been integrated as SCORM modules that can be used in learning platforms or LMS (Learning Management System). These modules have been tested in PLATEA, the institutional LMS of the University of Jaén.

1. INTRODUCCIÓN

En la gran mayoría de los estudios STEM (Science, Technology, Engineering and Mathematics) el trabajo práctico es indispensable, para ello en todos los planes de estudio de estos grados se incluyen prácticas de laboratorio que permiten a los alumnos experimentar con una realidad que va más allá de la teoría pura. Desde hace ya bastantes años se han venido utilizando para este fin Laboratorios Online (OL), ya sea Virtuales (VL) o Remotos (RL) [1]. El surgimiento de este tipo de laboratorio se debe a que los experimentos en plantas reales suelen ser costosos en términos de tiempo, costo económico y energía. Además, estas infraestructuras suelen estar infrautilizadas debido al escaso tiempo del que disponen los estudiantes. Estas nuevas tecnologías de la información y la comunicación, como los laboratorios virtuales, han revolucionado y actualizado los métodos de enseñanza al facilitar la comprensión de los alumnos.[2].

Para trabajar con los laboratorios online se hace necesario el uso de protocolos que permitan las comunicaciones de los usuarios con los mismos. En este sentido uno de estos protocolos es el protocolo RIP (Remote Interoperability Protocol) desarrollado por un equipo de investigadores de la UNED (Universidad Nacional de Educación a Distancia). El objetivo de RIP es proporcionar una solución de comunicación simple y potente que puedan utilizar los clientes web. Por lo tanto, RIP usa solo el protocolo HTTP estándar puro compatible con todos los principales navegadores web.

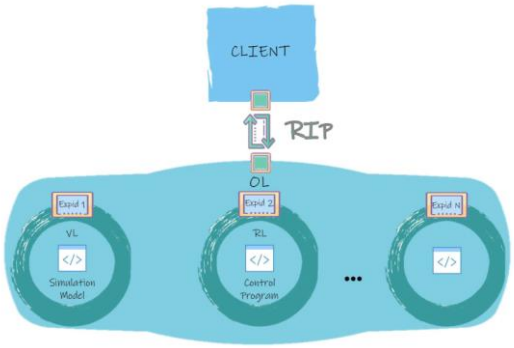

Figura 1-1.- El protocolo RIP es utilizado por un cliente RIP y un servidor RIP.

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

RIP está diseñado para comunicar clientes web con OL. Cuando se utiliza para comunicarse con una VL [3], RIP expone metadatos y métodos y variables de entrada y salida relacionados con un modelo de simulación que está alojado y se ejecuta en una computadora (generalmente, un servidor remoto). Cuando se utiliza para comunicarse con un RL, RIP hace lo mismo con un programa de control definido en una computadora (generalmente, un servidor remoto) para monitorear y manipular el equipo de laboratorio.

Este TFG presenta un trabajo que ha consistido en la creación de un laboratorio remoto para permitir el trabajo práctico de los estudiantes con un motor real a través de Internet. El trabajo de los alumnos consiste en el cálculo de los parámetros de control remoto del motor que permita realizar los ejercicios propuestos en cada práctica. Se ha seguido la metodología de trabajo propuesta en el proyecto UNILabs (University Network of Interactive Laboratories) [4], cuyo objetivo es construir una red interuniversitaria de laboratorios online interactivos con la participación de investigadores de la Universidad de Jaén.

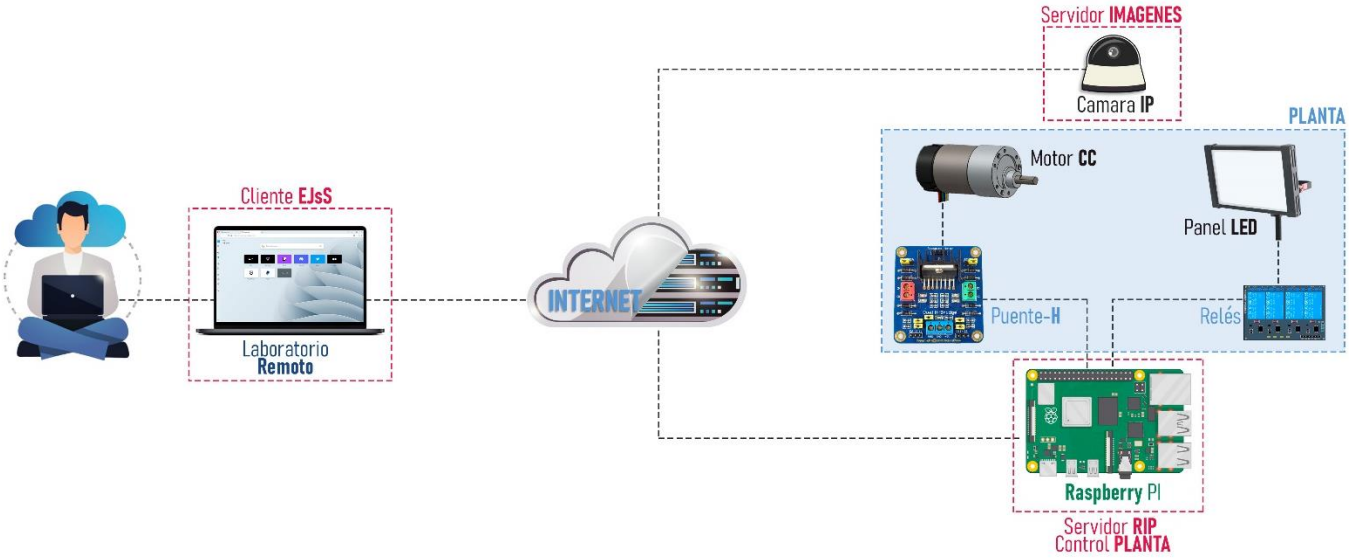

Figura 1-2.- Esquema general del sistema.

Los componentes empleados han sido los siguientes:

- Clientes: Programas JavaScript desarrollados con el software Easy Java/JavaScript Simulations (EJsS) [5] que incluyen soporte de cliente RIP y visión del sistema remoto. Se ejecutan en el equipo del estudiante desde cualquier navegador. Hay 2 tipos de versiones, unos pueden ser usados de forma independiente u otros embebidos en un paquete SCORM que se obtiene desde el LMS (Learning Management System) institucional (PLATEA en la Universidad de Jaén).
- Servidor: Está formado por un microcontrolador de bajo coste y mediante un software desarrollado en Python se encargará de comunicaciones RIP y el control de la planta.
- Planta: Compuesta por varios elementos, entre ellos el motor CC que estará controlado mediante un puente en H, además se ha incorporado un sistema de iluminación mediante un panel Led que estará controlado por un módulo de relés.
- Servidor imágenes: Está formado por una cámara IP, la cual está mostrando imágenes en todo momento mientras se realizan las prácticas en desde el laboratorio remoto.

Tras el desarrollo de este TFG en un domicilio particular, se realizó su montaje y configuración en la Universidad de Jaén, concretamente en el Laboratorio de Control de Procesos (A3-467), donde será su ubicación definitiva para su posterior utilización por los estudiantes que se conectarán remotamente al laboratorio.

Figura 1-3.- Laboratorio de Control Procesos de la EPS de Jaén

Se ha instalado en un armario protegido en la dependencia A3-467, el cual fue creado para albergar los distintos laboratorios remotos creados por el área de Ingeniería de Sistemas y Automática de la Universidad de Jaén.

Figura 1-4.- Armario laboratorio remotos (cerrado)

Figura 1-5.- Armario laboratorio remotos (abierto)

En la siguiente imagen se muestra el resultado final del laboratorio remoto, el cual se ha montado sobre una lámina de PVC de reducidas dimensiones donde se han distribuido todos los componentes utilizados en la planta además de la Raspberry Pi con el servidor. Gracias a dicho montaje ha facilitado su traslado y puesta en marcha una vez instalado en la Universidad de Jaén.

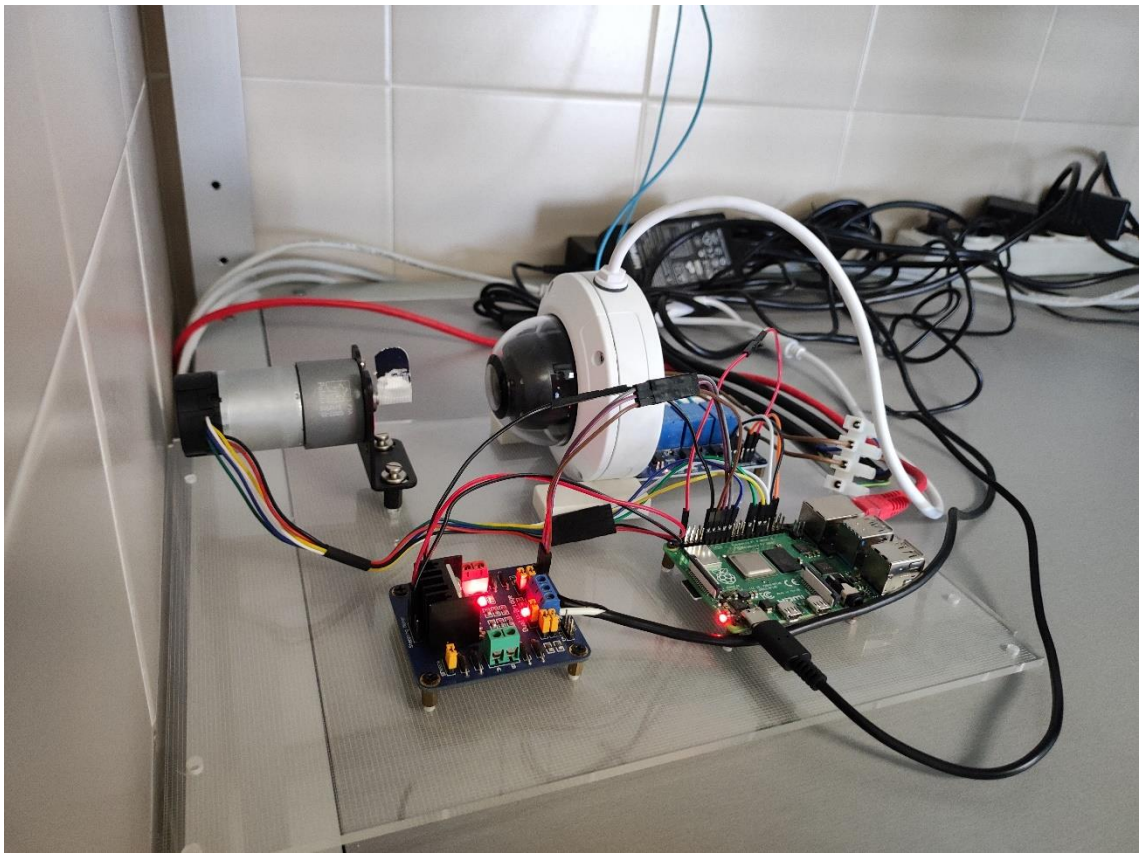

Figura 1-6.- Montaje laboratorio remoto

2. OBJETIVOS

El objetivo principal de este TFG es el diseño y construcción de un laboratorio online para permitir tanto el trabajo práctico de los estudiantes con un motor real a través del laboratorio remoto.

Como objetivos específicos necesarios para conseguir el objetivo principal se han definido:

- Diseño y construcción de un Laboratorio Online Remoto para el control eficaz y preciso de un sistema compuesto por Motor como uso didáctico.
- Estudio de metodología UNILabs.
- Diseño e implementación de prácticas de laboratorio para ser utilizada en el laboratorio online.
- Realizar pruebas del diseño final del laboratorio, con el fin de probar la compatibilidad y conectividad entre sus componentes.
- Estudio y uso de SCORM para el desarrollo de paquetes SCORM como soporte del software cliente en LMS, de forma que los alumnos puedan realizar sus prácticas de forma controlada.

3. TECNOLOGÍAS UTILIZADAS

Durante este apartado se explicarán las características de los elementos hardware y software utilizados para el desarrollo de un laboratorio remoto. Por un lado, el servidor y controlador que estarán implementados en la Raspberry Pi descrita en el punto 3.1.1. Por otro lado, el cliente o front-end del laboratorio, implementando mediante JavaScript y HTML5 utilizando EJsS para acceder desde cualquier dispositivo. Adicionalmente, se han creado versiones SCORM de los clientes que permiten ofrecer el laboratorio remoto a los alumnos de forma controlada (monitorizando accesos y resultados) en el entorno de trabajo natural, como es el LMS institucional.

3.1 HARDWARE

3.1.1 Raspberry Pi

Se engloba dentro de los ordenadores denominados de placa reducida de bajo coste (SBC) destinado principalmente para la formación de informática en centros educativos, desarrollada por Raspberry Pi Foundation con la idea de promover y enseñar electrónica en las escuelas de Reino Unido.

Está basado en hardware libre y normalmente se usan también software de código abierto basado en GNU/Linux, el SO oficial desarrollado para ello es Raspberry Pi OS (antes llamado Raspbian) que es una adaptación de Debian, aunque también es posible instalar otros SO como Windows 10 incluido.

Todas las versiones de placas Raspberry Pi están basadas en un SoC de arquitectura ARM con un procesador Broadcom de bajo consumo, incluyendo memoria RAM que variara su capacidad en función del modelo. Disponen de GPU, varios puertos USB, lector de tarjetas SD donde se instalará el SO, salidas de video, puertos Ethernet. En los últimos modelos la alimentación incluye un conector USB-C además de dos conectores MIPI, para una cámara y un display. Además de una tarjeta Wifi y Bluetooth integrada [6].

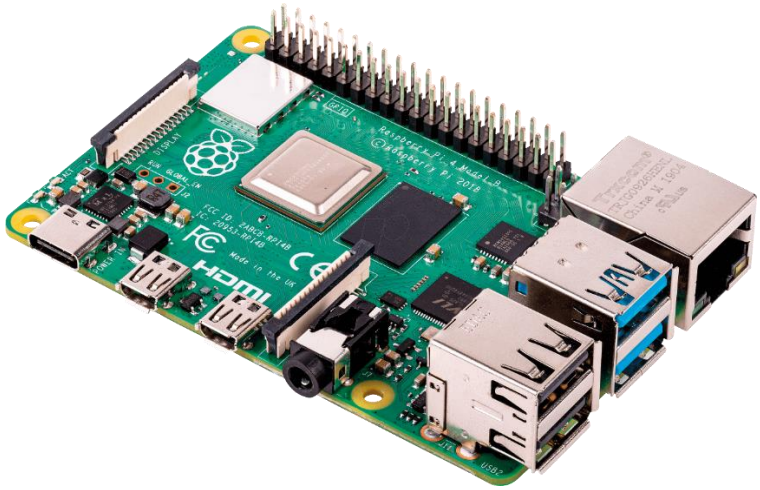

Figura 3-1.- Raspberry Pi 4 modelo B

Fuente: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Actualmente en el mercado se encuentran varios modelos. A continuación, se muestra en la Tabla 3-1 las características de las distintas versiones y en donde se aprecia la evolución de dicha computadora [7].

	SoC	CPU	GPU	RAM	USB	V/A	Boot	Red	Alimentación	Tamaño	Fecha	Precio
Model A	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	256MB	1	RCA Jack HDMI	SD	No	300mA 1,5w / 5v MicroUSB GPIO	85,6 x 53,98 mm	04/12	25$
Model A+	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	256MB	1	Jack HDMI	uSD	No	400mA 2w / 5v MicroUSB GPIO	65 x 56 mm	11/14	20$
3 Model A+	Broadcom BCM2837B0	1,4GHz QUAD ARM Cortex-A53	VideoCore IV	512MB	1	Jack HDMI	uSD	Dual-band WiFi, BT	2,5A 12,5w / 5v MicroUSB GPIO	65 x 56 mm	11/18	20$
Model B	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	512MB	2	RCA Jack HDMI	SD	ETH 10/100	700mA 3,5w / 5v MicroUSB GPIO	85,6 x 53,98 mm	04/12	35$
Model B+	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	512MB	4	Jack HDMI	uSD	ETH 10/100	500mA 2,5w / 5v MicroUSB GPIO	85 x 56 mm	07/14	35$
2 Model B	Broadcom BCM2836	900MHz QUAD ARM Cortex-A7	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100	800mA 4w / 5v MicroUSB GPIO	85 x 56 mm	02/15	35$
3 Model B	Broadcom BCM2837	1,2GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100 WiFi, BT	2,5A 12,5w / 5v MicroUSB GPIO	85 x 56 mm	02/16	35$
3 Model B+	Broadcom BCM2837B0	1,4GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100/300 (USB) Dual-band WiFi BT	2,5A 12,5w / 5v MicroUSB GPIO PoE (HAT)	85 x 56 mm	03/18	35$
4 Model B	Broadcom BCM2711	1,5GHz QUAD ARM Cortex-A72	VideoCore IV	1, 2 o 4GB	2 (2.0) 2 (3.0)	Jack 2 micro HDMI	uSD	ETH 1000 Dual-band WiFi BT	2,5A 12,5w / 5v USB-C GPIO PoE (HAT)	85 x 56 mm	06/19	35$
Zero	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	No	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	11/15	5$
Zero W	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	Wifi, BT	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	02/17	10$

Tabla 3-1.- Tabla comparativa modelos Raspberry PI.

Fuente: https://www.comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/

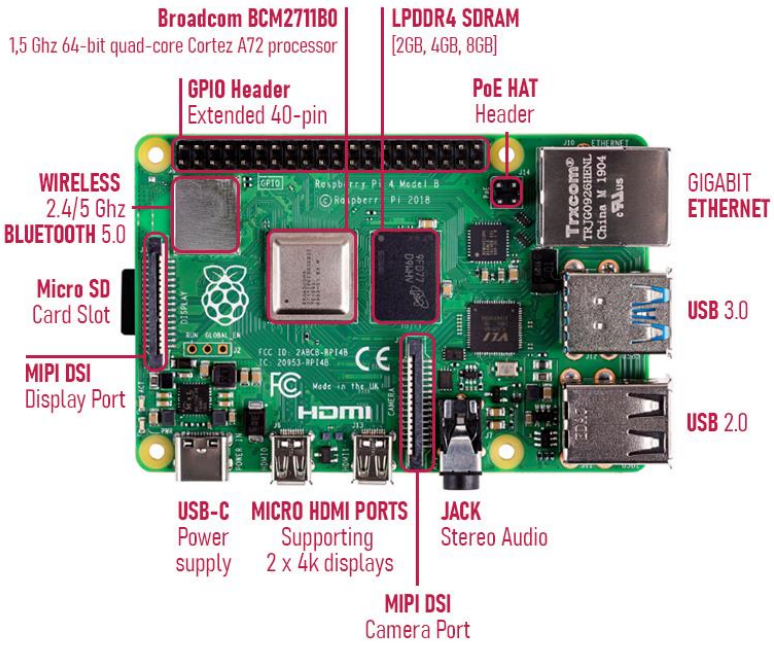

Figura 3-2.- Componentes Raspberry Pi 4 modelo B

Fuente: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Una de las principales características de la Raspberry Pi actual son la fila de pines GPIO (entrada / salida de propósito general) ubicados en la parte superior de la placa. Actualmente el encabezado GPIO consta de 40 pines, ya que hasta el modelo anterior a Raspberry Pi 1 modelo B + (2014) dicho encabezado solo constaba de 26 pines.

Cada pin GPIO se puede configurar mediante programación como un pin de entrada o un pin de salida para utilizarse con infinidad de propósitos [8].

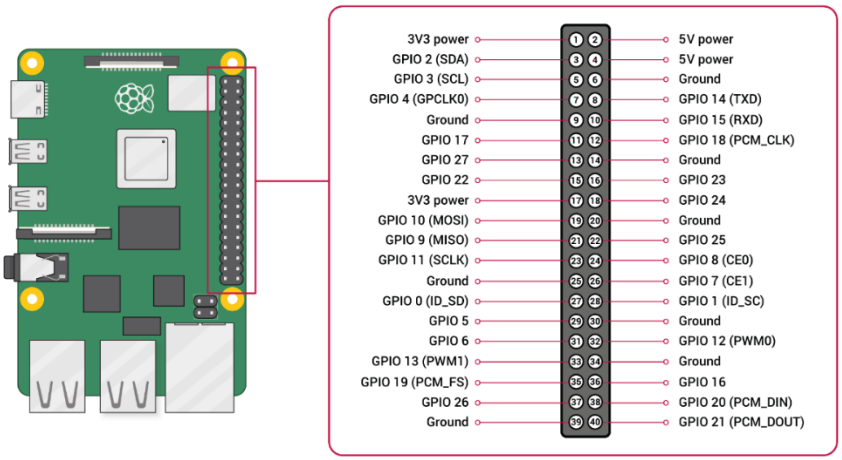

Figura 3-3.- Conexionado GPIO Raspberry PI 4

Fuente: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

El Sistema Operativo montado en la Raspberry Pi es Raspberry Pi OS (antes conocido como Raspbian)[9] se comenzó a usar oficialmente por primera vez en 2015, aunque su lanzamiento se produjo en junio de 2012. Es una distribución GNU/Linux basada en Debian por lo cual es libre y principalmente orientada a la enseñanza. Existen varias versiones de Raspbian, aunque actualmente está en uso es Raspbian Búster.

