ITCA FEPADE
TÉCNICOS E INGENIEROS

ISBN: 9978-9961-39-82-6 (Impreso)
ISBN: 9978-9961-39-90-1 (E-Book)

INFORME FINAL DE INVESTIGACIÓN

DISEÑO E IMPLEMENTACIÓN DE PLATAFORMA DE TELEINGENIERÍA PARA PRÁCTICAS EN TIEMPO REAL DE LABORATORIO A DISTANCIA DE CONTROL DE PROCESOS INDUSTRIALES

APLICACIÓN EN LABORATORIO DE ELECTRÓNICA DE ITCA-FEPADE SEDE CENTRAL

DOCENTE INVESTIGADOR PRINCIPAL:
TÉC. JUAN JOSÉ GUEVARA VÁSQUEZ

DOCENTE COINVESTIGADOR:
ING. JUAN JOSÉ CÁCERES CHIQUILLO

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ITCA-FEPADE SEDE CENTRAL

ENERO 2022

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

ITCA EDITORES

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

Comisión de Acreditación de la Calidad Académica
INSTITUCION ACREDITADA

ISBN: 978-99961-39-82-6 (Impreso)
ISBN: 978-99961-39-90-1 (E-Book)

INFORME FINAL DE INVESTIGACIÓN

DISEÑO E IMPLEMENTACIÓN DE PLATAFORMA DE TELEINGENIERÍA PARA PRÁCTICAS EN TIEMPO REAL DE LABORATORIO A DISTANCIA DE CONTROL DE PROCESOS INDUSTRIALES

APLICACIÓN EN LABORATORIO DE ELECTRÓNICA DE ITCA-FEPADE SEDE CENTRAL

DOCENTE INVESTIGADOR PRINCIPAL:
TÉC. JUAN JOSÉ GUEVARA VÁSQUEZ

DOCENTE COINVESTIGADOR:
ING. JUAN JOSÉ CÁCERES CHIQUILLO

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ITCA-FEPADE SEDE CENTRAL

ENERO 2022

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA

ITCA EDITORES

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

629.895
G939d slv

Guevara Vásquez, Juan José, 1978 -
Diseño e implementación de plataforma de teleingeniería para prácticas en tiempo real de laboratorio a distancia de control de procesos industriales [recurso electrónico] : aplicación en laboratorio de electrónica de ITCA-FEPADE sede central / Juan José Guevara Vásquez, Juan José Cáceres Chiquillo. -- 1ª ed. -- Santa Tecla, La Libertad, El Salv. : ITCA Editores, 2022.
1 recurso electrónico (70 p. : il. col. ; 28 cm.)

Datos electrónicos (1 archivo, formato pdf, 11 Mb). -- https://www.itca.edu.sv/produccion-academica/

ISBN : 978-99961-39-82-6 (Impreso)
ISBN : 978-99961-39-90-1 (E-Book, pdf)

1. Control electrónico. 2. Control remoto – Equipo. 3. Control del proceso. I. Cáceres Chiquillo, Juan José, 1978- coaut. II. Título.

Autor
Téc. Juan José Guevara Vásquez

Coautor
Ing. Juan José Cáceres Chiquillo

Docentes Participantes
Téc. Carlos Geovany Meléndez Molina
Ing. Elvis Moisés Martínez Pérez
Inga. Rina Elizabeth López de Jiménez

Año 2022

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

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

CONTENIDO

1. INTRODUCCIÓN

En el control de procesos electrónicos industriales es muy importante que tanto docentes como estudiantes dispongan de equipos de entrenamiento que bajo un entorno controlado puedan realizar prácticas, ya que permite aplicar los conceptos estudiados en la teoría de manera que se puedan solidificar las competencias.

ITCA-FEPADE dispone de entrenadores para el control de distintos procesos que se han utilizado a través de los años en cátedras de nivel técnico y de ingeniería, pero se ha tenido la limitante que solo se dispone de un entrenador para cada proceso industrial específico, lo cual reduce el tiempo de entrenamiento, a esto hay que agregar que uno de los efectos de la pandemia por COVID-19 es que se ha tenido que reducir aún más la presencialidad de los estudiantes y docentes por motivos de bioseguridad, limitando el acceso a los entrenadores.

Es por esta razón que se decidió realizar un proceso de reingeniería a uno de los entrenadores de Control de Procesos de Fluidos FPC con los que cuenta el laboratorio J-101 de ITCA-FEPADE Sede Central, con el objetivo de diseñar una Plataforma de Teleingeniería que permita controlar a distancia un entrenador de control de procesos industriales de temperatura para realizar prácticas de laboratorio de forma remota, con resultados y monitoreo en tiempo real. El componente de control del equipo es comandado de forma remota a través de una aplicación que se accede por medio de un navegador Web moderno, ya sea desde una computadora de escritorio o dispositivo móvil, de manera que docentes y estudiantes puedan realizar las prácticas de laboratorio de forma programada y segura sin necesidad de estar frente al equipo.

Se integró el hardware y software necesarios para la administración de forma remota del entrenador de control de temperatura. Se implementó un mecanismo de validación y administración de usuarios vía Web del laboratorio a distancia, con los mecanismos de seguridad necesarios para asegurar la integridad y disponibilidad del laboratorio de forma remota.

Una solución de hardware como la diseñada en este proyecto de investigación, puede llegar a tener un costo hasta cinco veces superior al utilizado, con la desventaja que por lo general se trabaja con una plataforma cerrada y el agregar más prestaciones a la solución requiere siempre de una mayor inversión.

Con los resultados de este proyecto se pueden realizar prácticas de laboratorio a distancia en entrenadores que tradicionalmente se manipulan únicamente de forma presencial. La Plataforma de Teleingeniería diseñada es única en el país, ya que no es un simulador, sino que todas las acciones de control que realiza el usuario se hacen en el entrenador físico, pero a distancia. Además, la incorporación del video en tiempo real mejora la experiencia del usuario, ya que puede observar de forma efectiva que todas sus acciones se ejecutan directamente en el entrenador como si estuviera en una práctica presencial.

El sistema o plataforma como tal, es una combinación de dispositivos electrónicos discretos, de potencia y control, que junto a una aplicación desarrollada utilizando lenguajes de programación orientados a la Web, logra acercar a los usuarios una experiencia real de trabajo de laboratorio sin importar la distancia física existente.

Como la Plataforma de Teleingeniería integra diversos sistemas y tecnologías, está sujeta a un proceso de depuración y mejora continua por lo que se deben realizar ajustes, correcciones de errores y adecuaciones que mejoren la experiencia del usuario.

La Plataforma de Teleingeniería diseñada demuestra que es posible modificar la circuitería electrónica de los entrenadores de control de procesos industriales que se usan en prácticas de laboratorio presenciales y hacer que éstos sean gestionados a distancia. Por lo que se pueden expandir las prestaciones de la plataforma existente, incorporando módulos de software y hardware a otros entrenadores que se deseen controlar a distancia.

2. Planteamiento del Problema

2.1. Definición del Problema

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

Este proyecto provee una solución que permite optimizar el tiempo de acceso a los equipos para las prácticas de control de procesos industriales y registrar las actividades que los estudiantes lleven a cabo para retroalimentar al docente oportunamente.