Figura 3-4.- Logotipo Raspberry Pi OS

Fuente: https://www.raspberrypi.com/software/

Aunque Debian es una distribución para la armhf, solo es compatible con las versiones posteriores a la usada en Raspberry Pi (CPU ARMv7-A y superiores) ya que no había versión Debian armhf para la CPU ARMv6 de Raspberry Pi.

El SO Raspberry Pi contiene más de 30000 paquetes o software precompilado incluido en el sistema para una fácil instalación en la Raspberry Pi.

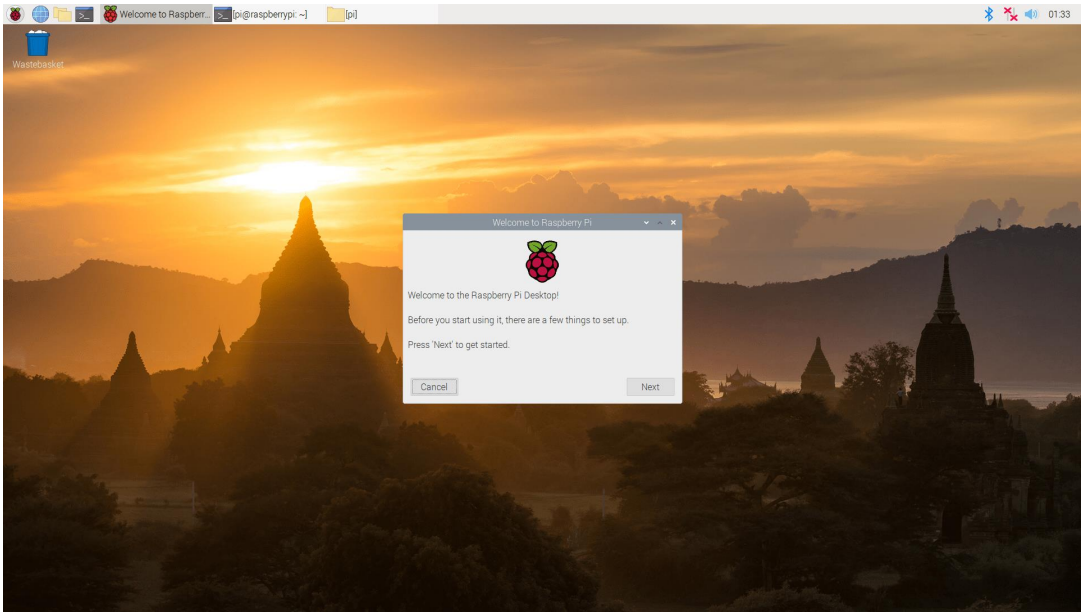

Figura 3-5.- Escritorio Raspberry PI OS

Como es una distribución libre de GNU/Linux es posible realizar cualquier proyecto, además cualquier software de código abierto se puede compilar en la misma Raspberry Pi. Además, es posible utilizar como ordenador personal siempre y cuando las aplicaciones a utilizar no requieran muchos recursos, lo cual se sea una solución para países subdesarrollados en la introducción de la informática.

3.1.2 Motor

El motor encargado del movimiento es un motorreductor de CC con escobillas de 12 V con una caja de engranajes con una relación 70:1, integrando un codificador en cuadratura, el cual proporciona una resolución de 64 pulsos por revolución en el eje del motor y en la salida de la caja de cambios se tendrán 4480 pulsos por revolución [10].

Figura 3-6.- Motor POLOLU

Fuente: https://www.pololu.com/product/4754

La caja de engranajes está montada en su mayoría por engranajes rectos, pero cuenta con engranajes helicoidales, gracias a los cuales reducen el ruido y mejoran la eficiencia.

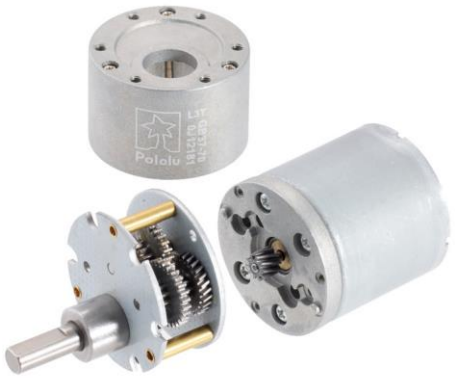

Figura 3-7.- Engranajes interiores Motor

Fuente: https://www.pololu.com/product/4754

Para la detección de la rotación es utilizado un codificador de efecto Hall con dos canales con un disco magnético, el cual tiene un saliente en la parte trasera del eje del motor. El codificador de cuadratura proporciona una resolución de 64 conteos por revolución del motor cuando se cuentan ambos extremos de ambos canales.

Para realizar el cálculo de los pulsos por revolución de la salida de la caja de engranajes, es necesario multiplicar por 64 la relación de engranajes. El motor/codificador está formado por seis cables de 8 ″ (20 cm) identificados mediante colores terminados por un cabezal hembra de 1 × 6 con un paso de 0,1 ″, como el mostrado en la Figura 3-8. La siguiente tabla contiene cual es la función de cada uno de los cables:

Color	Función
Rojo	Motor Power +
Negro	Motor Power -
Verde	Encoder GND
Azul	Encoder Vcc (3,5V – 20V)
Amarillo	Encoder A
Blanco	Encoder B

Tabla 3-2.- Funciones cables Encoder

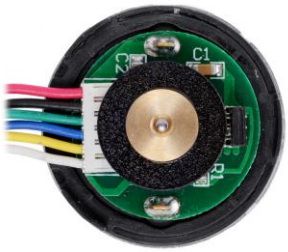

Figura 3-8.- Encoder

Fuente: https://www.pololu.com/product/4754

El sensor Hall [11] necesita un voltaje a la entrada, Vcc, que debe estar comprendidos por los valores 3,5V y 20V y consumir como mucho 10 mA. Ambas salidas A y B son señales cuadradas de 0 V a Vcc, las cuales están desfasadas aproximadamente 90° tal y como se puede observar en la Figura 3-9. La velocidad del motor viene marcada por la frecuencia de las transiciones y la dirección mediante el orden de las transiciones. En la siguiente captura del osciloscopio muestra las salidas del codificador A y B (amarillo y blanco) se ha utilizado un motor de 12 V a 12 V y un sensor Hall Vcc de 5 V:

Figura 3-9.- Comparativa canales Encoder

Fuente: https://www.pololu.com/product/4754

3.1.3 Módulo Puente H

Para realizar el control de motor se ha optado por el Controlador de Motor Dual H-Bridge SeeedStudio L298 [12] que utiliza un controlador de puente completo dual ST L298N, un circuito monolítico integrado en un paquete Multiwatt de 15 derivaciones y PowerSO20.

El controlador utilizado, ha sido diseñado para el manejo de niveles lógicos TTL estándar además para el manejo de cargas inductivas como solenoides, relés, motores paso a paso o motores de CC. Está formado por un puente doble de alto voltaje y alta corriente.

El módulo tiene dos entradas de activación para encender y apagar el dispositivo independientemente de la señal de entrada. Los transistores inferiores de cada emisor de los puentes están interconectados y los pines externos correspondientes se pueden usar para conectar resistencias de detección externas. Gracias a la entrada de energía adicional, la lógica puede operar a niveles de voltaje más bajos.

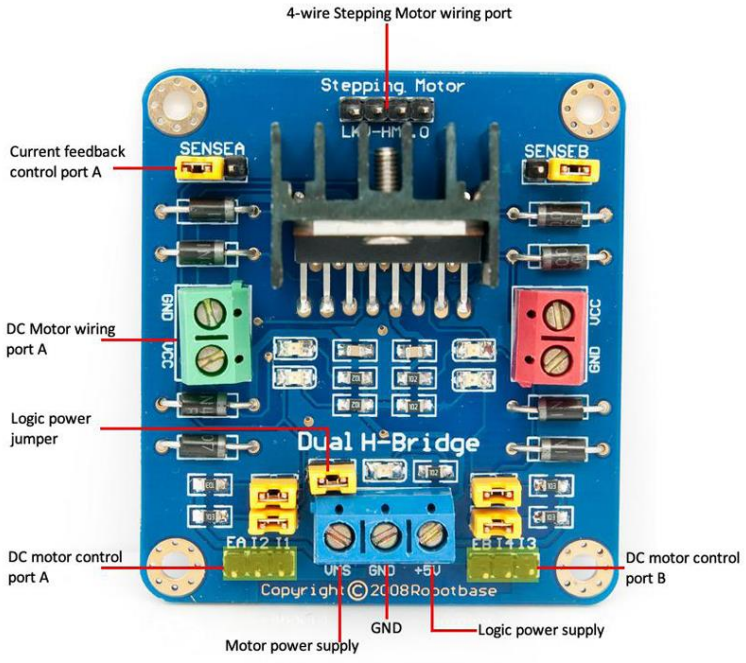

Figura 3-10.- Controlador de Motor Dual

Fuente: https://www.robotshop.com/es/es/controlador-motor-dual-h-bridge-seeedstudio-l298.html

El módulo de controlador doble H puede conducir dos motores de CC al mismo tiempo. El puerto A es completamente simétrico y el puerto B en la placa.

El puerto de entrada del motor de CC A tiene tres pines, I1, I2 y EA. I1 e I2 son puertos digitales que son utilizados por el motor para controlar la dirección, EA se conecta con el puerto PWM de la placa de control para controlar la velocidad del motor [13].

EA	I1	I2	Motor
HIGH	LOW	HIGH	Giro Derecho
	HIGH	LOW	Giro Izquierdo
	LOW	LOW	Paro Brusco Motor
	HIGH	HIGH	
LOW	-	-	Paro Libre Motor

Tabla 3-3.- Tabla verdad L298

Fuente: https://www.robotshop.com/es/es/controlador-motor-dual-h-bridge-seeedstudio-l298.html

Especificaciones:

- Conductor: L298
- Fuente de alimentación del controlador: +5 V a +46 V
- Controlador Io: 2 A
- Salida de potencia lógica Vss: +5 a +7 V (suministro interno +5 V)
- Corriente lógica: 0 a 36 mA
- Nivel de control: bajo -0,3 V a 1,5 V, alto: 2,3 V a Vss
- Habilitar nivel de señal: Bajo -0,3 V a 1,5 V, alto: 2,3 V a Vss
- Potencia máxima: 25 W
- Temperatura de trabajo: -25 °C a +130 °C
- Dimensión: 60 mm x 54 mm
- Peso del conductor: 48 g

3.1.4 Panel iluminación

El laboratorio remoto está disponible día y noche, por lo que se instala un sistema de iluminación formado por paneles LED de 24W de 60x30 mm [14]. El factor de potencia del modelo seleccionado es > 0,95, que es energéticamente eficiente. Proporciona una excelente uniformidad de luz y está compuesto por un cuerpo de aluminio blanco.

Figura 3-11.- Panel Led

Fuente: https://www.innova-led.es/index.php?id_product=4271&controller=product&id_lang=1

Parámetros básicos

- TIPO CORRIENTE : AC 170-265V
- Vatios(W) : 24W
- Factor potencia : 0.95
- Lúmenes : 2.100Lm
- Rendimiento : 88Lm/W
- Color Luz : 4500K / 6000K
- CRI (%) : >80%
- Ángulo de Luz : 120º

Parámetros externos

- Protección : IP40
- Dimensiones : 595X295x10mm

Certificación de calidad

- Vida Garantía : 2 Años
- Vida Normal : 30.000H
- Certificación CE : Si
- ROHS Certificación : Si

3.1.5 Relé

Para este proyecto, utilizamos el módulo de relé de 4 relés de 5V de Songle. Cada canal requiere 15-20mA. Los contactos del relé han sido diseñados para ser conmutados tensiones inferiores a 250 V CA a 10 A o menos de 30 V CC a 10 A. Posee una interfaz estándar y se puede controlar por un microcontrolador directamente.

Cada canal está separado por medio de un optoacoplador (817c) para minimizar el ruido para la conmutación de carga y el led indica el estado del relé [15].

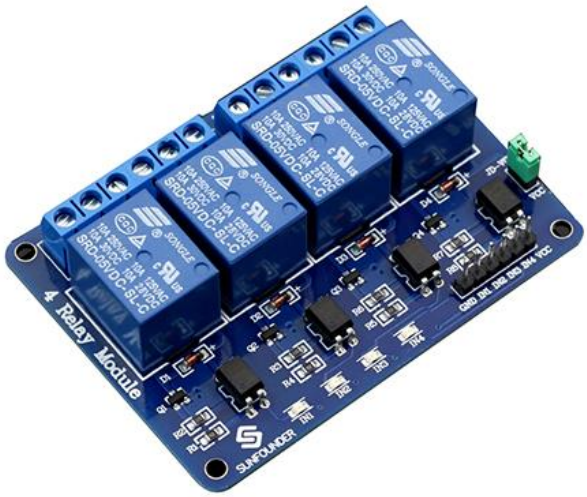

Figura 3-12.- Modulo 4 Relés

Fuente: http://wiki.sunfounder.cc/index.php?title=4_Channel_5V_Relay_Module

Especificaciones [16]:

- Número de serie: 2ph63083a
- Voltaje de operación: 5V DC
- Máxima salida del relevador (Voltaje/ Corriente): DC 30V/10A, AC 250V/10A
- Rangos de corriente: 10 A (NO) y 5 A (NC)
- Protocolo de comunicación: TTL (3.3 V o 5 V)
- Tiempo de acción: 10 ms/ 5 ms
- Tamaño: 75 mm x 55 mm x 19.3 mm (largo, ancho, alto)
- Peso: 61 gr.
- Pines:
  - ✓ Entradas: IN1, IN2, IN3, IN4, VCC, GND
  - ✓ Salidas: Conector hacia la carga (ejemplo, bombilla, bomba de agua, etc). La terminal de relés son COM, NO y NC (Común, Normalmente Abierto y Normalmente Cerrado respectivamente) están en la terminal de tornillo,
  - ✓ Jumper: JDVCC, VCC y GND

3.1.6 Cámara IP

Para poder visualizar en cada momento el RL se ha instalado una Cámara IP [17], la cual podrá ser accesible a través de una URL (IP configurada en la propia cámara).

La cámara IP de D-Link modelo DCS-4633EV seleccionada utiliza un potente sensor CMOS de 3 megapíxeles que proporciona imágenes nítidas con una resolución de 2048 x 1536. Obteniendo hasta 3 secuencias de seguimiento y grabación. simultáneas incluida la compresión de vídeo mediante el nuevo códec H.265 para máxima eficiencia de ancho de banda y calidad de transmisión de imágenes.

Figura 3-13.- Cámara IP

Fuente: https://eu.dlink.com/es/es/products/dcs-4633ev-3-megapixel-vandal-proof-outdoor-dome-camera

Características principales:

- Sensor CMOS de barrido progresivo de 3 megapíxeles
- Resolución HD de 2048 x 1536
- Configuración de los códecs H.265, H.264 y MJPEG para la emisión en streaming y la grabación
- Detector de movimiento
- Puerto Fast Ethernet 10/100
- Power over Ethernet (PoE) garantiza una integración simple y flexible en su red
- Ángulo de visualización: (H) 105° / (V) 80° / (D) 137°
- Ángulo de ajuste (no motorizado): Vertical: 60° / Horizontal: 350°

3.1.7 Fuente Alimentación

La fuente de alimentación es el dispositivo encargado de convertir la corriente alterna en una o más corrientes continúas utilizadas para alimentar el sistema.

La fuente de alimentación utilizada en este proyecto, modelo AF09, convierte 220 V CA a 12 V CC. Este es el voltaje necesario para alimentar diferentes dispositivos.

Figura 3-14.- Fuente alimentación

Fuente: https://tienda.bricogeek.com/117-fuentes-de-alimentacion

3.2 SOFTWARE

El objetivo de UNILabs[18] es la compartición de los laboratorios desarrollados por las 14 universidades españolas que participan en el mismo, entre ellas la Universidad de Jaén. Se ha seguido su metodología, por ello se hace uso del protocolo RIP que establece este esquema para las comunicaciones:

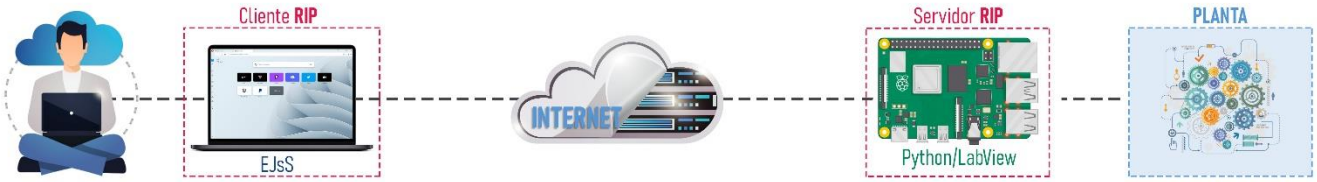

Figura 3-15.- Esquema general comunicaciones

Para el desarrollo de este TFG, por una parte, en el servidor se utiliza el protocolo RIP para las comunicaciones, en cual se explicará al detalle en el siguiente punto, dicho protocolo está implementado con Python como lenguaje de programación, en vez de LabVIEW, ya que debido a su gran potencial al tener diversidad para ser utilizado en varias plataformas y su gran número de librerías, proporcionando una amplia gama de opciones para lograr el propósito de este proyecto, además LabVIEW es necesario tener una licencia de pago mientras Python es un lenguaje de código abierto y, por tanto, gratuito.

Por otra parte, en el lado del cliente se ha optado por elegir para desarrollar con EJS, al ser una herramienta software enfocada al desarrollo de simulaciones interactivas, en este caso está basado en JavaScript al ser un lenguaje de programación orientado a objetos y estar incorporado como parte de los navegadores web, consiguiendo así una mejora en la interfaz de usuario y páginas web dinámicas.

Y por último en este apartado se detalla el estándar SCORM, que se usa para subir el contenido de las prácticas propuestas a la plataforma PLATEA.

3.2.1 Remote Interoperability Protocol

El protocolo RIP al ser de código abierto, según la Licencia Pública General GNU. Este es un protocolo de comunicaciones que se basa en el modelo cliente-servidor diseñado principalmente para OLS donde los clientes funcionan desde un navegador web. RIP proporciona a los usuarios y máquinas/programas cliente un mecanismo simple para recuperar la experiencia configurada en el servidor e información sobre cada entrada y salida. El protocolo también define métodos para leer y escribir estos valores de entrada y salida, respectivamente [1].

Las principales características de RIP son las siguientes:

- Definición experiencias en el OL.
- Obtención de metadatos relacionados con cada uno definido experiencia.
- Obtener una lista de variables que se pueden leer y escribir para cada experiencia.
- Obtener una lista de métodos para leer y escribir variables en cada experiencia.
- Invocar métodos para leer y escribir variables en cada experiencia.
- Definir eventos de servidor para enviar datos periódicamente o en función de cualquier otra condición de activación definida en una experiencia.
- Suscribir un cliente a cualquier evento de servidor declarado en una experiencia.

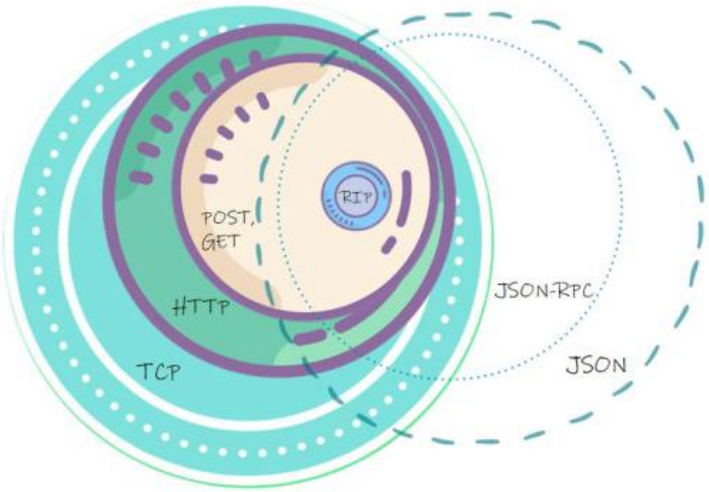

Figura 3-16.- RIP se basa en HTTP (POST y GET) y en el formato JSON-RPC.

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

RIP está basado en los métodos POST y GET para realizar el transporte de comunicaciones y JSON-RPC para darle formato a los mensajes. Los métodos POST y GET están agrupados en HTTP y se basan en conexiones TCP.

Por otra parte, JSON-RPC[19] está basado en el formato JSON. La Figura 3-16 representa estas ideas.

Si dos entidades de software usan el mismo protocolo, pueden comunicarse entre sí. Por lo tanto, se requiere la implementación de RIP tanto en el cliente como en el servidor.

Métodos Comunicación

Hay tres métodos de comunicación HTTP disponibles en RIP para comunicar al cliente con el servidor:

1. **GET** - Obtener metadatos OL. Un servidor RIP debe implementar un punto final de servicio web BaseURL: puerto / RIP para atender estas solicitudes.
2. **POST** - Enviar solicitudes de cliente a servidor para la escritura de los valores en las variables OL y lectura de los valores en las variables OL. Un servidor RIP debe implementar un punto final de servicio web BaseURL: puerto / RIP / POST para atender estas solicitudes.
3. **SSE** - Suscribir al cliente a flujos de datos para que reciba actualizaciones de valores de variables OL de servidor a cliente. Un servidor RIP debe implementar un punto final de servicio web BaseURL: puerto/RIP/SSE para atender estas solicitudes.

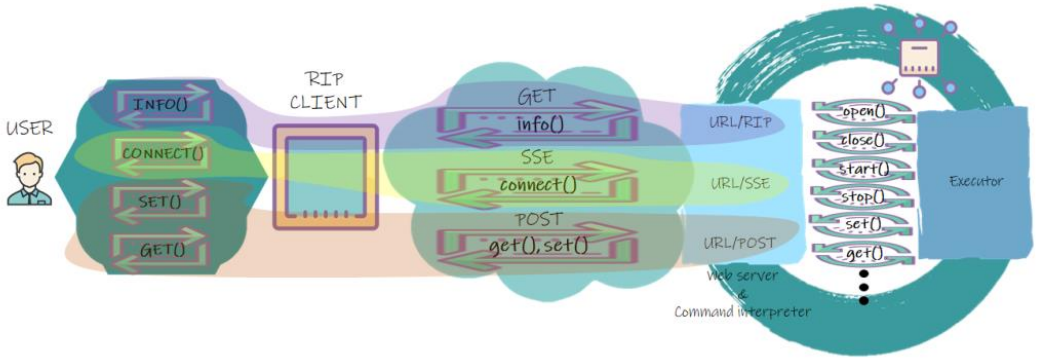

Figura 3-17.- Representación de la correspondencia entre funciones y métodos.

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

La Tabla 3-4 muestra la correspondencia entre las funciones de protocolo externo de la API de alto nivel y los métodos HTTP de la API de bajo nivel. La columna de la derecha también indica la URL del servidor web del RIP con el que se comunican cada uno de los métodos y funciones. Una representación de estas mismas ideas se muestra en la

Figura 3-17. Aunque deberían, no todas las funciones internas están representadas en la Figura; sólo seis de nueve aparecen en él.

Función Cliente RIP (High-Level API)	Método HTTP (Low-Level API)	Servidor Web
info (callback) **info** (callback, expId)	**GET**	/ BaseURL: puerto / RIP
set (variableNames, variableValues, callback, expId) **get** (variableNames, expId)	**POST**	/ BaseURL: puerto / RIP / POST
connect (expId, callback)	**SSE**	/ BaseURL: puerto / RIP / SSE

Tabla 3-4.- Correspondencia entre las funciones externas de RIP, los métodos HTTP y los puntos finales del servidor web de RIP

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

Servidor RIP

El Servidor Web admite (y maneja de diferentes maneras) tres tipos de solicitudes: GET (usado para recuperar experiencias metadatos), SSE (utilizado para obtener actualizaciones de datos de servidor a cliente) y POST (utilizado para enviar actualizaciones de cliente a servidor o solicitudes de cliente a servidor para actualizaciones de datos). Estos diferentes métodos están asociados cada uno con los tres casos básicos de uso, a saber:

- Metadatos - Los clientes que necesitan información sobre el laboratorio envían una solicitud HTTP GET a la URL asociada con el laboratorio. El servidor RIP responde con una estructura JSON-RPC que informa al cliente, según los parámetros de la solicitud:
    1. Información general sobre el OL: ¿cuáles son las experiencias definidas y cómo se puede acceder a ellas.
    2. Información detallada de una experiencia (cuando la solicitud incluye el ID de experiencia como parámetro).
- Observador - Los clientes que deseen actualizar el estado de la planta se suscribirán al flujo de eventos de SSE asociado con la experiencia en cuestión.
- Operador – Los clientes que desean interactuar con el OL o reciben unas actualizaciones a pedido envían solicitudes POST mediante comandos codificados como una estructura RIP-JSON-RPC.

La Figura 3-18 muestra cómo es implementado el protocolo RIP en la arquitectura de un Servidor RIP. En resumen, se definen las tres partes: Servidor web, en cual realiza el control de las conexiones de clientes, sesiones de usuario, etc., Intérprete de comandos, que habla RIP, y el Ejecutor, que controla la ejecución del laboratorio programas de controlo modelos de simulación.

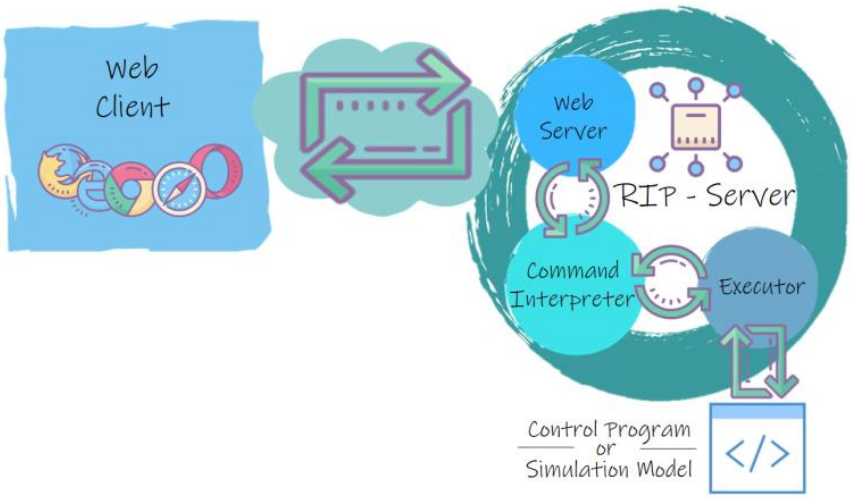

Figura 3-18.- Vista arquitectónica de un servidor RIP

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

Una experiencia representa una actividad de laboratorio asociada con un OL:

- En el caso de las RL, cada experiencia se implementa como un programa de control, que en general se encarga de gestionar las conexiones físicas con el hardware, medidas de seguridad y cualquier otra funcionalidad que el diseñador del laboratorio haya considerado apropiado incluir.
- En el caso de VL, cada experiencia se implementa como un modelo de simulación que representa un sistema real.

Cliente RIP

El cliente RIP debe implementar los métodos de protocolo de comunicación (POST, GET, SSE, etc.) requeridos en el formato adecuado para leer y escribir el contenido del mensaje (JSON-RPC, etc.), así como la estructura y funcionalidad definida por RIP. (detallado en secciones posteriores).

Funciones

La funcionalidad definida y utilizada en el protocolo se puede dividir en dos tipos: interna y externa.

- Las funciones internas están definidas como funciones privadas y son usadas internamente por las implementaciones del servidor RIP para comunicarse con el modelo de simulación o el programa de control utilizado en el OL.
- Las funciones externas al ser métodos en la parte del cliente son usadas por clientes RIP para obtener metadatos sobre las experiencias y escribir y leer datos desde y hacia el OL.

Por lo tanto, el cliente RIP debe implementar funciones externas y el servidor RIP debe realizar funciones internas (ver Figura 3-19). En un cliente RIP las funciones externas usadas por los métodos basados en el cliente interactúa con el servidor web utilizado en el un servidor RIP, transformando mediante el Shell en un conjunto de una o más funciones internas y finalmente es ejecutado por el componente Ejecutor (ver Figura 3-18).

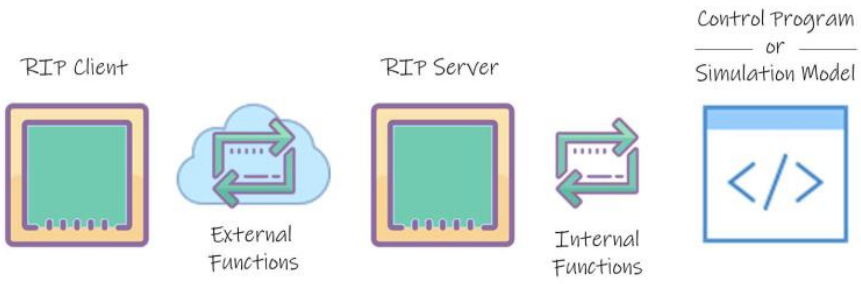

Figura 3-19.- Funciones internas y externas en clientes y servidores RIP

Fuente: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

3.2.2 Easy JavaScript Simulations

Easy JavaScript Simulations (EJsS) [20] es una herramienta software de código abierto, pertenece al proyecto Open Source Physics, desarrollada para ayudar a la creación de simulaciones científicas que ayuden a profesores y alumnos, es este caso para desarrollar laboratorios virtuales o remotos.

El diseño de EjsS se basa en el modelo-vista-controlador (MVC) [21], mediante el cual sigue el patrón de arquitectura de software, donde separa los datos de una aplicación, el interfaz de usuario y la lógica del controlador:

(1) **Modelo** que explica el fenómeno bajo estudio en términos de variables y las relaciones entre ellas, expresadas mediante algoritmos informáticos.

(2) **Controlador** que define las acciones que el usuario utiliza durante la simulación.

(3) **Vista** que incluye representaciones de varios estados en los que se puede representar el fenómeno.

A continuación, los pasos para crear una aplicación en EJS son los siguientes:

1. Defina un conjunto de variables que describen el sistema y sus fórmulas relacionadas.
2. Defina una vista que pueda representar el estado por el que pasa el modelo.
3. Especificar controles para describir las acciones que se pueden realizar en la simulación.

Estas tres partes están relacionadas entre sí porque el estado del modelo afecta la vista y las acciones de control afectan el estado del modelo. Por último, la GUI, o vista, contiene información sobre ambos, por lo que la vista afecta tanto al modelo como al control.

Además, EJS es capaz de simplificar la construcción de simulaciones al eliminar controles y fusionar partes en la vista con partes en el modelo. Por lo tanto, la aplicación se crea utilizando el motor de modelado interno de EJS para definir el modelo que se modelará y crear la vista donde muestre el estado del modelo y realice la respuesta correcta a la entrada del usuario.

Por lo tanto, para definir un modelo en EJS, debe definir variables que describan el proceso, inicializarlas y escribir las fórmulas que conforman el modelo.

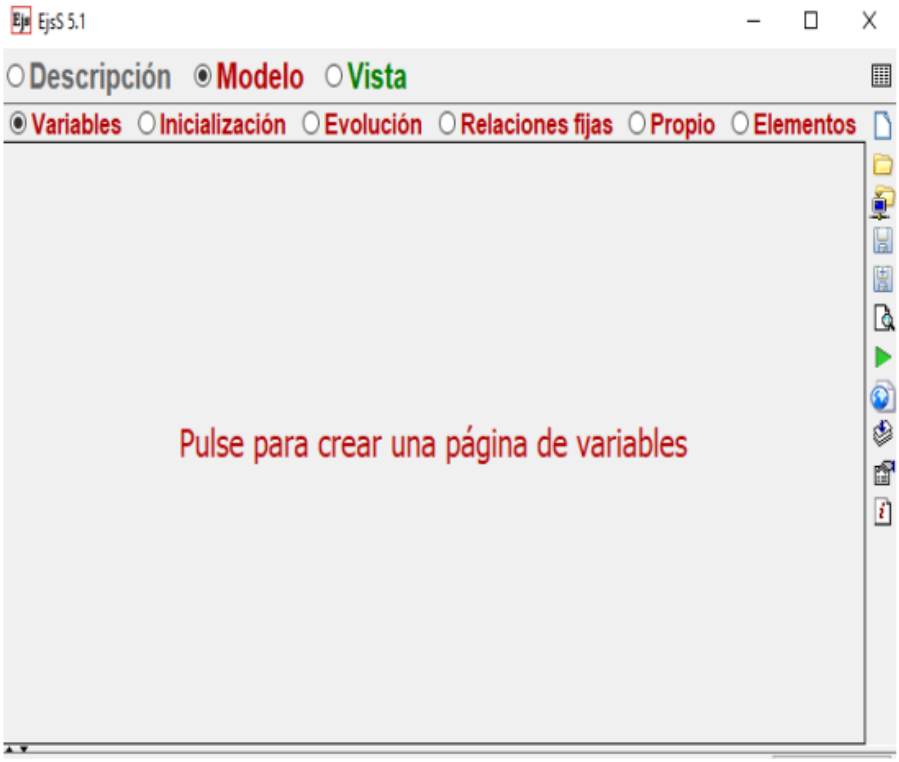

Figura 3-20.- Panel Modelo en EJS

Fuente: https://www.um.es/fem/EjsWiki/

Las vistas son la interfaz entre el usuario y el modelo. Fue creado para representar visualmente las propiedades más importantes del modelo y su comportamiento dinámico para garantizar la interactividad de las acciones del usuario.

EJS contiene una biblioteca completa de elementos visuales que permite a los diseñadores crear vistas muy complejas. Puede vincular las propiedades del elemento de vista a las variables del modelo para establecer comunicación en ambos sentidos entre el modelo y la vista [22]. Los cambios en las variables del modelo se capturan y transfieren automáticamente a la vista. En cambio, una acción del usuario en la vista cambia automáticamente el valor de la variable del modelo correspondiente.

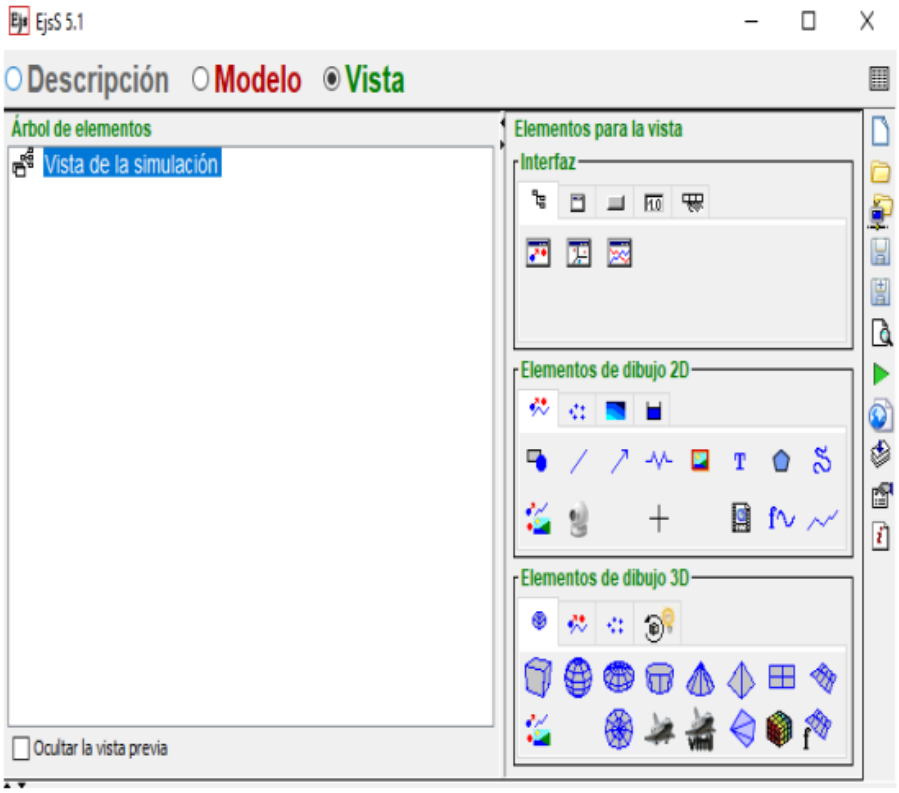

Figura 3-21.- Panel Vista en EJS

Fuente: https://www.um.es/fem/EjsWiki/

En resumen, la parte científica de la simulación es el modelo. Por otro lado, el modelado consiste en la creación de una interfaz gráfica de usuario (vista) para lo cual es necesario un alto grado de conocimiento en técnicas de programación avanzadas.

Cuando un diseñador define el modelo y la vista de la simulación, para ello se genera con EJS el código Java, que es compilado en el programa, comprimido los archivos generados en un solo archivo y ejecuta la simulación como su subprograma en la página HTML. Por lo tanto, cuando el usuario está completo, la simulación puede ejecutarse como un subprograma o una aplicación y publicarse en la web.

3.2.3 Sharable Content Object Reference Model

SCORM (Sharable Content Object Reference Model) [23] es el conjunto de especificaciones y estándares que permiten la creación de objetos educativos estructurados. SCORM puede considerarse un formato estándar para contenido de aprendizaje E-Learning.

El sistema de administración de contenido web original usaba su propio formato para el contenido que distribuía. Por lo tanto, no fue posible compartir dicho contenido. SCORM le permite crear contenido que se puede importar a varios sistemas de gestión de aprendizaje si es compatible con el estándar SCORM.

Las ventajas que SCORM ofrece son:

- ✓ **Accesibilidad**: La posibilidad de acceder a elementos de formación desde sitios remotos utilizando tecnología web, y la difusión de elementos de formación en distintos sitios.
- ✓ **Adaptabilidad**: La capacidad de personalizar la formación según las necesidades de las personas y organizaciones.
- ✓ **Durabilidad**: La capacidad de soportar cambios en la tecnología sin tener que rediseñar, reconstruir o reescribir el código.
- ✓ **Interoperabilidad**: Posibilidad de ser utilizado en otro lugar, con diferentes conjuntos de herramientas, o con diferentes componentes de aprendizaje de plataforma desarrollados en el sitio, conjuntos de herramientas específicas o plataformas específicas.
- ✓ **Reusabilidad**: La flexibilidad para integrar componentes de aprendizaje en diferentes contextos y aplicaciones.
- ✓ **Reducción de costes**: El paquete SCORM reduce significativamente los costos de capacitación debido a los beneficios anteriores.

4. HERRAMIENTAS Y MÉTODOS

4.1 SERVIDOR RIP

Se ha desarrollado un servidor RIP basado en una adaptación del servidor desarrollado por la UNED [24] a partir del cual se ha adaptado ser usado en una Raspberry Pi y añadido un sistema de control de la planta a través de GPIO, en la Figura 4-1 se detallan todas las conexiones realizadas en el sistema, detallando la conexión con los pines GPIO de la Raspberry Pi mediante los cuales serán controlados todos los elementos que componen la planta en función del control realizado desde la interfaz del cliente, en este caso un navegador web.