2.2. Antecedentes / Estado de la Técnica

A nivel internacional ya existen laboratorios que permiten la simulación y observación de prácticas, todos ellos son resultados de proyectos de investigación. Las universidades españolas Complutense de Madrid y Universidad Nacional de Educación a Distancia UNED de Costa Rica, poseen experiencia en esta área y han publicado los resultados de sus investigaciones en revistas y conferencias de ámbito científico. El Instituto Tecnológico de Monterrey ha implementado este tipo de laboratorios en electricidad, electrónica y automatización. En El Salvador no se tienen registros sobre plataformas de Teleingeniería en Instituciones de educación superior ni patentes relacionadas, y posiblemente tampoco en la región Centroamericana.

En los laboratorios de Teleingeniería existentes se han utilizado diversas tecnologías, cada una de ellas acorde a las necesidades puntuales y experticia de las instituciones que las diseñaron. Sin embargo, tienen en común la integración de servicios de Internet, seguridad informática e industrial y control de acceso a los estudiantes.

Como parte del programa de investigación del año 2020, la Escuela de Ingeniería Eléctrica y Electrónica de ITCA-FEPADE Sede Central realizó un estudio de la estructura de las plataformas de Teleingeniería por lo que se cuenta con la experiencia que permita determinar que tecnologías deben integrarse para construir un sistema de prácticas de laboratorio de este tipo. Además, se ha estudiado y simulado el comportamiento del entrenador de control de procesos de temperatura que servirá como base para la construcción de la plataforma que se propone.

2.3. Justificación

La pandemia por COVID-19 ha sido sin duda alguna un fenómeno que ha afectado a la sociedad y nos ha obligado a cambiar los sistemas de educación tradicional por los modelos de formación a distancia, esto con el objetivo de disminuir en lo posible el contacto físico con otras personas y así reducir significativamente el riesgo de contagio.

En este contexto, mediante la incorporación de herramientas de acceso y control remoto al entrenador de control de temperatura del laboratorio J-101, el instructor podrá programar eficientemente el acceso a los estudiantes para el desarrollo de las prácticas de laboratorio. Los estudiantes tendrán una experiencia real ya que, si bien no estarán físicamente frente al entrenador, se proveerá de videocámaras que proporcionarán retroalimentación visual, esto junto con la incorporación de un panel de control electrónico virtual proporcionarán información en tiempo real a los estudiantes y les permitirá interactuar oportunamente y desarrollar adecuadamente sus prácticas de laboratorio.

De esta manera, este proyecto de investigación contribuye no solamente a optimizar los recursos con los que cuenta la institución haciendo un uso efectivo de ellos y mejorando considerablemente el aprendizaje de los estudiantes, sino que también permite disminuir el riesgo de contagio por enfermedades altamente contagiosas como el COVID-19. La institución dará un significativo salto tecnológico en educación ya que la tendencia es conectar las islas (entrenadores) a Internet para hacerlos accesibles a estudiantes, docentes e investigadores que se encuentran en regiones lejanas.

3. Objetivos

3.1. Objetivo General

Diseñar una Plataforma de Teleingeniería que permita controlar a distancia un entrenador de control de procesos industriales de temperatura para realizar prácticas de laboratorio de forma remota con resultados y monitoreo en tiempo real.

3.2. Objetivos Específicos

1. Integrar el hardware y software necesarios para la administración de forma remota del entrenador de control de temperatura del laboratorio J-101.
2. Implementar un mecanismo de validación y administración de usuarios vía web del laboratorio a distancia.
3. Implementar los mecanismos de seguridad necesarios para asegurar la integridad y disponibilidad del laboratorio de forma remota.

4. Hipótesis

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

5. Marco teórico

5.1. El Control de Procesos

El control de procesos toma en cuenta la medición y el análisis de las variables que determinan el funcionamiento de un proceso, así como la toma de decisiones y la ejecución de acciones de control para gobernar dicho proceso[1]. Las plantas industriales se diseñan para llevar a cabo procesos productivos que permitan transformar la materia prima y los insumos en bienes cuyo valor lo establece el mercado.

En una economía globalizada y de alta competencia, la mejor manera de lograr un efectivo control de procesos es mediante la incorporación de la tecnología electrónica integrada a técnicas de control automático de procesos. Esta tecnología está basada en la adquisición de datos extraídos directamente del proceso para ser analizados con el fin de tomar decisiones de control con un mínimo de intervención humana. La ventaja de obtener información directa del proceso es que tan pronto se produce un cambio, este es registrado y utilizado para mantener el proceso bajo control.

En los controles de procesos que se utilizan actualmente, se integran recursos de hardware y software cuyo diseño y fabricación sigue estándares industriales de propósito general tales como: Controladores Lógicos Programables (PLC), Sistemas de Control Distribuidos (DCS) y Controladores Avanzados de Procesos (PAC). Adicionalmente, se pueden utilizar tecnologías que involucran especialistas en programación tales como lenguaje C, VHDL y LabVIEW.

Un sistema es un conjunto de elementos interrelacionados entre sí, los cuales se caracterizan por poseer parámetros inherentes que los definen y que muestran condiciones físicas asociadas, susceptibles de evolucionar con el tiempo. Estos parámetros son específicos de cada elemento y son considerados como constantes, por lo que se les llama parámetros del sistema.

Por otra parte, las condiciones físicas que cambian con el tiempo son las que determinan el estado del sistema por lo que se expresan por medio de las denominadas variables del sistema. La magnitud y evolución de dichas variables vienen regidas por las leyes físicas que las gobiernan.

En teoría de control, cada uno de los componentes elementales en los cuales puede descomponerse un sistema, recibe el nombre de bloque y puede representarse gráficamente por medio de un rectángulo al cual se le agrega una flecha de entrada y otra de salida en sus costados (fig. 1).[2]

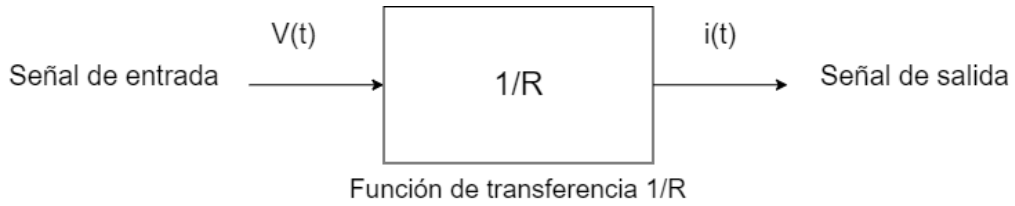


Fig. 1: Representación gráfica de un bloque.

La señal de entrada corresponde a la variable física o variable de entrada, esta será convertida o manipulada por el bloque para producir una señal de salida o variable de salida. Cada uno de los bloques que forman parte de un sistema, se unen mediante líneas que con sus flechas indican la dirección del flujo de la información o señales que circulan a lo largo del sistema. El diagrama de bloques es una forma convencional de representar gráficamente las interrelaciones entre las variables significativas del sistema, así como las características de los componentes que lo forman.