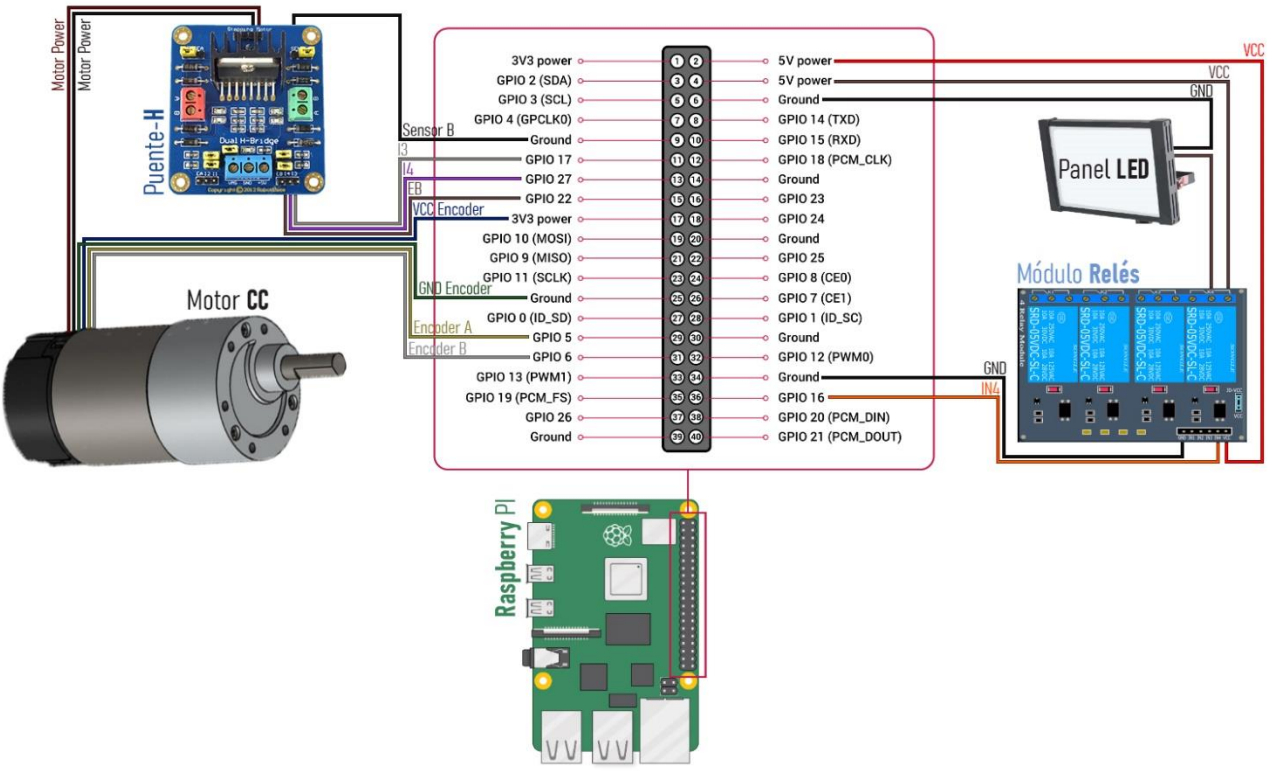

Figura 4-1.- Conexiones sistema con pines GPIO

Configuración Servidor RIP

Para configurar el servidor RIP en la Raspberry PI es necesario la instalación de varias librerías de terceros para su correcto funcionamiento, las cuales no se encuentran instaladas en Raspberry PI IOS

Para empezar, es necesario tener instalada la versión 3 Python. Además, se va a utilizar un entorno virtual para lo cual se usará los siguientes comandos:

$$pip3\ install\ virtualenv$$

$$virtualenv - p\ python3\ venv$$

Al usar la opción ***-p python3*** es para asegurar que es usada la versión correcta, y ***venv*** es la carpeta donde se almacena el entorno, el siguiente paso es instalar las dependencias:

$$venv/bin/pip\ install\ cherrypy\ ujson$$

Todos ellas son necesarias para poder ejecutar el servidor RIP correctamente y finalmente para iniciar el servidor RIP:

$$venv/bin/python3\ App.py$$

El servidor RIP debería comenzar a escuchar por el puerto 8080. Para verificar si realmente está funcionando, se abre un navegador y se escribe la URL:

$$http://localhost:8080/RIP$$

Se obtendrá una respuesta JSON como se muestra en la Figura 4-2. Donde se puede observar los metadatos del servidor RIP.

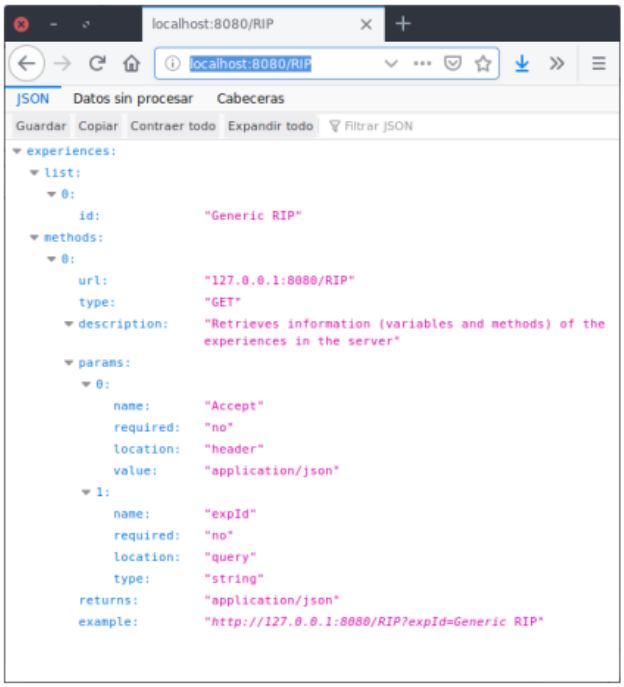

Figura 4-2.- Prueba de la respuesta del servidor RIP.

Arquitectura del Servidor RIP

El servidor está dividido en varios ficheros, a fin de facilitar su modularidad y futuras ampliaciones. En la Figura 4-3 se muestra la estructura de los principales ficheros del servidor y a continuación se analizará cuáles son sus funciones:

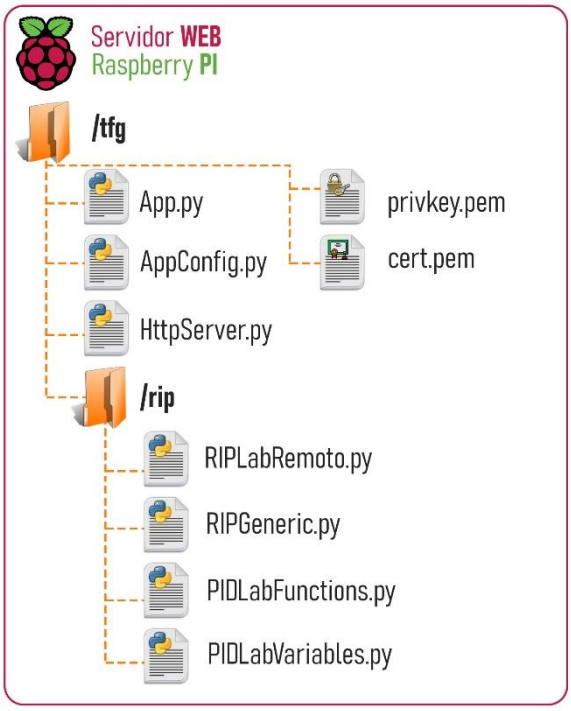

Figura 4-3.- Árbol archivos servidor WEB

✓ **HttpServer.py:** La función de este archivo es encargarse de activar el servidor HTTPS, mediante el cual gestionará todos mensajes recibidos (GET) desde el interfaz web creado con EJsS y enviará su correspondiente respuesta (POST) una vez procesado por el servidor RIP.

✓ **App.py:** Archivo principal, el cual es ejecutado (como se ha visto en el punto anterior) para activar el servidor RIP. Hace la importación de las diferentes clases necesarias para la configuración y funcionamiento del servidor RIP. Se puede ver una muestra del código en la Figura 4-4.

```
from HttpServer import HttpServer
from AppConfig import config
import rip

def load_control(control):
  import_str = 'from rip import {0}'.format(control['impl_module'])
  exec(import_str)
  RIPImpl = getattr(rip, control['impl_module'])
  RIPControl = getattr(RIPImpl, control.get('impl_name', control['impl_module']))
  info = config['control']['info']
  return RIPControl(info)

if __name__ == "__main__":
  control = load_control(config['control'])
  enable_ssl = config['server'].get('ssl', False)
  HttpServer(
    host=config['server']['host'],
    port=config['server']['port'],
    control=control
  ).start(enable_ssl)
```

Figura 4-4.- Código archivo App.py

- ✓ **AppConfig.py:** Este archivo contiene la configuración del servidor RIP, detallando la dirección IP y puerto, además de habilitar la seguridad en las comunicaciones con el protocolo SSL. También se configura el nombre del módulo desde el que se implementa el acceso al hardware (*RIPLabRemoto*), las variables de lectura (readables) y escritura (writables), autores, descripción o palabras claves. Como se muestra el código del archivo en la siguiente Figura 4-5.

```
# This file contains the configuration of the RIP server application.
config = {
  # TO DO: The server will listen to host:port
  'server': {
    'host': '192.168.18.7',
    'port': 8080,
    'ssl': True,
  },
  # The 'control' section configures the mapping between the RIP protocol
  # and the actual implementation of the functionality.
  # The 'impl' field should contain the name of the module (.py) and the
  # class that implement the control interface
  'control': {
    'impl_module': 'RIPLabRemote',
    # Also, if the class name is not the same as the module name:
    #'impl_name': 'RIPOctave',
    'info': {
        'name':'Laboratorio Remoto',
        'description':'Implementacion del control en la Raspberry',
        'authors':'Jesus Lucena',
        'keywords':'Raspberry PI',
        'readables':[
        {
```

Figura 4-5.- Código archivo AppConfig.py

A continuación, se muestra la relación de las variables utilizadas por el protocolo RIP y EJsS para la comunicación, se han agrupado por tipo, en función de si son de lectura(readables) o escritura(writables):

Variable	Tipo	Descripción
led	boolean	Activa o desactiva el panel led.
spClient	double	Variable introducida en EjsS para Set Point.
uClient	double	Variable introducida en EjsS para PWM.
kp	double	Establece la constante proporcional.
ki	double	Establece la constante integral.
kd	double	Establece la constante derivativa.
practica	int	Variable utilizada para indicar la práctica.

Tabla 4-1.- Tabla variables escritura ('writables' - SET).

Variable	Tipo	Descripción
time	double	Devuelve el tiempo de la simulación.
spServer	double	Variable mostrada en gráficas EjsS de Set Point.
uServer	double	Variable mostrada en gráficas EjsS de señal PWM.
vel	double	Variable calculada velocidad motor.
e	double	Variable error sistema.
y	double	Variable señal salida.
u	double	Variable señal PWM.

Tabla 4-2.- Tabla variables lectura ('readables' - GET).

- ✓ **RIPLabRemoto.py:** Archivo creado para el acceso al hardware de bajo nivel a través de la API de RIP, para el cual será importada la clase *RIPGeneric.* Este archivo está formado por las variables de gestión del protocolo RIP (readables y writables) además de las funciones para la gestión del laboratorio, entre ellas destacar la función SSE que se encarga de la actualización de las variables.

```
def getValuesToNotify(self):
    '''
    Variables to include in periodic SSE updates
    '''
    if(VAR.practica==1):
        return [['time', 'uServer', 'vel'],
        [self.sampler.lastTime(), self.pwm(), self.velocidad()]]

    else:
        return [['time', 'spServer', 'u', 'e', 'y'],
        [self.sampler.lastTime(), self.setPoint(), self.control(), self.error(), self.salida()]]
```

Figura 4-6.- Función SSE protocolo RIP.

✓ **RIPLabFunctions.py:** Este archivo contiene las funciones auxiliares que son independientes del protocolo RIP, de las que destaca la función ***readEncoder ()***, encargada de la lectura de pulsos a través de los encoders del motor se muestra a continuación el fragmento de código:

```
'''
METODO LECTURA ENCODER
'''
def readEncoder(self):

    encoderA = GPIO.input(VAR.ENCODER_A)
    encoderB = GPIO.input(VAR.ENCODER_B)

    if((encoderA==1 and VAR.encoderB_old==1)or(encoderA==0 and VAR.encoderB_old==0)):
        VAR.contPulse+=1
    elif ((encoderA==1 and VAR.encoderB_old==0)or (encoderA==0 and VAR.encoderB_old==1)):
        VAR.contPulse-=1

    VAR.encoderA_old = encoderA
    VAR.encoderB_old = encoderB
```

Figura 4-7.- Función lectura pulsos encoders.

Esta función realiza la medición del número de pulsos, que en función del sentido del mismo incrementa o decrementa la variable ***"contPulse"*** dependiendo del estado anterior de cada Encoder. A continuación, se muestra una imagen donde se observa el sentido del motor en función de los estados de cada Encoder gracias a que ambas señales van desfasadas 90º.

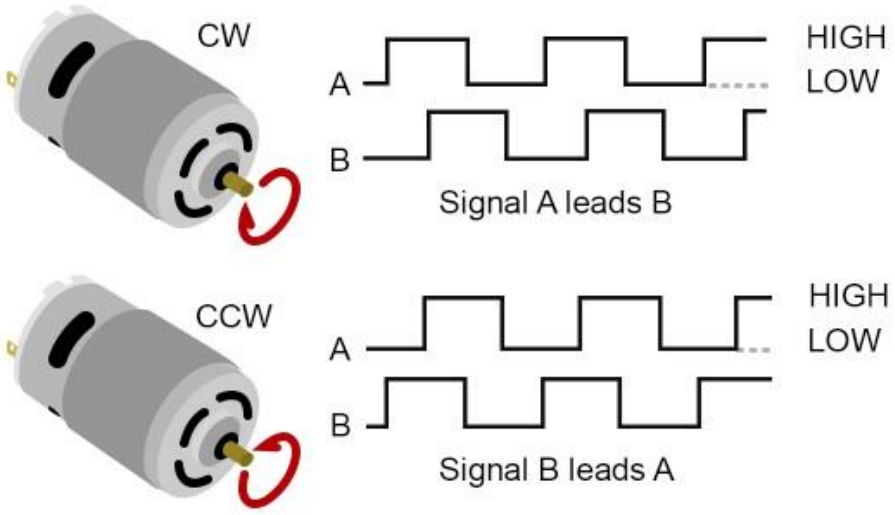

Figura 4-8.- Estados Encoder

Fuente: https://gimsonrobotics.co.uk/categories/dc-electric-motors/products/gr-wm2-high-torque-gearmotor-149-1-ratio-with-encoder

Por otro lado, la función ***loop ()*** es la encargada de realizar los cálculos en función de la práctica que se esté ejecutando, donde la variable ***"práctica"*** será la encargada de ver cual práctica está ejecutando el alumno:

```
'''
METODO CALCULO MOTOR
'''
def loop():

    # calculo velocidad - P1
    if(VAR.practica==1):
        VAR.v = VAR.contPulse*360/(VAR.sampleTime*VAR.pulsoRev)

    # calculos - P2
    else:
        # acciones del controlador
        VAR.e = VAR.spPulse-VAR.contPulse
        deltaT = time.time()-VAR.lastTime
        # derivativa
        VAR.ed = (VAR.e-VAR.eprev)/deltaT
        # integral
        VAR.ei += VAR.e*deltaT

        # acciones de control
        VAR.u = VAR.kp*VAR.e + VAR.ki*VAR.ei + VAR.kd*VAR.ed
        # sentido giro del motor
        if(VAR.u>0):
            GPIO.output(VAR.MOTOR_I3,GPIO.HIGH)
            GPIO.output(VAR.MOTOR_I4,GPIO.LOW)
        else:
            GPIO.output(VAR.MOTOR_I3,GPIO.LOW)
            GPIO.output(VAR.MOTOR_I4,GPIO.HIGH)

        # saturación de la acción del control
        if(VAR.u>100.0):
            VAR.u=100
        if(VAR.u<-100.0):
            VAR.u=-100

        # actaulizacion variables
        VAR.pwm.ChangeDutyCycle(abs(VAR.u))
        VAR.eprev = VAR.e

        threading.Timer(VAR.sampleTime, loop).start()
```

Figura 4-9.- Función calculo velocidad y PID.

- ✓ Si la variable "practica" es 1, corresponderá con la Práctica 1 – Identificación de la dinámica del motor, por lo cual para dicho laboratorio solo es necesario realizar el cálculo de la velocidad del motor, en este caso será revoluciones por segundo.

- ✓ Si la variable "práctica" es distinta de 1, corresponderá con la Práctica 2 – Control en posición del motor, aunque en el cliente EjsS se ha utilizado el valor "2" para identificar el laboratorio para futuras ampliaciones del código poder diferenciar los distintos laboratorios. En la primera parte del código dentro de la parte destinada a la práctica 2, se realizan los cálculos para el control PID, los cuales están explicados detalladamente en el **Apartado 4.4**, una vez obtenida la señal de control ***"u"*** en función de su valor se podrá actuar sobre la dirección de giro del motor y mandar señal PWM al mismo a través de la función *"ChangeDutyCycle"*.

✓ **RIPGeneric.py:** Clase encargada de proporcionar la funcionalidad común del protocolo RIP. Contiene la implementación de referencia del protocolo, donde se encuentran las funciones principales del mismo.

✓ **RIPLabVariables.py:** Contiene las distintas variables utilizadas por el servidor RIP, tales como el conexionado GPIO. Así como las variables auxiliares utilizadas por los distintos métodos para los cálculos realizados por el servidor.

✓ **cert.pem:** Archivo que contiene el certificado autofirmado de usuario final para realizar la comunicación segura a través de SSL. Como se puede observar en la Figura 4-10 un archivo PEM es simplemente un archivo de texto que incluye una codificación Base64 del texto del certificado.


```
-----BEGIN CERTIFICATE-----
MIIENzCCAx+gAwIBAgIUJ8I/MGwqWYT68X7y+Lqy5Zjm6iAwDQYJKoZIhvcNAQEL
BQAwgaoxCzAJBgNVBAYTAkVTMRAwDgYDVQQIDAdKYcODwqluMRAwDgYDVQQHDAdM
aW5hcmVzMR8wHQYDVQQKDBZVbml2ZXJzaWRhZCBkZSBKYcODwqluMRYwFAYDVQQL
DA1UZWxlbcODwqF0aWNhMRgwFgYDVQQDDA9KZXPDg8K6cyBMdWNlbmExJDAiBgkq
hkiG9w0BCQEWFWpsYzAwMDA4QHJlZC51amFlbi5lczAeFw0yMTExMDgyMDQ5MDla
Fw0yMjExMDgyMDQ5MDlaMIGqMQswCQYDVQQGEwJFUzEQMA4GA1UECAwHSmHDg8Kp
bjEQMA4GA1UEBwwHTGluYXJlczEfMB0GA1UECgwWVW5pdmVyc2lkYWQgZGUgSmHD
g8KpbjEWMBQGA1UECwwNVGVsZW3Dg8KhdGljYTEYMBYGA1UEAwwPSmVzw4PCunMg
THVjZW5hMSQwIgYJKoZIhvcNAQkBFhVqbGMwMDAwOEByZWQudWphZW4uZXMwggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDcKf2N6ugSLIHnXP01tHy2qcrT
zwhcYjpa/6WOIbid4IU1thFXVQvlHtNQ3pLogOlnWspCuYc5EpAr8md+txQbk8KX
dqk7ipRNTwXVGxvQgc64d573viVBts7/pF+tPjReiQlQZoHYcf1GNcPy53a3QQl/
Kztp5jpnO/iom24cS4V3CLsNBAX97A2lKYKj5ROur3SmMajW9gXHMW31EjDHu9e1
3E/bEQ+860DKExhCiZrum9Hv4nKlKlHLyUA6jVUEBa0HBknkYGiag3i3x//53WBV
CAFCkfayuOMPPaYhap9gQNwJz81bPkqavvXl3ESjlgWatN5YX9d83ZoloG8hAgMB
AAGjUzBRMB0GA1UdDgQWBBQA8MDyi9HpqOnHgBZjrSrurvE7lDAfBgNVHSMEGDAW
gBQA8MDyi9HpqOnHgBZjrSrurvE7lDAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3
DQEBCwUAA4IBAQAdHX0/VTXaLq5PyhHs9ysAWUqSptSFeoPIcTwb7kTGC1+TLL40
31OMZkw4Od2QDNf391lhaIx4LhGQCKweLmMH59PoJj27t+kgZKotCJOecKvwKha0
e3BJNSVmY8mZNVzPBHln/Bqc4qGavLHjJ4pWB8ElxZMaAuFkENrsJSU1Xhql3slJ
XdjwWKEKsrulSfF37MkIzUf5NwHL5V2bfY8btTLEDHFz4JidwLXCag0aNZaLfNWv
UtuPmRS05oXIHc//52Nc58oeh/MTAzlFJvVoyQKWe7HyBvn8F6zUMYyPrbAMyT1q
wt4aTsjxmfd3kHSUJ/0QJcZvU3n9nekFUdcf
-----END CERTIFICATE-----
```


Figura 4-10.- Contenido archivo cert.pem

✓ **privkey.pem:** Es una clave privada RSA generada junto con el certificado creado. A continuación, se muestra el contenido de dicho archivo en la siguiente figura.


```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDcKf2N6ugSLIHn
XP01tHy2qcrTzwhcYjpa/6WOIbid4IU1thFXVQvlHtNQ3pLogOlnWspCuYc5EpAr
8md+txQbk8KXdqk7ipRNTwXVGxvQgc64d573viVBts7/pF+tPjReiQlQZoHYcf1G
NcPy53a3QQl/Kztp5jpnO/iom24cS4V3CLsNBAX97A2lKYKj5ROur3SmMajW9gXH
MW31EjDHu9e13E/bEQ+860DKExhCiZrum9Hv4nKlKlHLyUA6jVUEBa0HBknkYGia
g3i3x//53WBVCAFCkfayuOMPPaYhap9gQNwJz81bPkqavvXl3ESjlgWatN5YX9d8
3ZoloG8hAgMBAAECggEBAK1x7yzUuQtIQPayTVD35aBJ115SJfYR9IMoD5cPfc/y
xz7vNm4ktABG7pcxIW9F0sGvxtQWMS+HLklQ6LPtK0wmdhC5DM25eOgGr0OZ3am/
+cldg8EM7XD9wcmoBzAGZ19qegn7F6Qa+TtMtBgyvVXVSNQEjQP6ulwF5Pl/qZlj
dvjdCPPcG8JxygoY83ugwLwZV55GXGoqwv0WM1dFWZVPJ+5CPq9+sferr2ZbtCFA
d2BJrDbWNxeFqdOpJxJj3inmX2JI0iLYC6bVZyLdcA/W7Yxw0tn19eUyzt6hbnLF
89OqDM1hUt1fwUCcPg2q3wTeJoUSsH/hQ4Tckv/WaqECgYEA/Fr6GiE5bjs1Eouw
xhIGoODuwE1jwqgrnwC6bOGlXGGgPQiEthz2o+tuHPBijm6HdnO/oBCZXPfrIEEa
wEDqxsBzih2PE7Kh83LtoLM4u4Bh4tGfg8WDVGn/+beIx58QPr9uLRsd4fshOtjE
x5JM78VQq2t7jtwMSeY42qz+hh0CgYEA31f+ZVZFlPEHL1JMpiKkUyU1N6Vmrrxd
oRONL47igdGwy4eePhf2LaD30wdmcrzALu2rsI5DiBtl4cFT6sAsQaiR5uSLxFeY
XXWQrWeEBG+cP4wBDWz88XaafJa5ytz4UWhv6RjFX/PwXSLTj94fHtfg9V//ia8c
bdWiUava7dUCgYBMl4Q/OARJS5kRFurBQylV3M85+IMlyorqyHrMnzPkNRFAVPyK
lYHaQJN8Hw2asztUHU900nTpvalUcc4fudRIHJM0wUHytY3n6OLsmljGt/F9uSfU
A7eXgFCbxzOxerqggNN8xzUrqF/uj2oyNFo7j8SrMoUe1/S23TQ2UrkxlQKBgHrO
o6+NFI2lStIWatcfTUP6vZxvclq6cLFVUtjHtHy1lIUVeUmjWlYToLGE7CxC5Vno
jfNzBmccyZSs/Tujl7Iywqc3HDNGBx28u2Td+pWF2WCH1zxOpXcQRDolltuZTWFW
D9MYL89nDs/ninUdKfeHtMa3lodapkMuqSZHXQZRAoGBAMyi6unJZTzERWYTJCaY
Pt2mRq2FsfJSk3XkcUBnC/mu66Qh6g/4HZXGDKPBRdmpawGwxv4f5q5wcbLdzapS
TDuT7bBYnnIR4fkeMEG6ZKoOTv5X51QRS4hPgaSyZ9pSUDSNEe5PkG0Nd2fTFKbk
Cxh6DeQlRSEpO2kmCN8RUWfH
-----END PRIVATE KEY-----
```


Figura 4-11.- Contenido archivo privkey.pem

4.2 CLIENTE

La interfaz gráfica del cliente (GUI) es la herramienta que los alumnos utilizaran para acceder al laboratorio remoto, configurar/parametrizar el comportamiento del motor, y visualizar la evolución de las variables de interés para el análisis del comportamiento del sistema. Dicha interfaz ha sido desarrollada utilizando EJsS [25].En este capítulo se describe el aspecto y funcionalidad del interfaz desarrollado.

Configuración Cliente

La aplicación EjsS posee un cliente RIP [26] desarrollado con JavaScript, el cual realiza la conexión con el servidor RIP que está operativo en el servidor localizado en la Raspberry PI.

Para agregar este elemento se debe acceder a la pestaña *Modelo* → *Elementos* → *SoftwareLinks* como muestra la Figura 4-12.

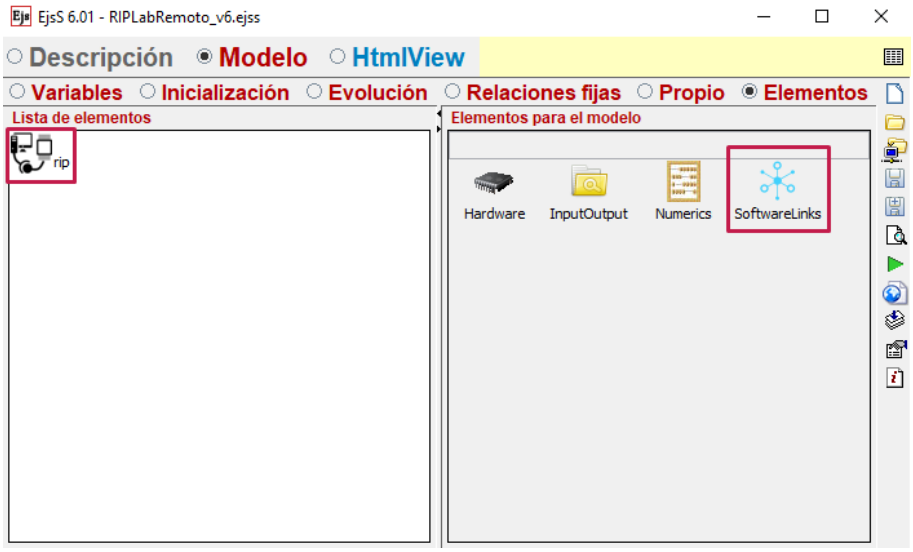

Figura 4-12.- Elemento rip en EjsS

Para ello se necesita configurar la dirección IP de la Raspberry PI, que es donde se encontrará el servidor RIP, en la pestaña *Server Configuration* → *Server url*. Como se muestra en la Figura 4-13.

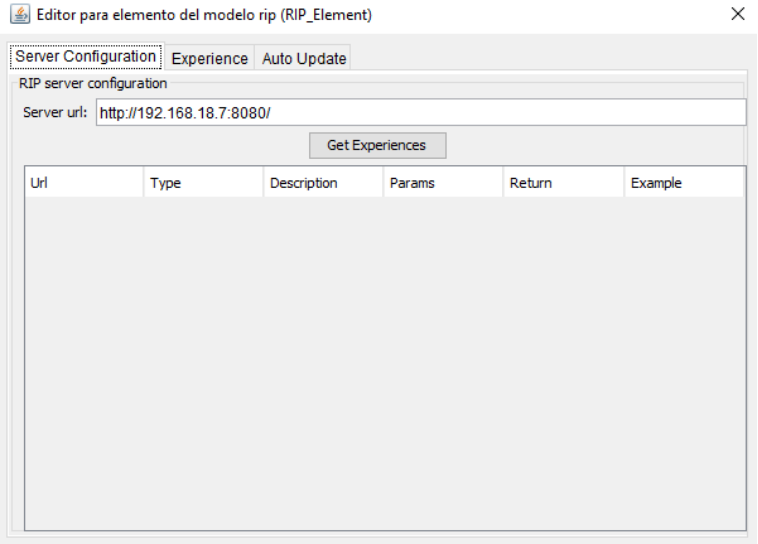

Figura 4-13.- Configuración rip en EjsS

Por último, en la pestaña *Auto Update* se deben enlazar las variables que va utilizar el sistema además de especificar si serán de escritura (*Writable*) o de lectura (*Readable*), tal como muestra la Figura 4-14.

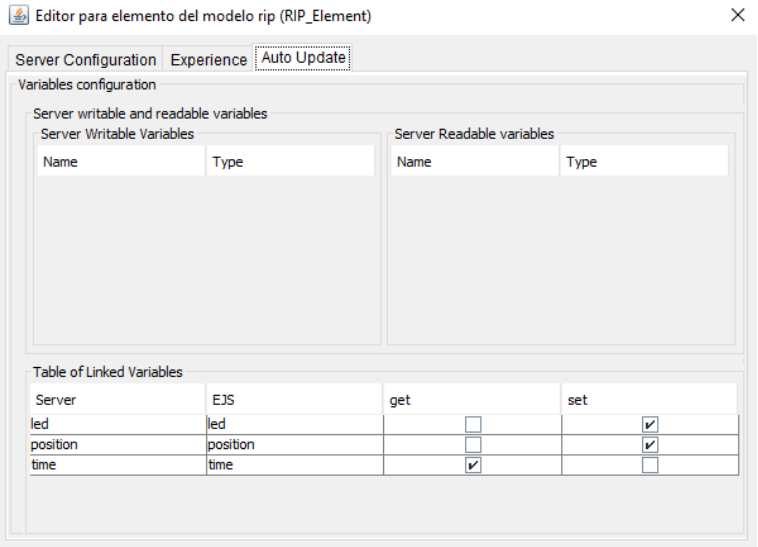

Figura 4-14.- Configuración variables rip en EjsS

Los métodos utilizados en la comunicación entre la aplicación web y el servidor RIP, realizada mediante el elemento RIP, dichos métodos se han explicado en el Apartado 4.1, a continuación, se vuelven a enumerar:

- ✓ **rip.connect():** Se realiza la llamada al método *connect* para el inicio de la comunicación con el servidor. Para ello se llama desde la pestaña *Modelo* → *Inicialización*, para conectar automáticamente cada vez que ejecute el cliente.

- ✓ **rip.get([variables]):** Se realiza la llamada al método *get* del servidor y este devuelve su valor. Es utilizado para actualizar la interfaz de usuario.

- ✓ **rip.set([variables], [values]):** Se realiza la llamada al método *set,* donde se actualizan las variables con las variables con el contenido de los parámetros *values*. Mientras dura la ejecución el programa utilizará dicho método para la configuración por parte del usuario de los parámetros del sistema.

Interfaz de usuario

La interfaz con la que el alumno trabajará será como la mostrada a continuación:

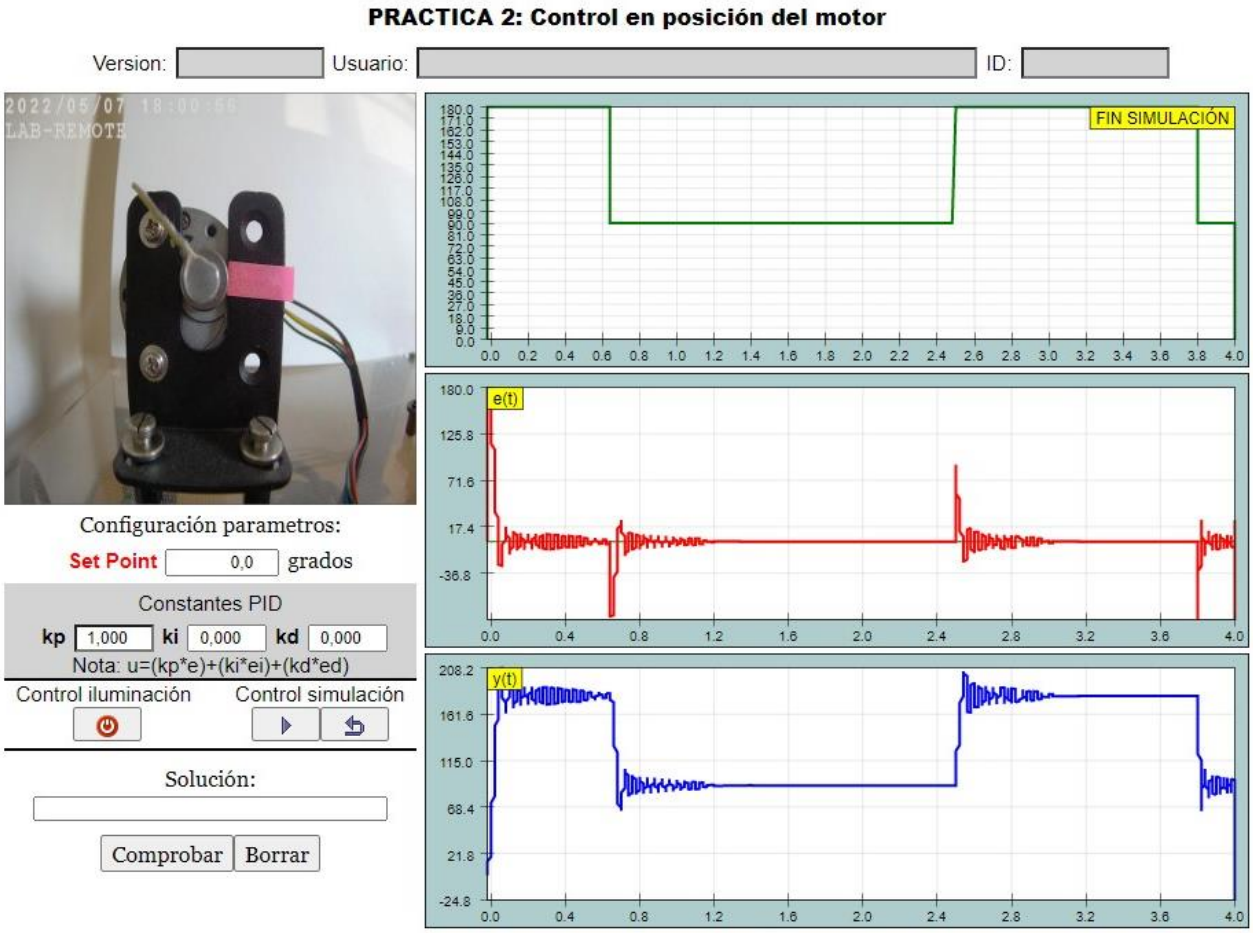

Figura 4-15.- Laboratorio remoto para el Control en posición del motor

En la parte superior justo debajo del título, se encuentra la zona donde aparecerán los datos del alumno que esté realizando la práctica en ese momento, los cuales son cargados a través de la plataforma mediante SCORM.

La parte central del interfaz se encuentra dividida en dos zonas:

En la parte izquierda se aprecia un frame o cuadro, donde se muestra la imagen en tiempo real del motor mientras se realiza la simulación, la cual es obtenida mediante el servidor de imágenes instalado en la planta a través de una cámara IP.

Justo debajo se encuentra el panel de control, donde en primer lugar hay un cuadro de texto para mostrar el Set Point que se está mandando al sistema, a continuación, estará la zona donde se introducirán los valores para las constantes que realizarán el control PID al sistema. Posteriormente es la zona encargada del control de la planta, por un lado, se ha implementado un botón  con el cual se controlará el sistema de iluminación instalado en la planta, y los botones correspondientes a la simulación  para iniciar la misma y reiniciarla.

Por último, está la zona destinada a escribir la respuesta del alumno en función de los resultados obtenidos en la simulación, tras deberá pulsar el botón “Comprobar” para ver el resultado o “Borrar” 

en el caso de tener que rectificar la respuesta.

En la parte derecha se encuentran las gráficas, donde se podrá ir visualizando la evolución de los resultados obtenidos en función de los parámetros introducidos en el panel de control. La primera gráfica muestra el Set Point configurado para la simulación, la gráfica central irá mostrando el error(e) obtenido durante el control PID, mientras en la última gráfica mostrará la señal de salida(y) del motor.

4.3 SERVIDOR DE IMÁGENES

Para la monitorización del motor en tiempo real del laboratorio remoto mientras se interactúa desde el cliente, se ha implementado un servidor de imágenes mediante la cámara IP instalada, la cual han sido descritas sus características en el punto 3.1.6.

Configuración

La propia cámara IP trae un completo software de gestión, mediante el cual se puede configurar multitud de opciones, para configurar correctamente se debe acceder a través desde un navegador web con la dirección IP asignada (192.168.18.60) donde se accederá a la página de inicio o login.

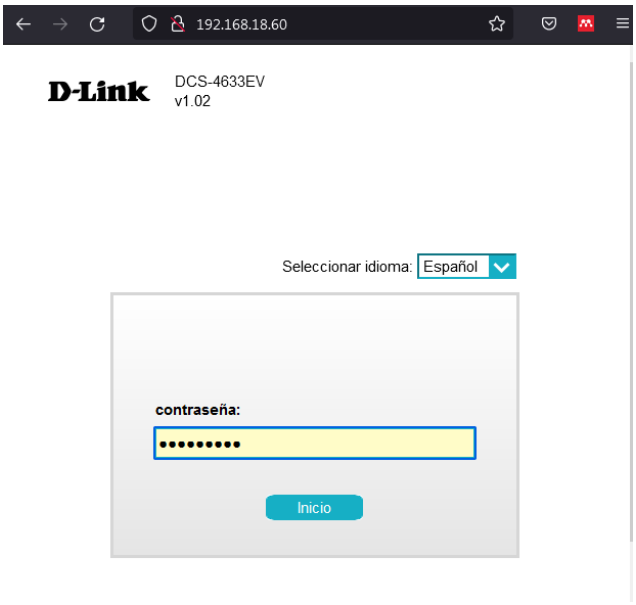

Figura 4-16.- Página login DCS-4633EV

Una vez logueado se configurarán los parámetros necesarios para la visualización de imágenes, en la cámara IP instalada se hará desde el menú Configuración -Secuencia de video, como se muestra en la figura siguiente.

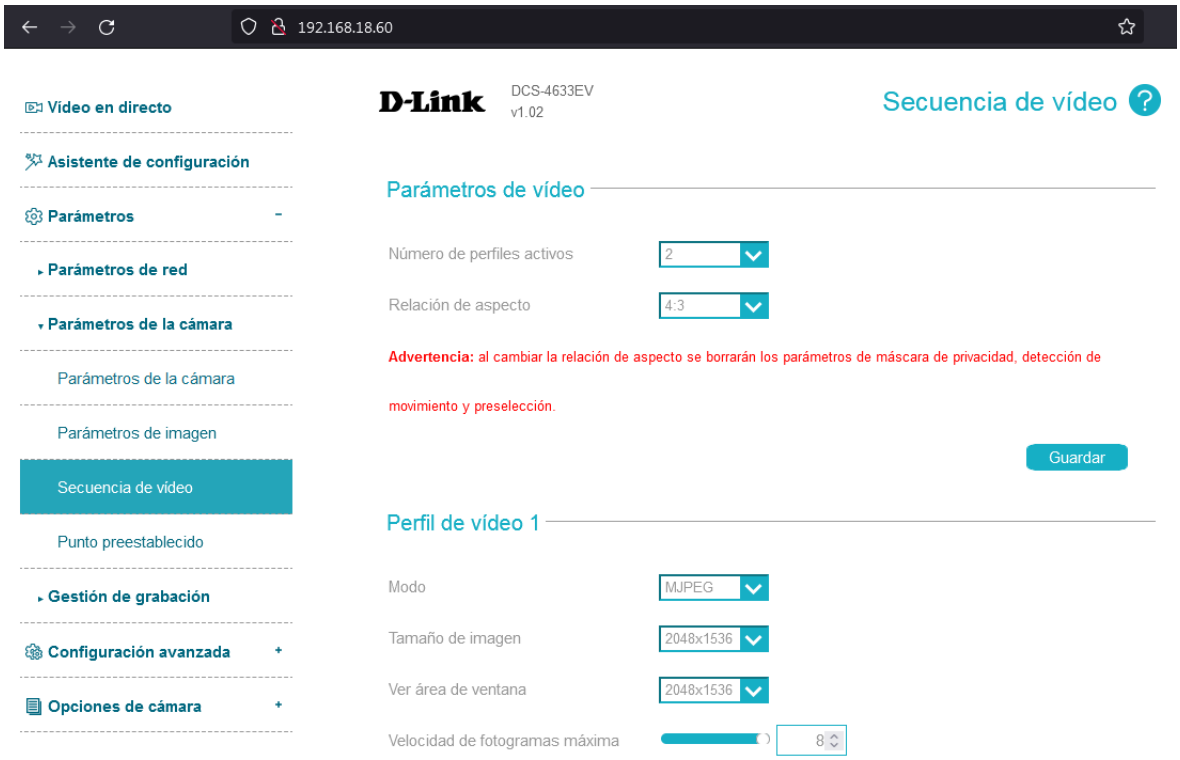

Figura 4-17.- Panel configuración video

4.4 SISTEMA DE CONTROL

La definición de control es la regulación manual o automática de un sistema, una comprensión del sistema como un conjunto de elementos que interactúan para lograr un propósito particular. El conjunto de estas definiciones se denomina sistema de control. En la figura Figura 4-18 se muestra la arquitectura de un sistema de control en bucle abierto, en el cual encontramos el controlador, el actuador y el proceso [27].