Un proceso es un conjunto de equipos o dispositivos mecánicos, eléctricos, físicos, químicos, o de cualquier índole, dispuestos de tal manera que puedan en su conjunto las actividades necesarias para alcanzar un objetivo. Un sistema controlado está formado por el proceso y el sistema de control.

La regulación o controla automático en lazo cerrado consiste en sustituir la acción del hombre por un dispositivo llamado controlador o regulador, de manera que al conjunto de los componentes que llevarán a cabo el control de un proceso se llama sistema de control automático.

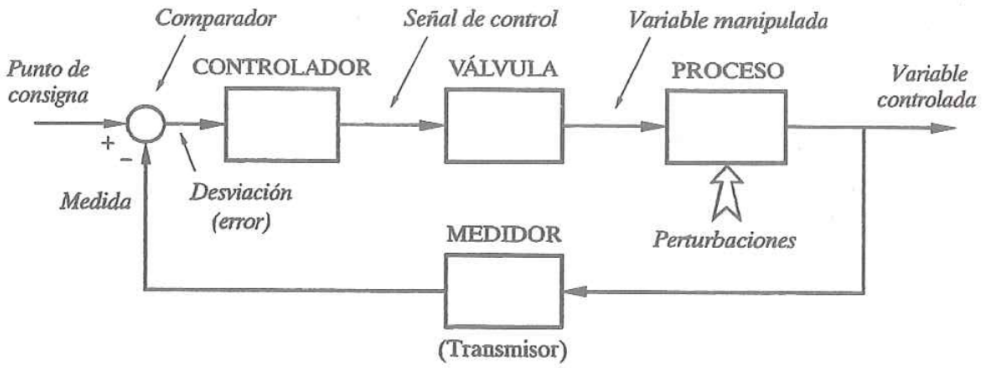


Fig. 2: Disposición básica de un proceso controlado automáticamente.

En la figura 2, se muestra un proceso que se controla automáticamente en lazo cerrado, en él puede observarse que el flujo de las señales se cierra a sí mismo, y que el sistema tiene como señal de entrada el punto de consigna (referencia o set point) y como señal de salida la variable controlada. El dispositivo comparador entre las señales de consigna y medida suele formar parte del controlador.

La realimentación consiste en tomar una medida de la variable controlada a través de sensores y enviarla hacia el controlador para que genere la señal de error o desviación por medio de la comparación con la señal de referencia, a partir de la señal de error el controlador genera una señal de control o de ajuste de manera que en el ejemplo de la figura 2, se ajuste apropiadamente la posición de la válvula para que a la salida del proceso se alcancen condiciones más próximas a las deseadas.

Es entonces, objetivo del control automático la forma como, a partir de la aparición de una desviación, se efectuará el restablecimiento o recuperación del valor deseado en la variable manipulada. A esta forma de respuesta se le denomina respuesta transitoria. Mientras que, una vez que se ha extinguido el transitorio (perturbación) y se ha alcanzado el punto deseado se genera una respuesta transitoria o estacionaria. Sin embargo, debido a las características de todo proceso real, no es técnicamente posible lograr una regulación ideal. Así pues, la variable controlada puede efectuar la aproximación al punto de referencia de manera diferente a la ideal y de esta manera se pueden distinguir formas no satisfactorias de hacerlo:

- Aproximación demasiado lenta o errática.
- No estabilizarse en el valor de referencia.
- Rebasamiento transitorio excesivo del valor de referencia.
- Restablecimiento al valor de referencia después de excesivas oscilaciones amortiguadas alrededor del mismo.
- Presentar oscilaciones mantenidas crecientes en amplitud, lo cual puede afectar algún componente del sistema.
- Una combinación de las formas anteriores.

La recuperación del sistema ante perturbaciones debe efectuarse siempre con la máxima rapidez, el mínimo de rebasamiento y con una desviación permanentemente nula [3].

5.2. Sistemas Embebidos

Un sistema embebido es una computadora basada en microprocesador que integra hardware y software para realizar una determinada función junto a una determinada cantidad de dispositivos mecánicos y eléctricos ya sea como un sistema independiente o como parte de un sistema muy grande. En el núcleo hay un circuito integrado diseñado para realizar operaciones de cálculo en tiempo real.[4]

Los sistemas embebidos no están diseñados para que el usuario final pueda programarlos los cual es una diferencia importante respecto a una computadora personal, si es posible realizar cambios en la configuración de la funcional del sistema, pero no se puede efectuar mediante el reemplazo o modificación del software o firmware de este. Es decir, que se pueden realizar los ajustes especificados al momento de crear el software del sistema.

Los sistemas embebidos están fundamentalmente formados por microcontroladores de diversa gama de robustez y poder de proceso, hoy en día son muy populares los procesadores de Microchip (PIC), AVR (ATMEGA) y ARM.

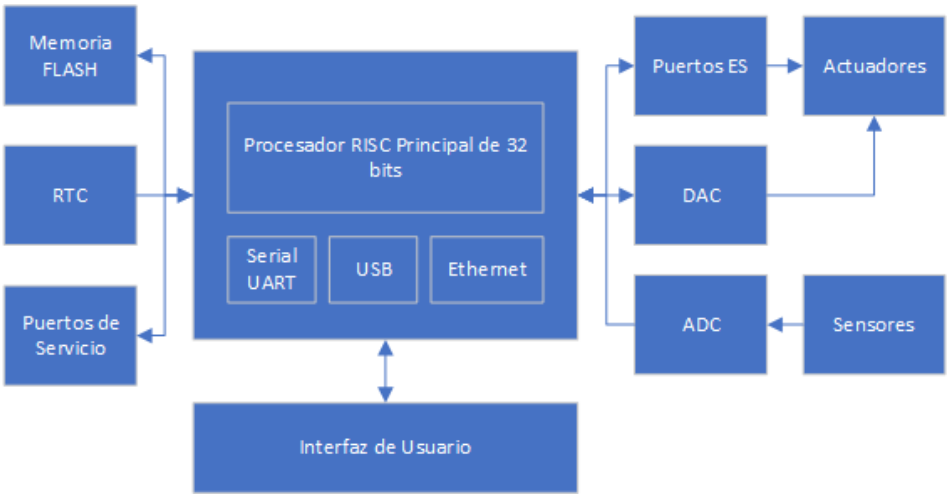


Fig. 3: Diagrama de bloques de un sistema embebido típico.

Protocolo de Comunicación TCP

TCP (Protocolo de Control de Transmisión), es el protocolo de transporte más utilizado desde hace más de 30 años. Es el estándar 7 de Internet definido en la RFC 793 [5] y su identificador de protocolo es el 6. Por lo general, casi todos los servicios que se ejecutan por sobre Internet que utilizan usuarios emplean a TCP como protocolo de transporte (correo electrónico, intercambio de archivos, etc.).

TCP es un protocolo tan extendido por que proporciona un servicio confiable, orientado a conexión para aplicaciones que se dedican esencialmente a intercambiar información las cuales son extremadamente habituales hoy en día. Por esta razón, TCP facilita la creación e implementación de aplicaciones las cuales pueden dedicarse a definir servicios y mensajes en protocolos específicos mientras que TCP se encargará de enviar la información entre ambos extremos de manera segura y ordenada.