Figura 4-18.- Sistema de Control en bucle abierto

Si agregamos la llamada retroalimentación a estos elementos pone al sistema de control en un ciclo cerrado. La diferencia entre los dos sistemas radica en la retroalimentación que consiste en medir la salida del sistema y compararla con el valor establecido o deseado. De esta forma se comprueba si hay errores y mandar la orden hacia el controlador. Esto indicará la acción adecuada al actuador y corregirá este error. La arquitectura del sistema de control de retroalimentación se muestra en la siguiente figura:

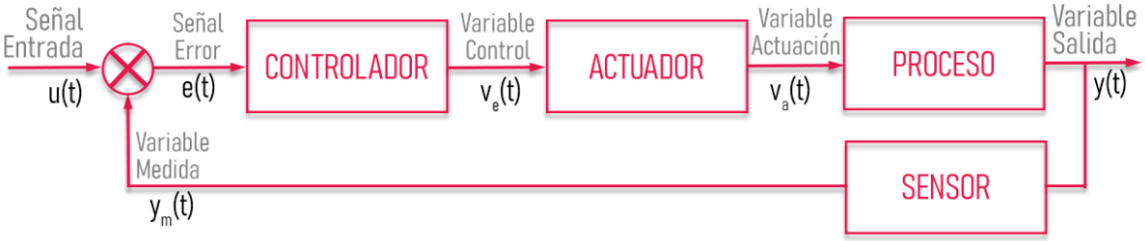

Figura 4-19.- Sistema de Control en bucle cerrado

Controlador PID

El funcionamiento del controlador PID consiste en la acción que realizan sobre la señal de error los tres componentes [28] que a continuación se explican, definiendo como actúan cada una de ellas individualmente:

Control Proporcional (P): Devuelve una señal de control $u(t)$ que es proporcional al error existente $e(t)$, donde K_p es el valor de la ganancia. Este control es capaz de controlar cualquier planta estable, pero presenta error en régimen permanente (off-set).

$$P = K_p \cdot e(t)$$

Ecuación 4-1.- Control Proporcional (P)

Con este control se busca reducir al máximo el error del sistema, ya que, al ser el error grande, la acción de control también es grande por lo que tiende a reducir dicho error.

Efectos derivados por el aumento de la acción proporcional K_p :

- ✓ La velocidad de respuesta del sistema se ve aumentada.
- ✓ El error del sistema disminuye en régimen permanente.
- ✓ La inestabilidad del sistema aumenta.

Control Integral (I): Devuelve una señal proporcional al error acumulado, para lo cual realiza la integral del error existente. K_i es el valor de la ganancia. Tiene como finalidad que el error durante el régimen permanente sea cero cuando toma una referencia constante.

$$I = K_i \cdot \int_0^t e(t) \cdot dt$$

Ecuación 4-2.- Control Integral (I)

Con esta acción de control se busca reducir el error en régimen permanente.

Los efectos de aumentar la acción integral K_i:

- ✓ En régimen permanente disminuye el error del sistema.
- ✓ La inestabilidad del sistema se ve incrementada.
- ✓ La velocidad del sistema aumenta.

Control Derivativo (D): Devuelve una señal proporcional a la derivada del error existente $e(t)$, el cual controlará a través del parámetro K_d, comprueba las variaciones del

error en función del tiempo. Presenta el problema que consiste en que amplifica las señales, por lo que el error también es amplificado.

$$D = K_d \cdot \frac{de(t)}{dt}$$

Ecuación 4-3.- Control Derivativo (D)

Con esta acción de control se consigue estabilizar una respuesta para evitar la sobre oscilación.

Los efectos de aumentar la constante de control derivativa K_d:

- ✓ La estabilidad aumenta en un sistema controlado.
- ✓ La velocidad del sistema disminuye.
- ✓ Se mantiene el error en régimen permanente.

Tras combinar las tres acciones de control conseguimos una acción global del controlador según el algoritmo no interactivo o ideal, tal que:

$$u(t) = (K_p \cdot e(t)) + (K_i \cdot \int_0^t e(t) \cdot dt) + (K_d \cdot \frac{de(t)}{dt})$$

Ecuación 4-4.- Control PID

4.5 INTEGRACIÓN LMS

Cuando la simulación ha sido desarrollada en la herramienta EJS, se exportará el proyecto como un paquete SCORM. Para ello será necesario pulsar el botón secundario del ratón para que se despliegue el menú mostrado como muestra la Figura 4-20.

Tras ello se deberá pulsar la opción "***Create SCORM package***".

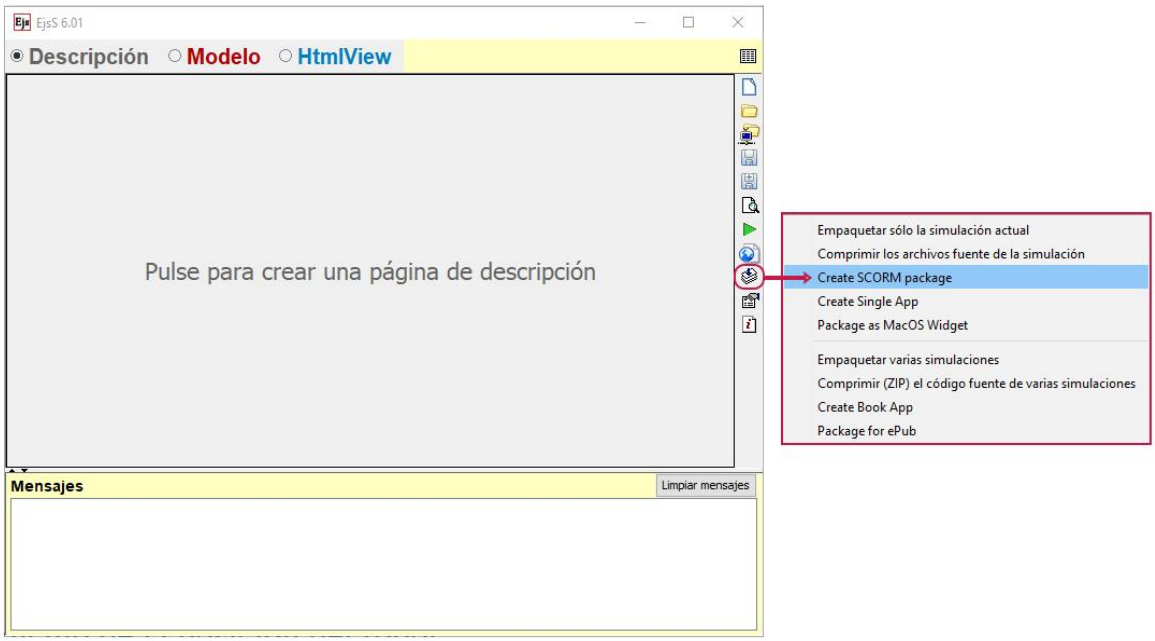

Figura 4-20.- Creación paquete SCORM

A continuación, mostrará una ventana EJsS donde se deberá seleccionar el tipo de versión SCORM, para este TFG se ha optado por usar la versión 1.2 en vez de la 2004 v4, al ser una versión menos compleja y más fácil para trabajar.

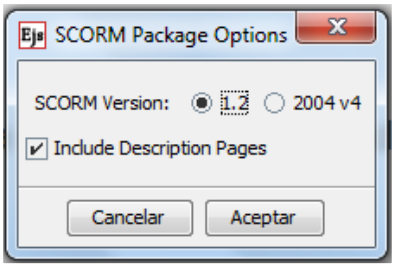

Figura 4-21.- Opciones paquete SCORM

Tras ello, EJsS generará un archivo comprimido **.zip** el cual contendrá todos los archivos necesarios para su posterior integración en la plataforma de Docencia Virtual.

Una vez en la web de Docencia Virtual de la UJA (PLATEA), para poder crear el contenido donde albergar el paquete SCORM creado anteriormente, será necesario realizar los siguientes pasos que a continuación se detallan mediante imágenes:

1.- Primer paso es "***Activar edición***".

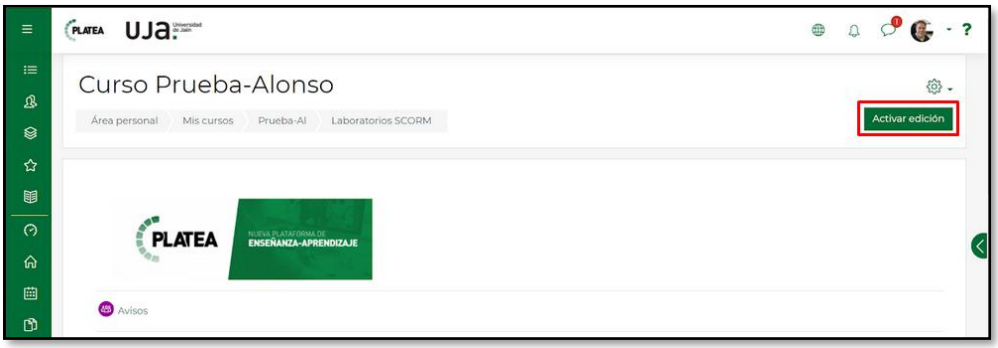

Figura 4-22.- Activar edición SCORM

2.- Seguidamente se pulsará el botón "***Añadir una actividad o un recurso***".

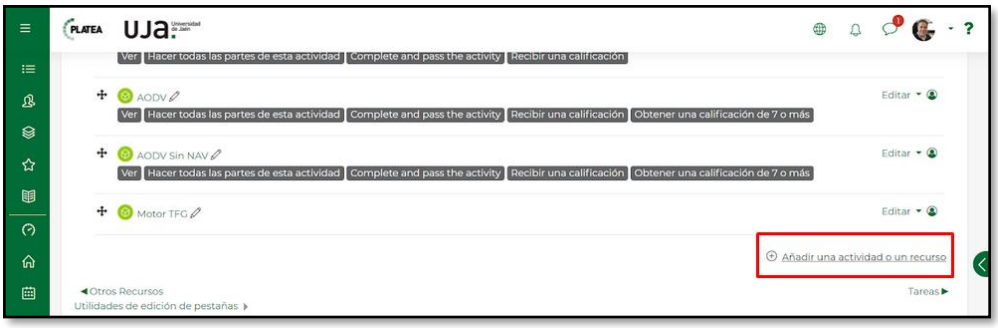

Figura 4-23.- Añadir actividad SCORM

3.- Seleccionar el tipo de actividad/recurso que se va a añadir, en este caso se seleccionara "***Paquete SCORM***".

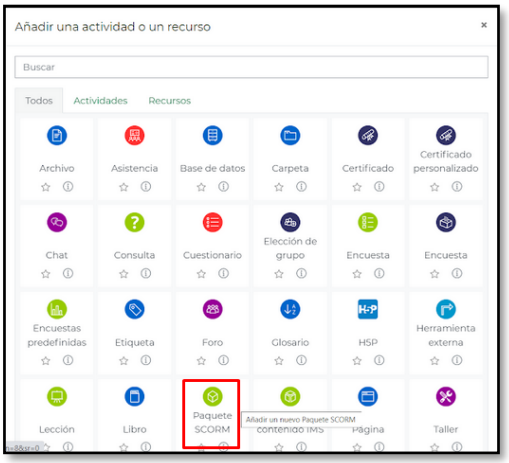

Figura 4-24.- Selección paquete SCORM

A continuación, será el momento de configurar las características específicas del Laboratorio SCORM, como muestra las siguientes capturas, en las que cabe destacar como subir el paquete ***.zip*** creado en EJsS, el cual será añadido en el apartado "***Paquete***" desde el cual se puede arrastrar o seleccionar el mismo.

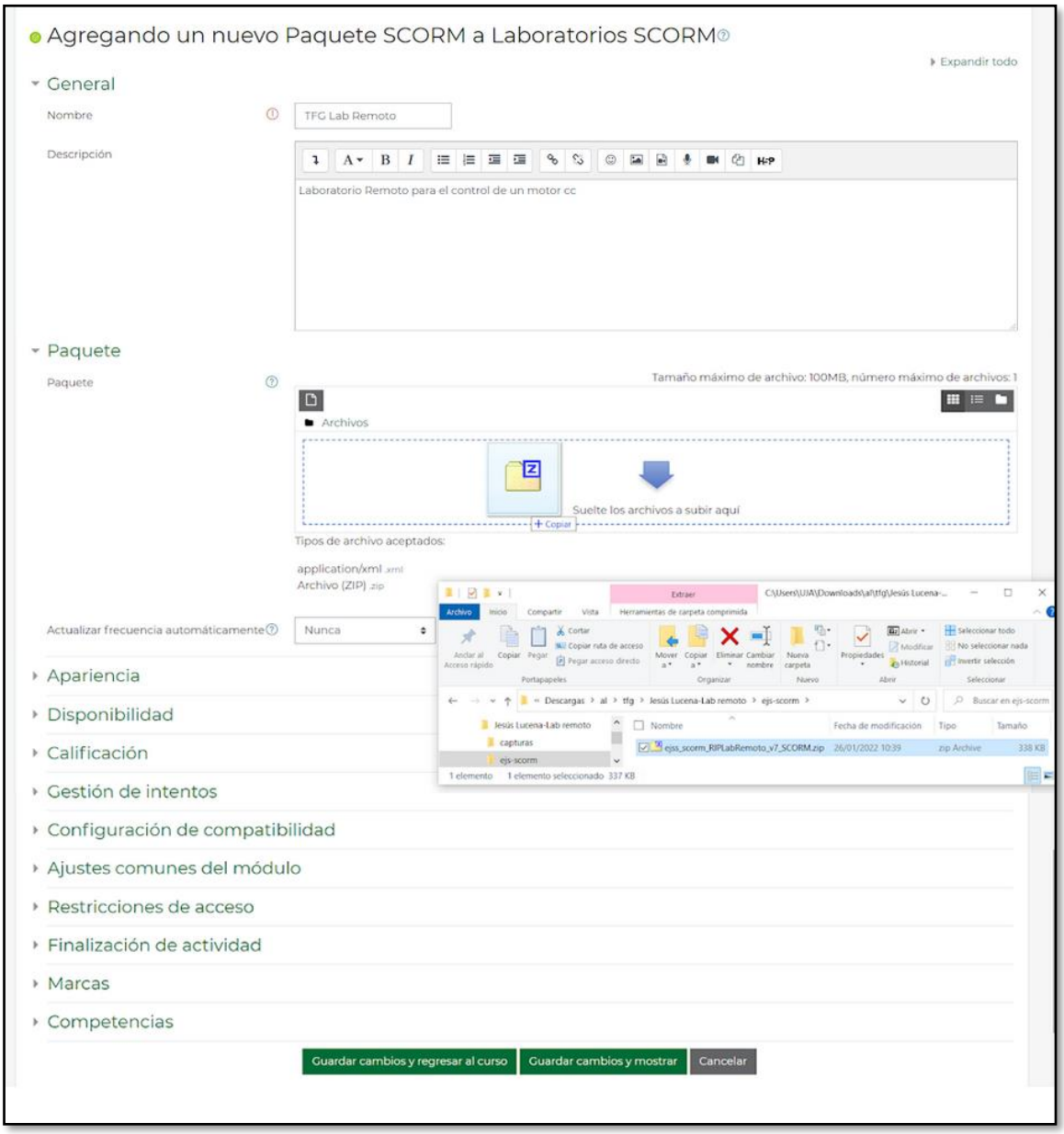

Figura 4-25.- Creación y configuración paquete SCORM.

En esta última imagen se muestra la cantidad de parámetros y configuraciones que pueden ser utilizadas en la plataforma para crear un laboratorio SCORM, en los cuales no se va a profundizar más en este TFG.

5. RESULTADOS

Siguiendo la estructura explicada hasta este punto, se han desarrollado dos laboratorios remotos, uno para identificar la dinámica del motor utilizado en este TFG y otro para controlar mediante un controlador PID la posición del motor, en este punto se comprobarán los resultados mediante la realización de las prácticas con la finalidad de ser utilizadas por los alumnos.

5.1 Identificación de la dinámica del sistema

La primera de las prácticas realizadas para este TFG consiste en la identificación de la dinámica del sistema. En concreto se pretende identificar la dinámica que caracteriza cómo evoluciona la velocidad angular de un motor ante una excitación de tensión al mismo.

Para ello se ha partido del conocimiento del funcionamiento de PWM, en (1) se debe tener señal Duty Cycle(u) que puede tener un valor entre 0 y 100. En terminología de control será la señal de referencia y será facilitada por el profesor (véase Figura 5-1).

A continuación, en (2) se tratará de una señal que va de 0 a 3.3 voltios. Donde sí en (1) hay 0 en (2) habrá 0. Si en (1) hay 100 en (2) será 3.3. Como se ha explicado anteriormente, la finalidad de Driver LM 298 es escalar la tensión por lo que se pasará a 0-12V, tensión de trabajo del motor (3).

El Encoder facilitará a través de (4) 2 pulsos, pulsos tanto del Encoder A como del Encoder B, los cuales como se ha explicado anteriormente están desplazados 90º. Con esta información se podrá sacar el sentido de giro.

Finalmente, a través de un algoritmo software (5) se pasa a calcular la velocidad del motor en vueltas por segundo en función del número de pulsos contados.

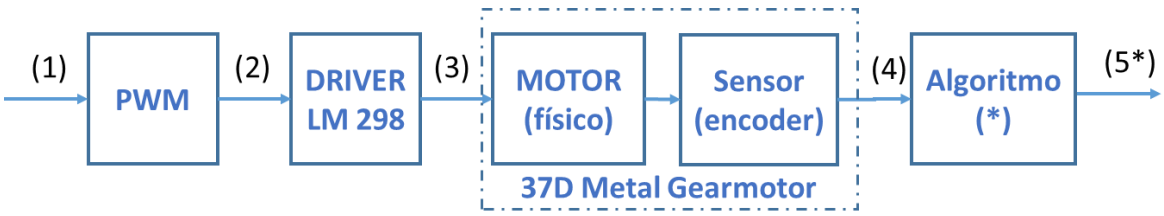

Figura 5-1.- Esquema interno para la identificación de la dinámica del motor.

El alumnado, tendrá únicamente como información la señal de referencia ***u(PWM)***(1) que corresponde con la gráfica superior, así como la velocidad del motor ***w(rad/s)***(5) respectivamente, este caso estará representada gráficamente en la simulación con la gráfica inferior.

Siendo el enunciado de la práctica a realizar es el siguiente:

Enunciado

Dado el esquema de la figura, se pide, identificar la dinámica del motor que representa cómo evoluciona la velocidad del mismo ante una tensión aplicada.

Como ejemplo, en la Figura 5-2 se muestra una simulación realizada en el laboratorio remoto sobre esta práctica:

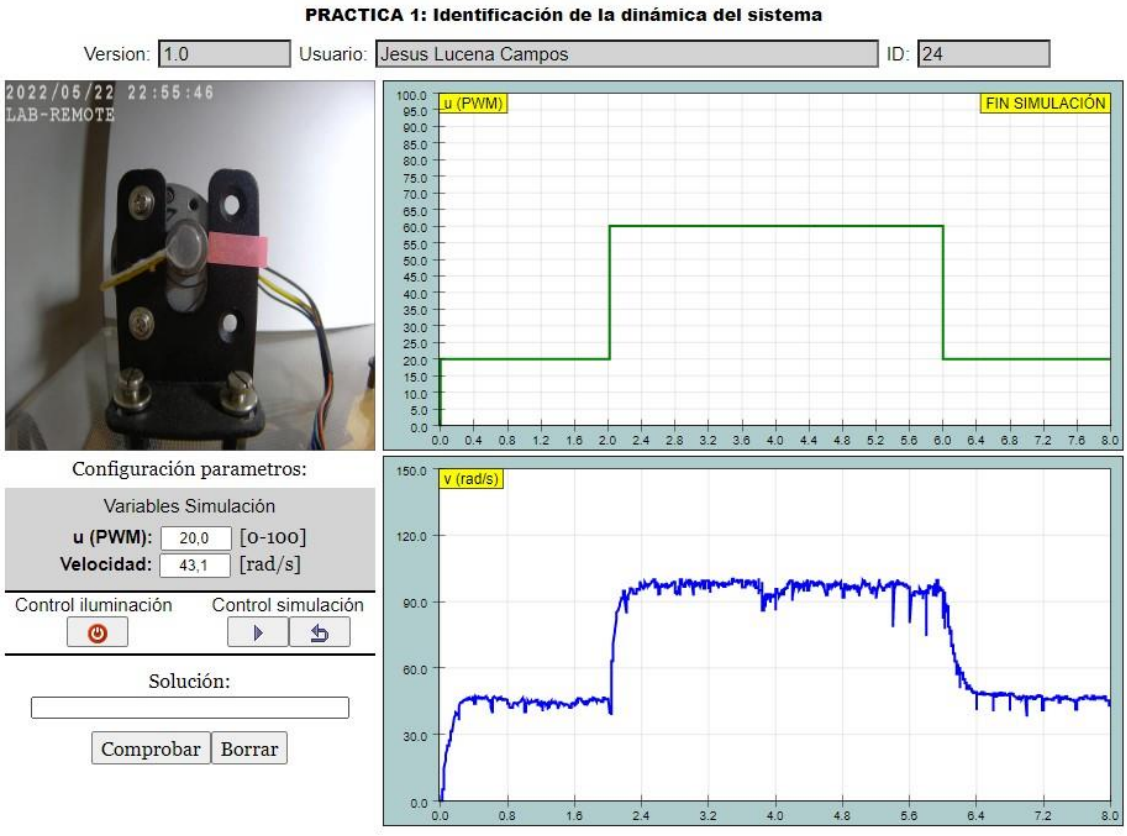

Figura 5-2.- Práctica 1: Identificación de la dinámica del sistema

5.2 Control de la posición del motor

Una vez identificada la dinámica que represente cómo evoluciona la ω ante una tensión aplicada a la entrada, es relativamente sencillo conocer la dinámica que relacione la posición del motor.