Socket TCP

En redes de datos TCP/IP todos los dispositivos poseen una dirección IP para identificarse. Sin embargo, una dirección IP por sí sola no es suficiente para ejecutar aplicaciones de red por lo que se hace necesario utilizar puertos de comunicación de manera que, la dirección IP será utilizada para identificar al dispositivo mientras que el puerto se utilizará para identificar la aplicación o servicio que se ejecutará en la computadora.

Un puerto se identifica con un valor de 16 bits por lo que pueden existir hasta 655535 puertos en una computadora, los cuales se dividen en los siguientes tipos:

- Puertos conocidos: 0-1023 en los cuales se alojan servicios de servidor indicados por medio de la IANA como por ejemplo el puerto 80 para servicios Web y el 25 para SMTP.
- Puertos registrados: 1024-49151, se consideran semi reservados por IANA ya que pueden ser utilizados en cualquier momento por esta. Estos puertos no deben ser utilizados por programas Web.
- Puertos de uso libre: 49121-65535, los cuales pueden ser utilizados para aplicaciones del cliente por medio de un navegador Web. Estos puertos también son conocidos como efímeros.

Los sockets son un punto final de un enlace de comunicación bidireccional entre dos programas que se ejecutan en la red. Un socket está vinculado a un número de puerto para que la capa TCP pueda identificar la aplicación a la que se enviarán los datos. [6]

Por lo general, un servidor se ejecuta en una computadora específica y tiene un socket que está vinculado a un número de puerto específico. El servidor solo espera, esto significa que se mantiene "escuchando" una solicitud de conexión por parte de un cliente el cual conoce el nombre de host de la computadora en donde se está ejecutando el servidor y el puerto en el cual "escucha" las conexiones específicas del servicio que ofrece.

5.3. MICROPYTHON

MicroPython es una reimplementación del lenguaje de programación Python enfocada a microcontroladores y sistemas embebidos.

El lenguaje de programación Python es fácil de aprender, ampliamente utilizado y expresivo. Es sencillo escribir lo que se desea con un código simple y conciso. MicroPython es una implementación completa de Python 3. A parte de algunas diferencias que tienen que ver con el hardware en donde se ejecuta, lo que se sabe de Python se puede también aplicar el MicroPython. La diferencia más significativa entre Python y MicroPython es que este último está diseñado para trabajar bajo condiciones extraordinariamente limitadas (como por ejemplo 16Kb de memoria RAM).

MicroPython se ejecuta "bare-metal" lo cual significa que corre directamente en el hardware por lo que no hay ningún sistema operativo subyacente como Windows, Linux o macOS. Todas las operaciones y servicios que son generalmente proporcionados por un sistema operativo son directamente manejados por MicroPython por lo que este tiene control total y directo sobre el hardware, de manera que para todos los efectos, MicroPython es el sistema operativo [6].

Todas las versiones de MicroPython poseen módulos para interactuar con el hardware como pines GPIO (Entradas/Salidas de propósito general), periféricos de comunicación y componentes que se conectan en sus pines. A parte de lo anterior, debido a que se ejecuta en dispositivos de diversas capacidades, la disponibilidad de librerías y prestaciones puede ser diferente. Adicionalmente, se proporcionan librerías para dispositivos con capacidades especiales como Wifi y Bluetooth.

5.4. MICROCONTROLADOR ESP32

Es un MCU (Unidad de Microcontrolador) que tiene integrados Wifi y Bluetooth lo cual le brinda conectividad a una amplia gama de aplicaciones.

Es un procesador de doble núcleo que puede funcionar a frecuencias de 160Mhz o 240Mhz, dependiendo de su configuración puede poseer 512Kb de memoria RAM, aunque por lo general se integra en placas de desarrollo con una memoria Flash Externa de hasta 4MB. Además, posee 36 pines GPIO para interactuar con dispositivos periféricos, aunque solo están disponibles 34.

Sus capacidades de red son dos: Wifi con soporte para IEEE 802.11 b/g/n/e/i, y Bluetooth versión 4.2 en modos clásico y bajo consumo de energía. A nivel de seguridad es compatible con WFA, WPA/WPA2 y autenticación Wifi WAPI, secure boot, encripta miento flash y aceleración de hardware criptográfico para algoritmos AES, SHA-2, RSA, y ECC [7].

Si bien este tipo de MCU puede adquirirse en un encapsulado tradicional SMD por lo general, se presenta en diversas placas de desarrollo que permiten realizar un prototipado rápido. Una de las placas ESP32 más populares es la ESP32 Devkit V1 que incorpora un ESP32 WROOM-32 cuya especificación de pines se muestran en la figura 4.

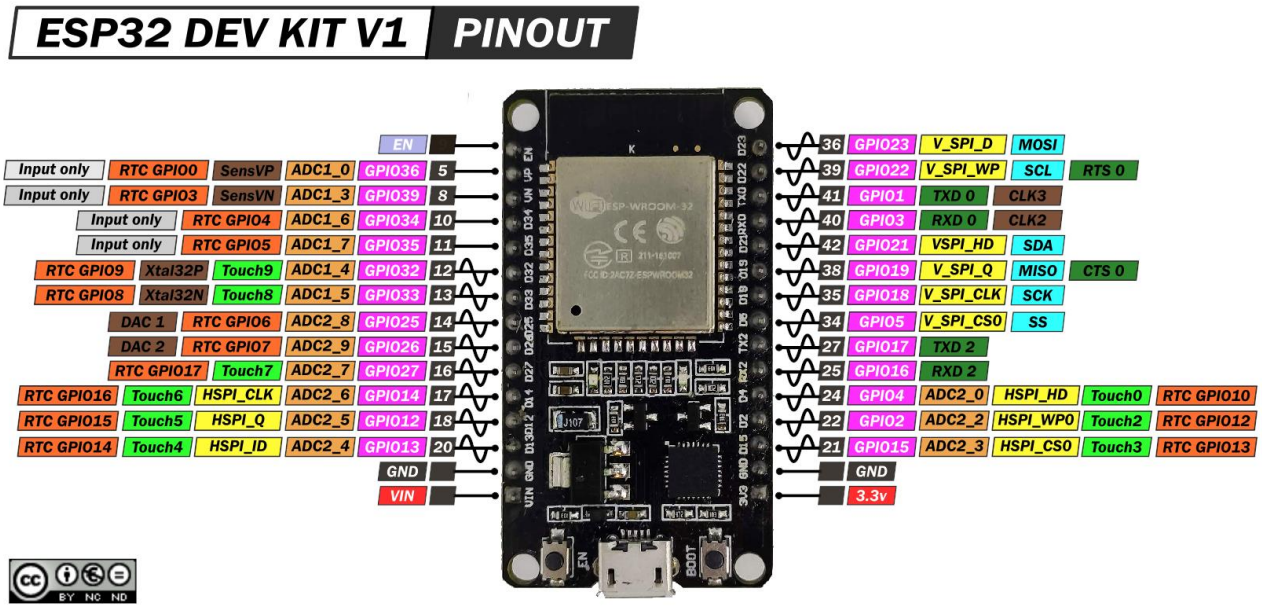


Fig. 4: Pines y funciones I/O de la placa de desarrollo ESP32 DEV Kit V1 de 30 pines.