Así, la segunda práctica tiene como objetivo identificar dicha dinámica del motor y poder realizar así la sintonía del controlador para que siga una referencia de posición determinada.

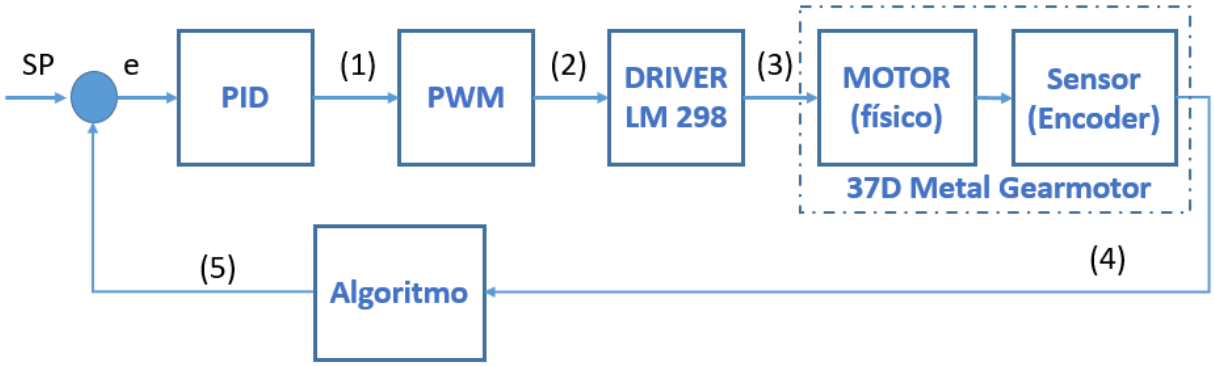

Figura 5-3.- Esquema interno para el control del sistema con un PID.

Se parte de una señal de referencia o Set Point (SP) que puede tener un valor entre 0º y 360º ya que marcará la posición del motor, dicha señal será facilitada por el profesor.

Al tratarse de un sistema de control en lazo cerrado existe una retroalimentación con la señal de salida (y)(5) que a su vez actuará como entrada al sistema para poder ser comparada con la señal de referencia.

Una vez comparadas ambas señales se obtendrá la señal de error (e) será la entrada al controlador PID [29]donde se buscará obtener el estado deseado a la salida (1), que será una señal Duty Cycle (u) con un valor comprendido entre 0 y 100.

A continuación, en (2) se tratará de una señal que va de 0 a 3.3 voltios. Donde sí en (1) hay 0 en (2) habrá 0. Si en (1) hay 100 en (2) será 3.3. Como se ha explicado anteriormente, la finalidad de Driver LM 298 es escalar la tensión por lo que se pasará a 0-12V, tensión de trabajo del motor (3).

El Encoder facilitará a través de (4) 2 pulsos, pulsos tanto del Encoder A como del Encoder B, los cuales como se ha explicado anteriormente están desplazados 90º. Con esta información se podrá sacar el sentido de giro.

Finalmente, a través de un algoritmo software (5) se pasa a calcular la posición del motor en grados en función de los pulsos contados.

Enunciado

Una vez conocida la dinámica del motor que representa la evolución de la velocidad ante una tensión aplicada a la entrada, obtener la dinámica correspondiente a la evolución de la posición ante una tensión aplicada a la entrada es trivial. Por ello, se pide hacer uso de dicha dinámica para identificar y posteriormente calcular los parámetros de sintonía de un controlador PID donde se siga la referencia lo antes posible y con las menores sobre oscilaciones posibles en régimen transitorio.

Como ejemplo, en la Figura 5-4 se muestra una simulación realizada en el laboratorio remoto sobre esta práctica:

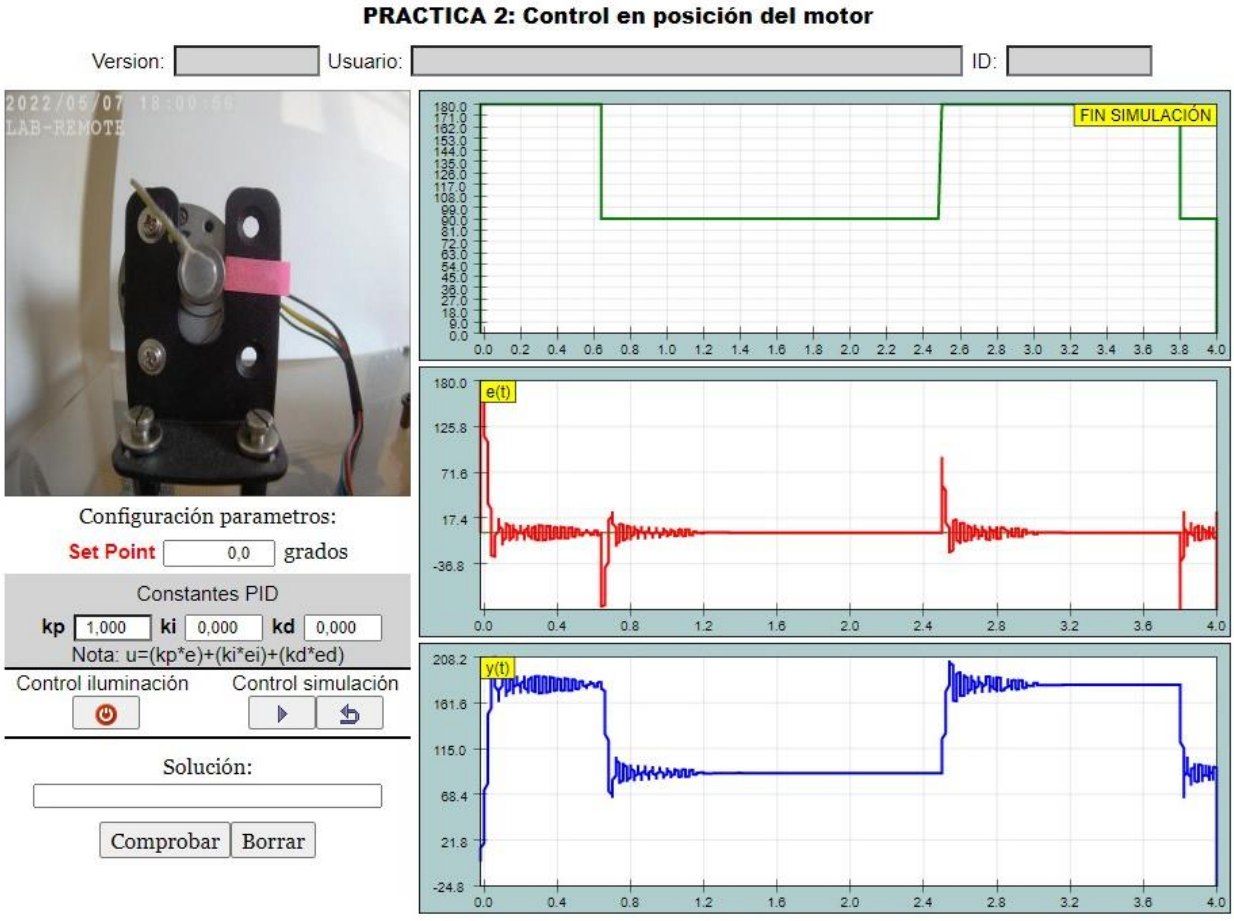

Figura 5-4.- Práctica 2: Control en posición del motor

6. PRESUPUESTO

En este apartado se detalla el presupuesto para llevar a cabo la parte práctica de este TFG.

Dispositivo	Cantidad	Precio unitario	Precio total
Raspberry Pi 4 (Model B) 4Gb	1	62,89 €	62,89 €
Tarjeta SD 32 GB	1	5,25 €	5,25 €
Raspberry Fuente de Alimentación USB-C	1	5,89 €	5,89 €
Motor DC Pololu 70:1 12V con Encoder	1	40,32 €	40,32 €
Puente H L298N	1	4,66 €	4,66 €
Panel Led 60x30mm de 24W 2100 Lm	1	15,76 €	15,76 €
Modulo 4 Relés 5V	1	6,31 €	6,31 €
Cámara IP D-Link DCS-4633EV	1	173,14 €	173,14 €
Fuente Alimentación	2	10,98 €	21,96 €
Kit Cableado Dupont	1	5,45 €	5,45 €
		TOTAL	**341,63 €**

Tabla 6-1.- Presupuesto materiales

En el presente TFG, se han invertido un total de 100 horas, con un importe por hora de 50€. A continuación, se muestra la distribución de tiempo por cada una de las etapas del mismo:

Actividad	Horas	Precio/hora	Precio total
Especificación de requisitos	5	50,00 €	250,00 €
Diseño	20	50,00 €	1.000,00 €
Implementación	70	50,00 €	3.500,00 €
Pruebas	5	50,00 €	250,00 €
		TOTAL	**5.000,00 €**

Tabla 6-2.- Presupuesto mano de obra

Ítem	Precio
Gastos Equipamiento	341,63 €
Gastos Recursos Humanos	5.000,00 €
Base Imponible	5.341,63 €
IVA (21%)	1.121.74 €
TOTAL	**6.463,37 €**

Tabla 6-3.- Presupuesto total

El importe total del proyecto asciende a la cantidad de **SEIS MIL CUATROCIENTOS SESENTA Y TRES EUROS CON TREINTA Y SIETE CÉNTIMOS DE EURO** (6.463,37 €).

7. CONCLUSIONES Y TRABAJOS FUTUROS

En este TFG se presenta una herramienta de aprendizaje para los alumnos, basada en el uso de EJsS, la Raspberry Pi. El primer elemento se utilizó para desarrollar un Front-end de laboratorio HTML5+JavaScript, accesible desde dispositivos móviles y computadoras. La segunda permite ejecutar, en la misma computadora de placa única de bajo costo, el servidor de laboratorio y controlador.

El uso combinado de todos los elementos desarrollados permite:

1. Cerrar un bucle de control en tiempo real sobre la planta seleccionada.
2. Desarrollar una GUI accesible desde cualquier dispositivo desde un navegador y en cualquier momento y lugar.
3. Reducir los retrasos en la comunicación entre los diferentes elementos colocando el servidor y el controlador en la misma computadora y usando un servidor liviano y eficiente para la página web del laboratorio.

Durante el desarrollo de este TFG, surgieron varias ideas que nos permitieron realizar algunas mejoras o completar parte del contenido de las herramientas desarrolladas. Aquí hay algunas ideas para tener en cuenta sobre posibles mejoras en el futuro.:

- ✓ Implementar un sistema para el control de la posición del motor, mediante la detección de la imagen a través de la cámara IP para poder poner el motor a su posición inicial cada vez que un alumno vaya a utilizar el laboratorio.
- ✓ Desarrollar un sistema para el control de la velocidad del motor, con el cual se pueda se controlar desde un laboratorio y los alumnos tengan el control sobre el motor.

8. ANEXOS

8.1 MANUAL INSTALACIÓN Y CONFIGURACIÓN

A continuación, se van a detallar los pasos a seguir para la instalación y configuración realizados en este TFG para el funcionamiento del mismo:

1

Para comenzar es necesario instalar el SO en la Raspberry Pi 4 B, se ha elegido Raspberry Pi OS, para ello se utilizará la aplicación ***Raspberry Pi Imager***, la cual se puede descargar desde la página oficial en el siguiente enlace:

https://www.raspberrypi.com/software/

Donde será necesario ejecutarlo y seleccionar tanto el SO además de la ubicación donde se instalará, en este caso la tarjeta SD.
En el **Apartado 3.1.1** se detallan las principales características tanto de la Raspberry Pi como de Raspberry Pi OS.

2

A continuación, se procederá a configurar la dirección IP estática, en este caso con los parámetros de red facilitados para la instalación del servidor en el laboratorio de la UJA.

Para ello es necesario editar el archivo **/etc/dhcpcd.conf** que se ejecutara desde un terminal desde la Raspberry Pi con la siguiente instrucción[33]:

sudo nano /etc/dhcpcd.conf

Tras ello se abrirá en el terminal el archivo y se deberán configurar los parámetros de red como se muestra en la Figura 8-1.

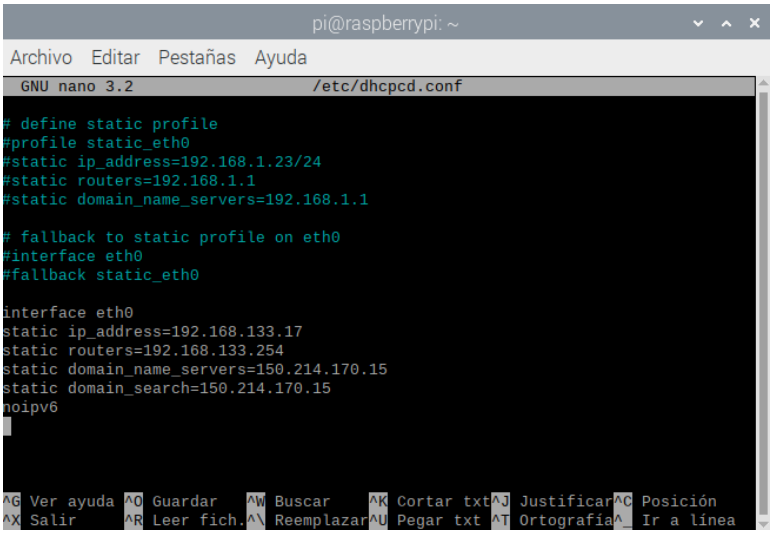

Figura 8-1.- Archivo /etc/dhcpcd.conf

3

Una vez instalado y configurado el SO en la Raspberry Pi, se procederá al conexionado de los distintos elementos del sistema a los pines GPIO, se puede observar detalladamente las conexiones realizadas en la **Figura 4-1**.

4

Posteriormente, se descargará la implementación del servidor RIP desde el repositorio de GitHub:

https://github.com/UNEDLabs/rip-python-server

Puede descargarse desde la URL anterior como un archivo zip, o ser clonado el repositorio desde un terminal con el siguiente comando:

git clone https://github.com/UNEDLabs/rip-python-server

Una vez descargada la implementación del servidor RIP es necesario realizar unas modificaciones en el SO, las cuales se explican en el **Apartado 4.1 – Configuración Servidor RIP**

5

Tras haber instalado el servidor RIP, es necesario configurar los parámetros del servidor. Para ello se configurará la dirección IP y puerto por el que escucha e interactúa, todo ello se hará en el archivo **"AppConfig.py"** como se muestra en la **Figura 4-5**, además se deberá añadir al código las variables tanto *readables* como *writables* que interactúen con EjsS. Dichas variables deben aparecer exactamente igual en el archivo **"RIPLabRemoto.py"**

6

Se ha añadido al servidor una opción para en el caso de que se produjera un corte eléctrico o similar, cuando la Raspberry se reiniciará un script para lanzar el servidor RIP, para lo cual se ha creado un script que se ejecute al inicio [34].

Para ello es necesario agregar un comando al archivo **/etc/rc.local** que se ejecuta al iniciar el sistema, el cual se lanzará desde un terminal con la siguiente línea:

sudo nano /etc/rc.local

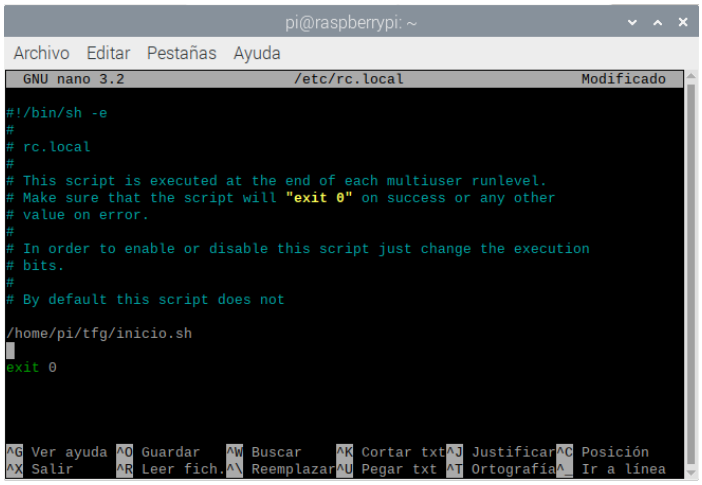

Figura 8-2.- Archivo /etc/rc.local

A continuación, se muestra el contenido con las instrucciones necesarias para la ejecución del servidor:

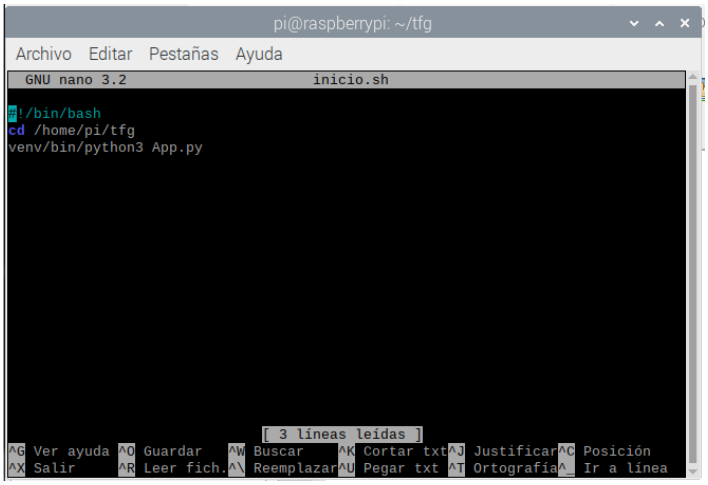

Figura 8-3.- Archivo inicio.sh

Para finalizar hay que de darle permisos de ejecución al fichero que se desea ejecutar:

sudo chmod +x inicio.sh

7

Para la configuración de la cámara IP, será necesario utilizar el software Setup Wizard incluido en el CD de instalación, mediante el cual una vez instalado se ejecutará y mostrará el asistente de configuración [35].

Se mostrarán los datos de las cámaras IP (dirección MAC y dirección IP).

A continuación, se seleccionará la cámara IP y se hará clic en el botón “Wizard” para seguir.

Figura 8-4.- Asistente D-Link

En la siguiente pantalla, se debe introducir el ID del administrador y su contraseña. La primera vez que se inicia sesión, el ID predeterminado es “admin” y la contraseña en blanco.

Se debe pulsar en el botón “Next” para continuar.

Figura 8-5.- Pantalla login D-Link

En esta página se procede a configurar la dirección IP por DHCP o estática.

En el caso de este TFG al estar montado un laboratorio de UJA se ha configurado con la dirección IP estática suministrada por el departamento.

Dirección IP Camara: 192.168.133.16/24

Figura 8-6.- Direccionamiento IP D-Link

Una vez identificada se accederá a través de un navegador escribiendo la dirección IP como muestra **Figura 4-16** para su configuración como se explica en el **Apartado 4.3**.

8

Por último, se debe realizar la configuración de las conexiones seguras SSL tanto en el Servidor RIP como en el Servidor de Imágenes para el correcto funcionamiento desde la plataforma de PLATEA.

Se detalla en el **Apartado 8.2.1** cómo implementar el Servidor RIP el protocolo SSL para realizar las comunicaciones seguras y en el **Apartado 8.2.2** cómo realizar las conexiones HTTPS seguras en el Servidor de Imágenes.

Los puertos reservados por el departamento de Informática de la Universidad de Jaén para los dispositivos utilizados en el laboratorio han sido: PC Servidor (Puerto: 4002) y para la Cámara IP (Puerto:8002)

8.2 SEGURIDAD COMUNICACIONES

Desde el pasado día 1 de septiembre de 2021 está en funcionamiento la nueva plataforma de docencia virtual de la Universidad de Jaén, PLATEA (Plataforma de Enseñanza-Aprendizaje)[30] esta nueva herramienta viene a sustituir a la antigua plataforma de docencia virtual (ILIAS).

Tras este cambio de plataforma, todo el contenido que esté albergado en PLATEA debe cumplir unos requisitos de seguridad para las comunicaciones, por lo cual ha sido necesaria utilizar el protocolo SSL (Secure Sockets Layer) [31], el cual proporciona autenticación, encriptación y desencriptación de datos enviados a través de Internet.

SSL funciona con una combinación entre un certificado público y una clave privada. La clave SSL es mantenida secreta en el servidor. Es usado para el cifrado del contenido que se envía al cliente. El certificado SSL está abierto a todos los que soliciten el contenido. Se puede utilizar para descifrar contenido firmado con la clave SSL correspondiente.

Por ello el servidor ubicado en la Raspberry Pi debe incorporar seguridad SSL. Para ello hay que realizar las siguientes acciones en el servidor RIP y en el servidor de imágenes de la cámara IP.

8.2.1 Configuración segura del servidor RIP

Se puede crear un par de clave y certificado autofirmados con OpenSSL [32] en un único comando:

openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365 -nodes

```
Generating a RSA private key
..........+++++
.....+++++
writing new private key to '/home/pi/tfg/pruebas/key_1.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:JAEN
Locality Name (eg, city) []:Linares
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Universidad de Jaén
Organizational Unit Name (eg, section) []:Grado Telemática
Common Name (e.g. server FQDN or YOUR name) []:Jesús Lucena Campos
Email Address []:jlc00008@red.ujaen.es
```

Figura 8-7.- Generación certificado SSL autofirmado

Tras rellenar los datos solicitados se generan 2 archivos:

- **key.pem**: Contiene la clave privada RSA de 2048 bits.
- **cert.pem**: Certificado SSL autofirmado generado.

Para implementar el protocolo SSL en las comunicaciones del servidor RIP, primero se debe obtener la clave privada y el certificado autofirmado como se ha explicado anteriormente, dichos archivos (***key.pem*** + ***cert.pem***) deben estar ubicados en la carpeta donde esté instalado el servidor RIP, posteriormente en el archivo AppConfig.py se debe configurar 'ssl' con valor a 'True' como se muestra en la imagen, ya que de inicio el protocolo no viene activado.

```
2  config = {
3    # TO DO: The server will listen to host:port
4    'server': {
5      'host': '192.168.18.7',
6      'port': 8080,
7      'ssl': True,
```

Figura 8-8.- Código fuente archivo AppConfig.py

8.2.2 Configuración segura de la cámara IP

Por su parte en el servidor de imágenes se ha configurado también la seguridad en la comunicación segura mediante HTTPS, el cual usa SSL sobre la capa de transporte. Para ello el propio software del servidor de imágenes (Cámara IP D-Link DCS-4633EV) es configurable como se muestra en la siguiente captura del panel de control.

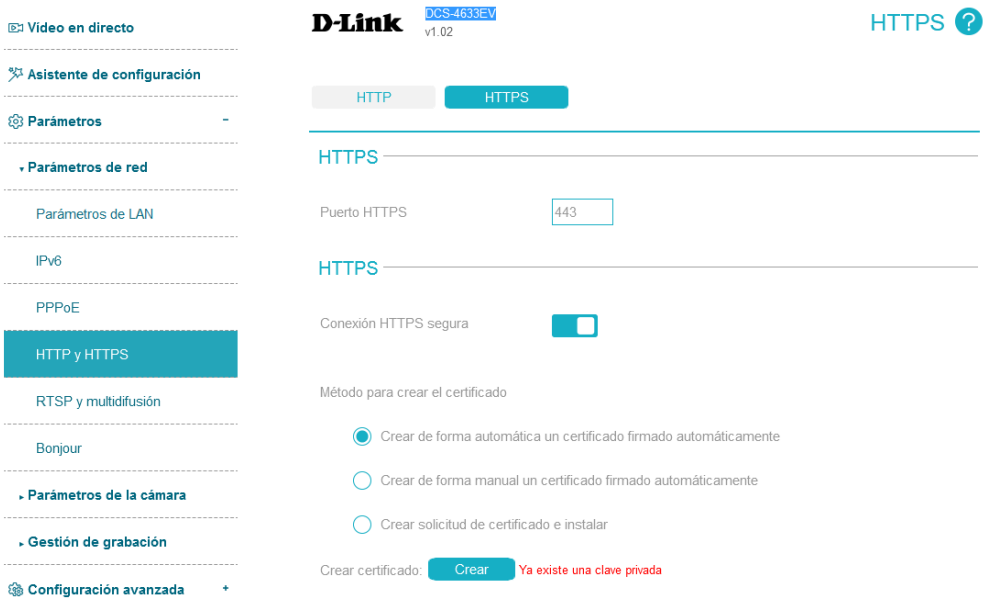

Figura 8-9.- Panel configuración HTTPS

Al igual que el servidor RIP, el certificado configurado en el servidor de imágenes es un certificado autofirmado, por lo cual tampoco ha sido verificado.

Figura 8-10.- Certificado Cámara IP

Como no están verificados los certificados por una autoridad confiable, los navegadores no confiarán en las conexiones y al intentar acceder a dichas URLs

devolverán advertencias como la mostradas en la siguiente figura, por lo que será necesario una conexión previa para poder añadir la excepción antes de poder utilizar el laboratorio.

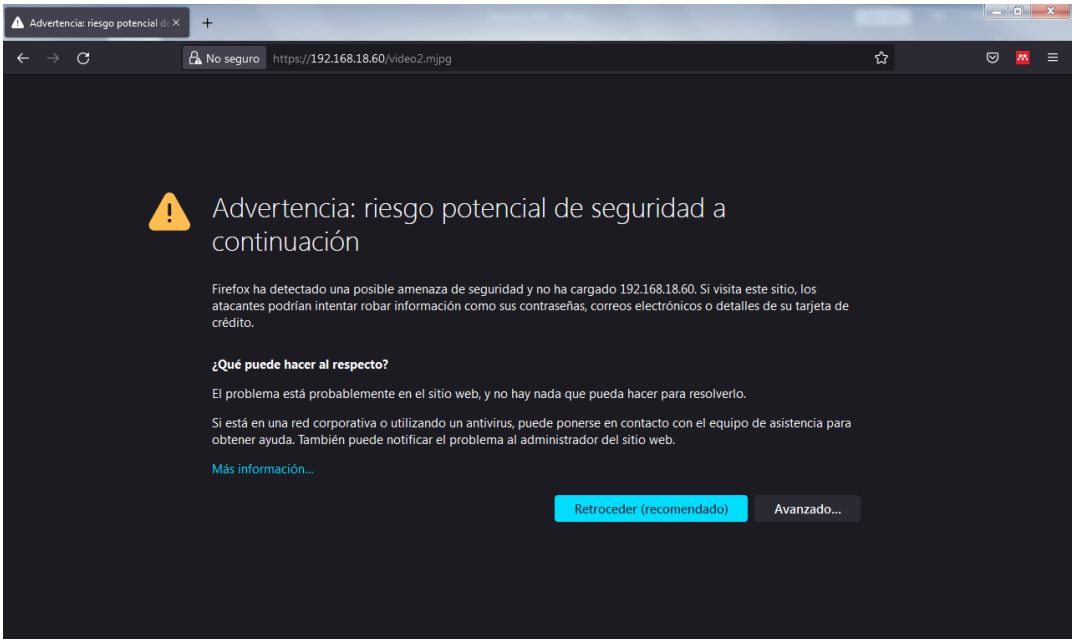

Figura 8-11.- Advertencia Navegador conexión no segura

Para ello se debe acceder desde cualquier navegador a las URLs de los servidores (Servidor RIP y Servidor de Imágenes), pulsar el botón “Avanzado” y posteriormente será necesario pulsar el botón de “Aceptar el riesgo y continuar” como se muestra en la imagen inferior.

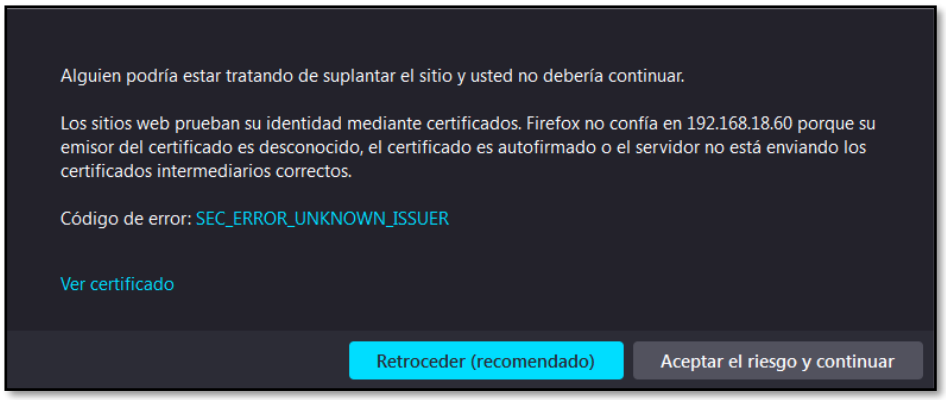

Figura 8-12.- Código de error navegador

8.3 CONFIGURACIÓN EXTERNA UJA

Debido a que el laboratorio remoto ha estado instalado en el domicilio particular, ha sido necesario realizar una serie de cambios en la configuración del Router para poder tener acceso externo, para ello se va a explicar brevemente cómo realizarlo.

Se comenzará por localizar la dirección del router, para ello si no se conoce, será necesario abrir un terminal de Windows(CMD) y ejecutar el comando ***"ipconfig"***, el cual devolverá la siguiente información:

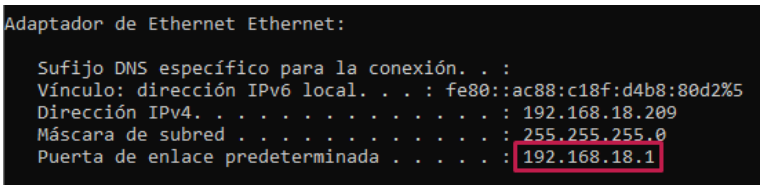

Figura 8-13.- Terminal de Windows.

Se accede desde un navegador y se introducirá como URL la dirección IP indicada como puerta de enlace predeterminada, en este caso ***http://192.168.18.1***, una vez se muestre la pantalla de login como la Figura 8-14 se debe acceder con las credenciales suministradas por el fabricante, en este caso se trata de un Router HUAWEI modelo EG8245H.

Figura 8-14.- Página Login router HUAWEI EG8245H

Una vez logueado en el Router, se accede al apartado superior ***"Forward Rules"***, tras ello se seleccionará en el menú lateral ***"Port Mapping Configuration"***, y se pulsará ***"New"*** si no está creada la regla, en la Figura 8-15 se muestra las reglas creadas.

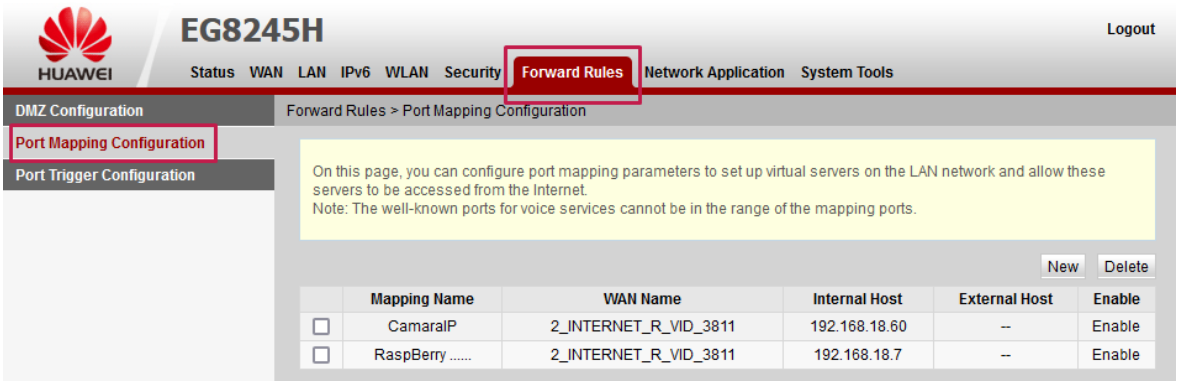

Figura 8-15.- Reglas habilitadas en Router

Como se observa en la Figura 8-16 se debe seleccionar (Internal Host) la dirección IP del dispositivo al cual se quiere abrir la regla, además se indica el rango de puertos (Internal port number) que se quiere abrir, sin olvidar de marcar que la casilla ***"Enable Port Mapping"*** para que la regla esté habilitada.

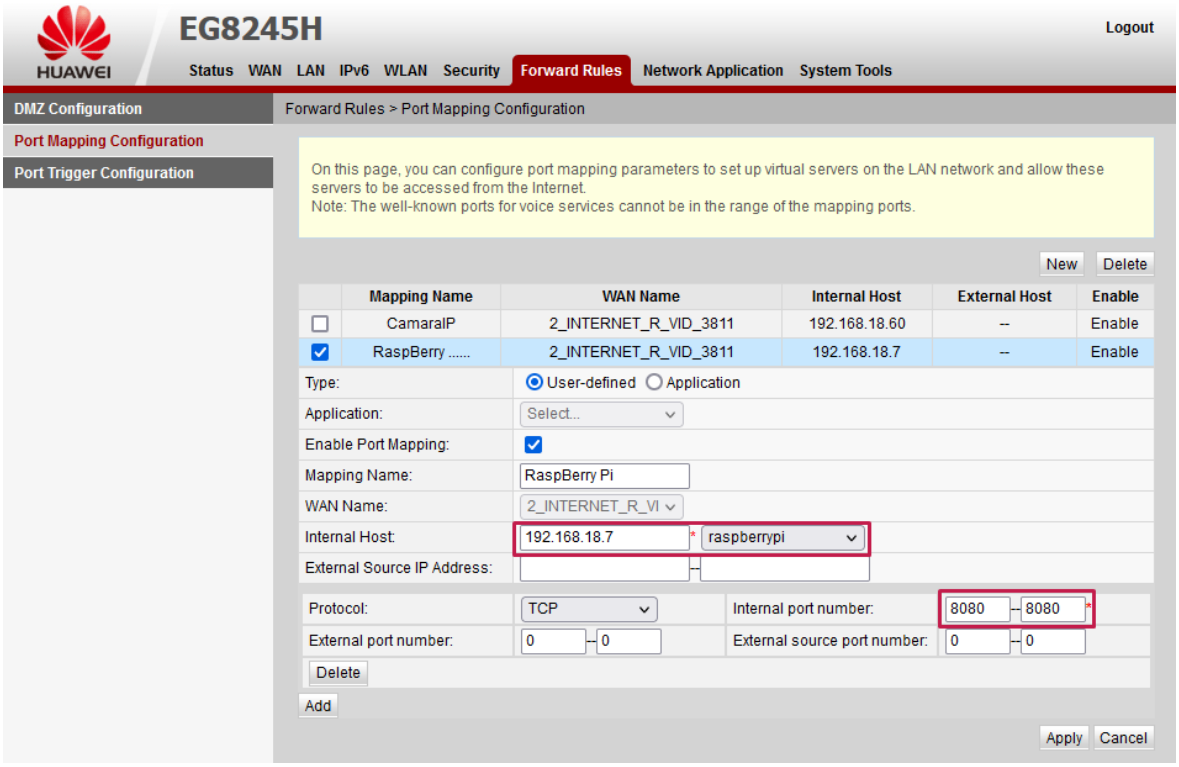

Figura 8-16.- Configuración regla

8.4 ABREVIATURAS Y ACRÓNIMOS

Abreviatura	Significado
API	Application Programming Interfaces
ARM	Advanced RISC Machine
ARMHF	ARM Hard Float
CA	Corriente Alterna
CC	Corriente Continua
CMD	Command Prompt
CPR	Counts Per Revolution
CSS	Cascading Style Sheets
DHCP	Dynamic Host Configuration Protocol
DOM	Document Object Model
EJS	Easy Java Simulations
GNU	GNU Not Unix
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
LMS	Learning Management System
NTIC	Nuevas Tecnologías de la Información y las Comunicaciones
OL	Online Laboratory
PID	Proporcional Integral Derivativo
POE	Power Over Ethernet
PWM	Pulse Width Modulation
RIP	Remote Interoperability Protocol
RL	Remote Laboratory
RPC	Remote Procedure Call
RPM	Revolution Per Minute
SBC	Single Board Computers
SSE	Server-Sent Events
SSL	Secure Sockets Layer
STEM	Science, Technology, Engineering and Mathematics
TCP	Transmission Control Protocol
TTL	Transistor-Transistor Logic
UJA	Universidad de Jaén
UNED	Universidad Nacional Educación a Distancia
Vcc	Voltaje Corriente Continua
VL	Virtual Laboratory
WWW	World Wide World

9. REFERENCIAS BIBLIOGRÁFICAS

[1] B. Balamuralithara and P. C. Woods, “Virtual laboratories in engineering education: the simulation lab and remote lab,” *Computer Applications in Engineering Education*, vol. 17, no. 1, 2009, doi: 10.1002/cae.20186.

[2] L. Gomes and S. Bogosyan, “Current trends in remote laboratories,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 12. 2009. doi: 10.1109/TIE.2009.2033293.

[3] Luis de la Torre, Jesús Chacón, and Dictino Chaos, “Specification of the Remote Interoperability Protocol for Online Laboratories.” Accessed: Oct. 10, 2021. [Online]. Available: https://github.com/UNEDLabs/rip-spec/blob/master/RIP_Specification.pdf

[4] “University Network of Interactive Laboratories.” https://unilabs.dia.uned.es/

[5] F. Esquembre, “Easy Java Simulations: A software tool to create scientific simulations in Java,” *Computer Physics Communications*, vol. 156, no. 2, 2004, doi: 10.1016/S0010-4655(03)00440-5.

[6] “Raspberry Pi - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Raspberry_Pi

[7] “Comparativa y análisis: Raspberry Pi vs competencia.” https://www.comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/

[8] “Raspberry Pi Documentation - Raspberry Pi OS.” https://www.raspberrypi.org/documentation/computers/os.html#gpio-and-the-40-pin-header

[9] “Raspberry Pi OS – Raspberry Pi.” https://www.raspberrypi.com/software/

[10] “37D Metal Gearmotors.” [Online]. Available: www.pololu.com

[11] “MOTORREDUCTOR POLOLU 25Dx65L MM HP 12V 47:1 CON ENCODER 48 CPR – Grupo Electrostore.” https://grupoelectrostore.com/shop/motores/pololu-motores/motorreductor-25dx54l-mm-hp-12v-471-con-encoder-48-cpr/

[12] “Controlador de Motor Dual H-Bridge SeeedStudio L298 - RobotShop.” https://www.robotshop.com/es/es/controlador-motor-dual-h-bridge-seeedstudio-l298.html

[13] “Datasheet L298 Dual H-Bridge Motor Driver,” 2010.

[14] “Panel Led 24W 2.100LM 120º 30x60cm IP40 Blanco.” https://www.innova-led.es/panel-led/panel-led-60x30/panel-led-24w-2100lm-120-30x60cm-ip40-blanco

[15] “Módulo de 4 Relés.”

[16] “Placa De 4 Canales A Relé 5v Con Optoacoplador Y Led Arduino - Robot Electronica.” http://robot.com.ve/product/placa-de-4-canales-a-rele-5v-con-optoacoplador-y-led-arduino/

[17] “Cámara Vigilance domo de 3 megapíxeles a prueba de vandalismo para exteriores”.

[18] “University Network of Interactive Laboratories.” https://unilabs.dia.uned.es/

[19] “JSON-RPC 2.0 Specification.” https://www.jsonrpc.org/specification

[20] “Easy Java Simulations Wiki | Main / EJS Home Page.” https://www.um.es/fem/EjsWiki/

[21] N. Duro, H. Vargas, R. Dormido, S. Dormido, and J. Sánchez, “El sistema de tres tanques: un laboratorio virtual y remoto usando Easy Java Simulations,” 2005.

[22] W. Christian and F. Esquembre, “EjsS Manual,” 2015.

[23] “SCORM: qué es y por qué es clave en el e-learning.” https://www.homuork.com/es/scorm-que-es-y-por-que-es-clave-en-el-e-learning_226_102.html

[24] “GitHub - UNEDLabs/rip-python-server: A server implementation of the RIP protocol in Python.” https://github.com/UNEDLabs/rip-python-server

[25] “Easy Java Simulations Wiki | Main / EJSUserInterface.” https://www.um.es/fem/EjsWiki/Main/EJSUserInterface

[26] U. Complutense *et al.*, “UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA ‘Control de un robot basado en Raspberry,’” 2016.

[27] “Control Automático - Control Automático - Picuino.” https://www.picuino.com/es/arduprog/control-auto.html

[28] K. J. Aström and T. Hägglund, *Control PID avanzado*. 2009.

[29] “Controlador PID - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Controlador_PID

[30] “PLATEA (Plataforma de Enseñanza y Aprendizaje de la Universidad de Jaén): Entrar al sitio.” https://platea.ujaen.es/login/index.php

[31] “Seguridad de la capa de transporte - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Seguridad_de_la_capa_de_transporte

[32] “Cómo crear un certificado SSL autofirmado para Apache en Ubuntu 18.04 | DigitalOcean.” https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04-es

[33] “Configurar IP estática en Raspberry Pi.” https://www.luisllamas.es/raspberry-pi-ip-estatica/

[34] “Tutoriales - Arranque automático en Raspbian.” https://www.programoergosum.es/tutoriales/arranque-automatico-en-raspbian/

[35] “Configuración De La Cámara - D-Link DCS-4633EV Quick Installation Manual [Page 42] | ManualsLib.” https://www.manualslib.com/manual/1532407/D-Link-Dcs-4633ev.html?page=42#manual