5.5. Plataforma de Teleingeniería

Una plataforma de Teleingeniería para un laboratorio remoto virtual es un ambiente de experimentación en el cual los usuarios pueden operar una serie de objetos gráficos, cada uno de ellos representa un objeto que efectúa experimentos virtuales vía internet por medio de operaciones locales realizadas con un ratón o teclado. Con la ayuda de los laboratorios remotos virtuales, los estudiantes pueden efectuar experimentos a través de internet 24 horas al día, siete días de la semana, sin limitaciones de espacio y tiempo. Adicionalmente, proporcionan a los estudiantes la posibilidad de efectuar prácticas y ejercicios de forma selectiva de manera que pueden seguir un proceso lógico de aprendizaje.

Los proyectos de investigación relacionados con este tipo de laboratorios coinciden en que la arquitectura está basada en dos partes: servidor y cliente, esto significa que se aplica una arquitectura típica cliente-servidor. Sin embargo, difieren en cuanto a las prestaciones que posee la aplicación de lado del cliente.

Una plataforma de Teleingeniería con el enfoque de la Escuela Especializada en Ingeniería ITCA-FEPADE es aquella en donde la máquina (entrenador) es gestionada eléctrica y/o mecánicamente por medio de un circuito electrónico inteligente con conectividad a una red de datos. Este circuito electrónico puede constituirlo un controlador lógico programable al cual se le han incorporado periféricos de comunicación o, puede ser un circuito controlador diseñado a medida de las necesidades.

El controlador es quien gestiona las acciones del entrenador y está escuchando continuamente las peticiones del cliente que es una aplicación Web alojada en un servidor. Los usuarios se conectan a esta aplicación Web que realiza la gestión, registro y control de acceso a las prácticas de laboratorio y que a su vez envía comandos de ajuste y de solicitud de datos al controlador electrónico del entrenador. En esta plataforma es esencial la presencia de cámaras que transmiten video en tiempo real el cual es observado por el usuario en todo momento mejorando su experiencia práctica.

6. METODOLOGÍA DE INVESTIGACIÓN

En procedimiento de investigación realizado se esquematiza a continuación:

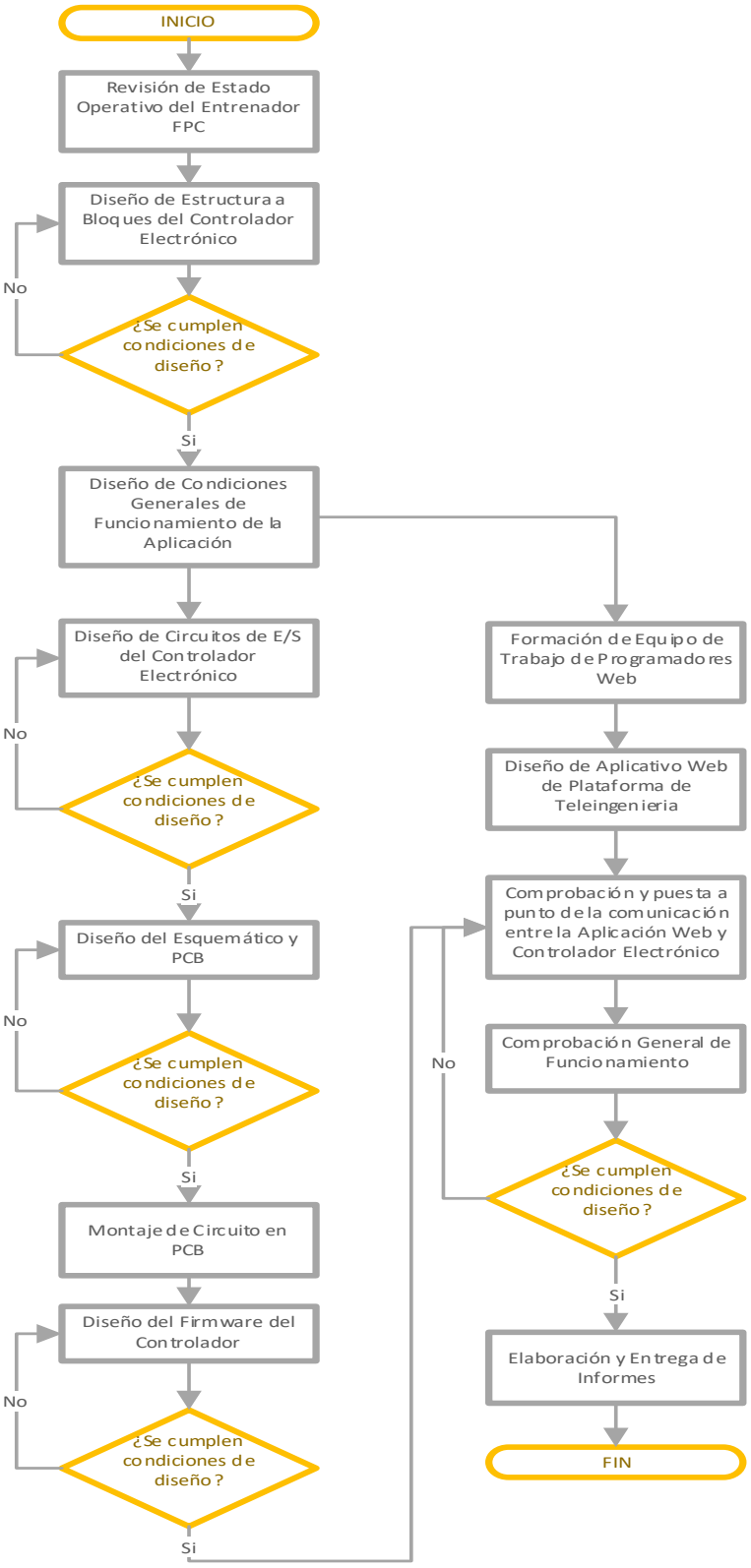


Fig. 5: Procedimiento lógico de investigación realizado.

El trabajo de investigación fue liderado por los docentes investigadores de la Escuela de Ingeniería Eléctrica y Electrónica y contó además con el apoyo de docentes investigadores y estudiantes de la Escuela de Ingeniería en Computación. Las etapas de ejecución del proyecto son las siguientes:

6.1. Revisión del Estado Operativo del Entrenador FPC

Inicialmente, se verificó el estado de cada una de las partes electrónicas y mecánicas que forman parte del entrenador con el objetivo de determinar si alguna presentaba fallas que impidieran el correcto funcionamiento de la máquina. No se encontraron problemas.

El esquema del entrenador de control de procesos de temperatura (FPC) que se modificó para controlar de forma remota es el siguiente:

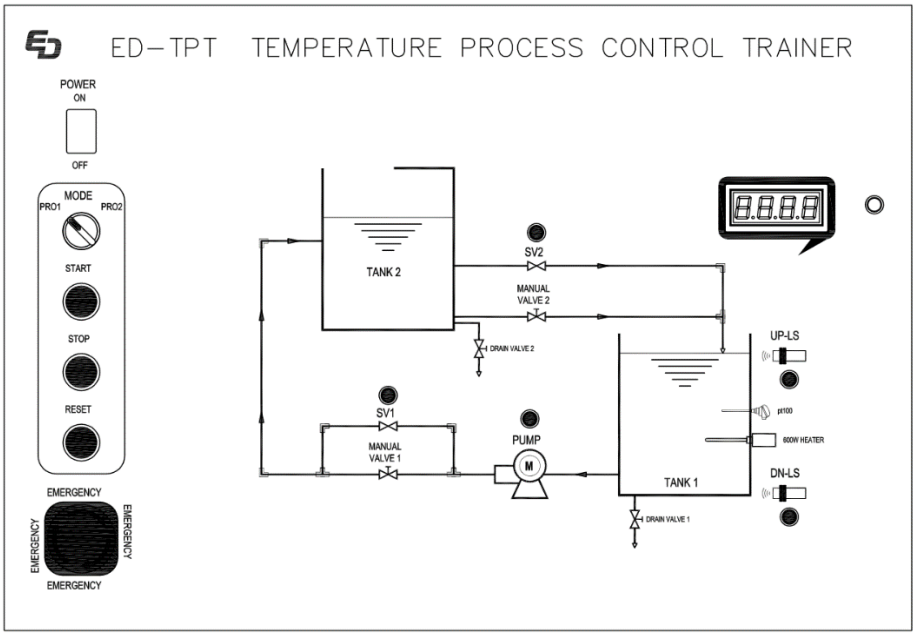


Fig. 6: Diagrama del proceso de control de temperatura junto con los interruptores de entrada, sensores y actuadores.

Adicionalmente, se analizaron las características de operación de los dispositivos del entrenador FPC que originalmente son controlados por un PLC WAGO 750-342 y a partir del análisis se creó la tabla de entradas y salidas.

Tabla 1: Definición de entradas y salidas del entrenador FPC.

CONTROLADOR FPC2 WAGO 750-342					
ENTRADAS			SALIDAS		
NOMBRE	TIPO	DESCRIPCION	NOMBRE	TIPO	DESCRIPCION
UP-LS	DIGITAL	Activo en H	SOL VALVE 1	DIGITAL	Activo en H
DN-LS	DIGITAL	Activo en H	SOL VALVE 2	DIGITAL	Activo en H
MODE 1	DIGITAL	Activo en H	PUMP MOTOR	DIGITAL	Activo en H
MODE 2	DIGITAL	Activo en H	HEATER	PWM	1V - 5V
START	DIGITAL	Activo en H			
STOP	DIGITAL	Activo en H			
RESET	DIGITAL	Activo en H			
EMERGENCY	DIGITAL	Activo en L			
RTD	ANALOGICO	0V - 5V DC			
ETHERNET	DIGITAL	192.168.1.102			
La lógica del controlador: 24VDC					

Con la información de la tabla 1, se pudieron definir los requerimientos de elección del controlador de entradas y salidas, también permitió tomar la decisión de utilizar una tarjeta de captura para la resistencia RTD y algunas premisas de diseño que se muestran en la tabla 2.

Tabla 2: Requerimientos de elección del controlador ES y premisas de diseño.

E/S con Microcontrolador					
ENTRADAS			SALIDAS		
NOMBRE	TIPO	DESCRIPCION	NOMBRE	TIPO	DESCRIPCION
UP-LS	DIGITAL	Activo en H	SOL VALVE 1	DIGITAL	Activo en H
DN-LS	DIGITAL	Activo en H	SOL VALVE 2	DIGITAL	Activo en H
MODE 1	DIGITAL	Activo en H	PUMP MOTOR	DIGITAL	Activo en H
MODE 2	DIGITAL	Activo en H	HEATER	PWM	0V - 5V
START	DIGITAL	Activo en H			
STOP	DIGITAL	Activo en H			
RESET	DIGITAL	Activo en H			
EMERGENCY	DIGITAL	Activo en L			
RTD	DIGITAL con BUS SPI	0V - 3.3V DC			
Lógica del procesador: TTL a 5V					
*La lógica del controlador es de 5V, excepto el controlador para la tarjeta de captura RTD PT100 que es de 3.3VDC					
**Las entradas digitales pasarán por un circuito de acople de voltajes que cambiará la lógica de activación a L					
***Cada una de las entradas digitales deberá poseer un circuito de debounce apropiado					

6.2. Diseño de la Estructura a Bloques del Controlador Electrónico

Todos los dispositivos electrónicos y mecánicos son gobernados por un circuito electrónico embebido que integra una parte lógica y de procesamiento, así como interfaces de acople para sensores de entrada e interruptores y, acoples para actuadores de salida. Lo anterior implica que el controlador electrónico está dividido en una serie de subcircuitos con características eléctricas de funcionamiento diferentes pero que deben funcionar de forma sincronizada. En el anexo 1, se muestra el diagrama de bloques completo del circuito controlador.

Los subcircuitos son los siguientes:

a. Sistema de control principal (SYSCON)

Este circuito tiene como principal función el correcto establecimiento de la comunicación con la aplicación web de control, motivo por el cual debe tener la capacidad de conectarse a una red de datos Ethernet. Además, debe gestionar el funcionamiento del circuito convertidor del sensor de temperatura PT100 y del microcontrolador gestor de entradas y salidas, el diagrama resultante se muestra en la figura 7.

Como microcontrolador principal se eligió el ESP32 WROOM32 de Expressif que viene embebido en una placa de desarrollo ESP32 Devkit V1 de 30 terminales. Este dispositivo cumple con todos los requerimientos de diseño que principalmente son:

- Conectividad Ethernet Wifi 802.11b/g/n/e/i.
- 4MB de memoria EEPROM.
- Puertos GPIO de propósito general.
- Soporte para bus SPI.
- Soporte para Micropython.
- Frecuencia de reloj de hasta 240MHz.
- Programación vía puerto USB.

- Fuente de alimentación integrada de 3.3V.
- Hardware de código abierto.

Al ser un procesador con niveles de tensión de hasta 3.3V fue necesario dotar al circuito de acople de niveles de tensión basado en el circuito YF08E.

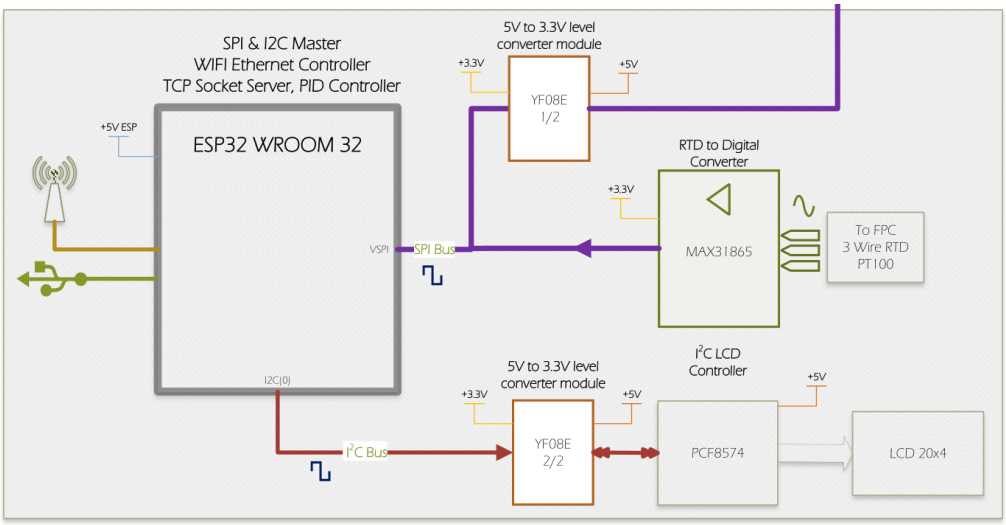


Fig. 7: Diagrama a bloques del Syscon.

b. Sistema de control de entradas y salidas

Este es un sistema de control robusto que opera como esclavo SPI del SYSCON principal. El requerimiento de robustez implica un mayor manejo de corrientes de entrada y salida, así como la generación y control de una señal PWM. Se eligió el microcontrolador PIC18F4550 de Microchip que está ampliamente documentado y puede programarse utilizando lenguaje XC8 que es ANSI C.

Los requerimientos de diseño de este circuito son:

- Niveles de voltaje de operación TTL (hasta 5V).
- Compatible con bus SPI como dispositivo esclavo.
- Convertidor analógico a digital de 10 bits.
- Módulo generador de señal PWM con frecuencia programable.
- Temporizadores programables de 16 bits.
- Interrupciones internas y externas programables.
- Compatible con lenguaje XC8 de Microchip.
- Programable vía Pickit 3 gestionado por MPLAB X vía ICSP.
- Reloj interno de hasta 8MHz.

c. Acoples de señales de entrada y salida

Debido a que el entrenador posee dispositivos industriales que trabajan con niveles de tensión de 24V, el circuito controlador debe cumplir con un requisito indispensable que es el aislamiento galvánico entre los sensores e interruptores de entrada y el sistema de control de entradas y salidas, esto se consiguió con optoacopladores PC817. Adicionalmente, el circuito de salida debe amplificar

las señales para activar los actuadores correspondientes, para lograrlo se seleccionó el circuito integrado ULN2003A que es compatible con señales TTL de entrada y sus salidas pueden manejar niveles de voltaje de 24V.

El diagrama de bloques resultante del sistema de control de entradas y salidas, así como los acoples se muestra en la figura 8.

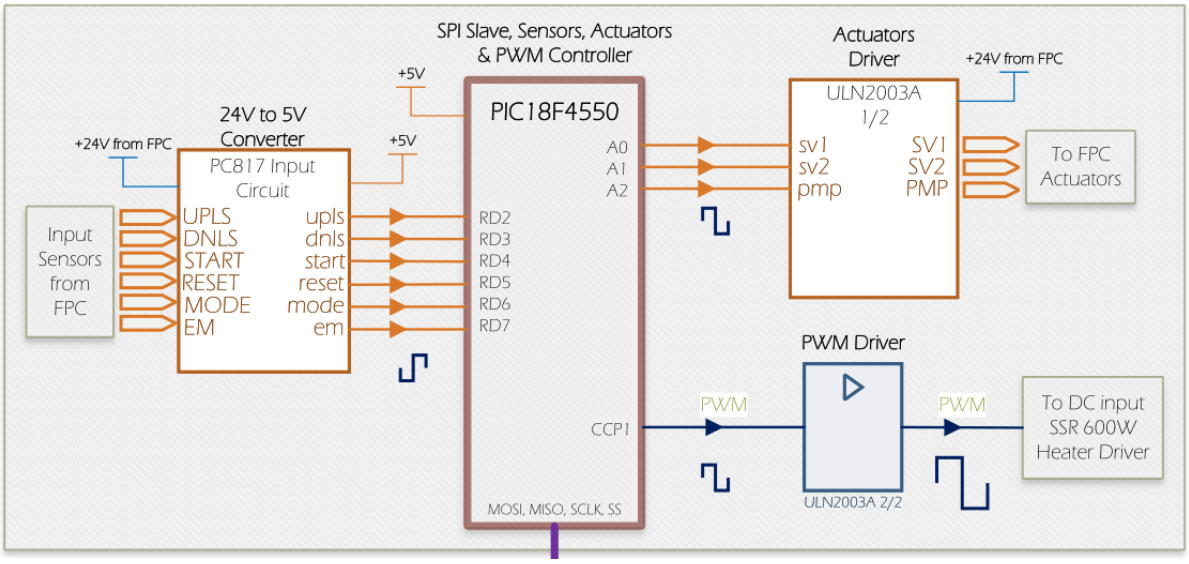


Fig. 8: Diagrama a bloques del controlador E/S y circuitos de acople.

d. Fuente de alimentación

El circuito de alimentación está conformado por una fuente regulada de uso industrial de 12V y 5A de corriente, este voltaje es a su vez ajustado por reguladores de voltaje lineal 7805 para +5V y AMS1117 para +3.3V, este último está embebido en la placa de desarrollo ESP32 Devkit V1. La figura 9, muestra el diagrama a bloques de la fuente de alimentación.

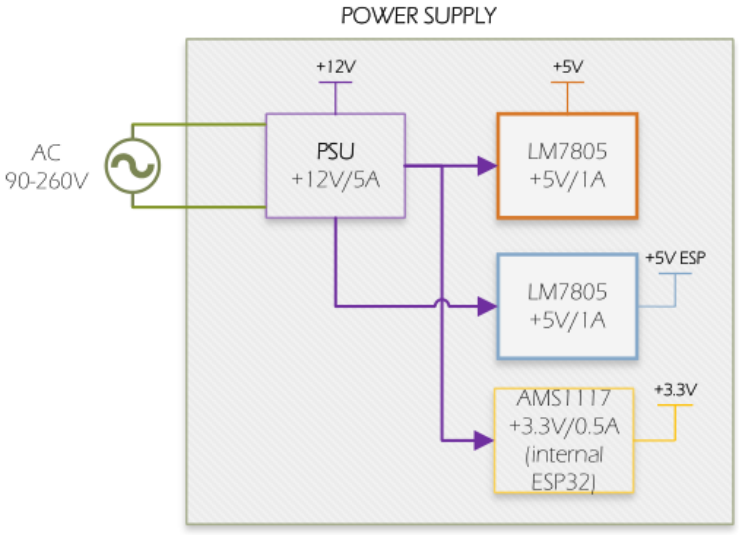


Fig. 9: Diagrama a bloques de la fuente de alimentación.

6.3. Diseño de las Condiciones Generales de Funcionamiento de la Aplicación

Una vez finalizado el diseño general a bloques del controlador electrónico y habiendo seleccionado el dispositivo controlador de comunicación ESP32 WROOM 32 se procedió a crear las condiciones de funcionamiento de la aplicación de software que forma parte de la plataforma de Teleingeniería.

En esta etapa, el trabajo estuvo orientado a determinar dos cosas:

1. **Módulos de gestión, control de acceso y pantalla de control del entrenador.**

   Se determinó que la aplicación debía contar con los siguientes módulos:

   - Control de acceso seguro para que docentes y alumnos puedan hacer uso de la plataforma validándose por medio de correo electrónico y contraseña de acceso.
   - Administración de usuarios mediante la cual el docente puede registrar y realizar modificaciones en los datos de los estudiantes registrados.
   - Administración de horarios de prácticas a distancia en donde los estudiantes pueden programar y reservar una práctica de laboratorio semanal la cual debe a su vez ser aprobada por el docente.
   - Pantalla de control del entrenador en donde se tendrá acceso a los controles del entrenador, cajas de texto para la introducción y modificación de parámetros de control, así como gráficas e indicadores que se utilizarán para el monitoreo del estado del proceso que se está controlando.

2. **Protocolo de comunicación entre la aplicación y el controlador electrónico del entrenador FPC.**

   Una de las etapas más críticas del proyecto consistió en el diseño y mejoramiento del protocolo de comunicación el cual describe la forma en que la aplicación sabrá que está haciendo el entrenador en un momento determinado, así como el estado del proceso, también debe servir para que la aplicación gire órdenes al entrenador a través del controlador electrónico.

   La definición del protocolo de comunicación está directamente relacionada con la estrategia de comunicación a utilizar y la red de datos. Debido a que por la naturaleza del proceso a controlar se requiere de un nivel de respuesta y control en tiempo real se decidió utilizar el estándar de comunicación Ethernet sobre todo porque los usuarios de esta plataforma utilizarán una computadora y navegador web moderno para conectarse y realizar sus prácticas de laboratorio.

   Una vez definido el estándar de comunicación se pasó a determinar qué protocolo de comunicación compatible con Ethernet sería el más adecuado, se evaluaron varias alternativas desde el uso de MQTT [8]hasta el diseño de una Rest API [9].

   Sin embargo, debido a la necesidad de mantener una comunicación y actualización de la información en tiempo real se decidió utilizar Sockets TCP como método de transporte de los comandos y datos que fluirán entre la aplicación y el controlador electrónico que a su vez forman parte de un protocolo de comunicación hecho a medida de las necesidades. La figura 10, muestra las capas del modelo comunicación utilizado.

Fig. 10: capas del modelo de comunicación.

Para la elaboración del protocolo de comunicación se definieron las transacciones de datos en donde se definen los datos y sus tipos que estarán fluyendo entre la aplicación y el controlador electrónico (tabla 3).

Tabla 3: Transacciones de datos entre aplicación y controlador electrónico.

Tabla de transacciones de DATOS con el Servidor de Aplicaciones

ESTADO DE SENSORES DE ENTRADA

NOMBRE	TIPO DE DATO	DESCRIPCION	RANGO DE VALORES	ENCAPSULAMIENTO	ACCION-DEFAULT
UPLS	BIT	Sensor capacitivo de nivel superior de H2O	1/0	Byte de estado	NA
DNLS	BIT	Sensor capacitivo de nivel inferior de H2O	1/0		
LEVEL	BIT	Entrenador en modo de ajuste de nivel	1/0		
FLOW	BIT	Entrenador en modo normal de operación	1/0		
START	BIT	Inicio de funcionamiento	1/0		
STOP	BIT	Detiene el funcionamiento	1/0		
RESET	BIT	Modo de operación del entrenador que establece las condiciones de inicio (es un RESET) o acción de control	1/0		
EMERGENCY	BIT	Detención forzada del entrenador, se deben detener todas las acciones sin aplicar un reinicio. Requiere acción humana en sitio	1/0		

ACTUADORES / SALIDAS

NOMBRE	TIPO DE DATO	DESCRIPCION	DESCRIPCION	ENCAPSULAMIENTO	ACCION-DEFALUT
SOL VALVE 1	BIT	Válvula de paso inferior	1/0	Byte de estado	NA
SOL VALVE 2	BIT	Válvula de paso superior	1/0		
PUMP	BIT	Bomba de H2O	1/0		

PARAMETROS PARA EL CONTROLADOR

NOMBRE	TIPO DE DATO	DESCRIPCION	DESCRIPCION	ENCAPSULAMIENTO	PERMISO-DEFAULT
TEMPERATURA (Variable de Proceso)	FLOAT	Temperatura actual en grados Celsius de H2O medida por el sensor PT100	0.00 a 100.00	Cadena	R-0.00
SETPOINT	FLOAT	Es la temperatura en grados Celsius a la cual se desea calentar el H2O	0.00 a 100.00	Cadena	RW-0.00
PROPORCIONAL	FLOAT	Valor de la ganancia proporcional del controlador PID, un P = 0.00 indica que está apagado	0.00 a 1000.00	Cadena	RW-0.00
INTEGRAL	FLOAT	Valor de la ganancia integral del controlador PID, un I = 0.000 indica que está apagado	0.000 a 1000.00	Cadena	RW-0.000
DERIVATIVO	FLOAT	Valor de la ganancia derivativa del controlador PID, un D = 0.000 indica que está apagado	0.000 a 1000.00	Cadena	RW-0.000
CICLO DE TRABAJO DEL CALENTADOR	FLOAT	Corresponde al porcentaje de energía que se está aplicando al calentador de agua del entrenador, es un valor float de entre 0.0 y 100.0	0.0 a 100.0	Cadena	R-0
REFRESCO	FLOAT	Tiempo estimado de actualización de parámetros (en segundos) entre controlador y servidor de aplicaciones	0.1 a 30	Cadena	RW-1.0

Como puede apreciarse en la tabla 3, las transacciones de datos se dividen en tres áreas, las primeras dos son de uso exclusivo del circuito controlador electrónico y permiten una comunicación eficiente entre el SYSCON y el controlador de entradas y salidas. La aplicación, a través de comandos puede llegar a modificar el estado de los sensores y actuadores que sean requeridos. Ambos se encapsulan en bytes de estado cuya estructura se muestra en la figura 11.

- Estado de sensores de entrada

  Cada sensor tiene un valor de un bit y en su conjunto describen el estado de los interruptores de entrada del entrenador y se encapsulan en un byte de estado.

- Actuadores / Salidas

  Al igual que los sensores de entrada tienen un valor de un bit y en su conjunto describen que actuadores están activados / desactivados.

Estructura de los bytes de estado del controlador

LEYENDA			
R	W	-	0/1
Lectura	Escritura		Valor por defecto

*Si un bit es R, solo podrá ser leído por el servidor de aplicaciones

**Si un bit es RW, podrá ser leído y modificado por el servidor de aplicaciones
***El valor por defecto se establece al iniciar el controlador o cuando se produce un RESET

SENSORES DE ENTRADA

R-0	R-0	R-0	RW-0	RW-1	RW-0	R-0	R-0
EMERGENCY	RESET	STOP	START	FLOW	LEVEL	DNLS	UPLS
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Ejemplo: si byte enviado es igual a:

0	0	0	1	1	0	0	0

Se leería: el entrenador está funcionando (START = 1) en modo flujo (FLOW = 1)

ACTUADORES

					RW-0	RW-0	RW-0
NA	NA	NA	NA	NA	PUMP	SOLV2	SOLV1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Ejemplo: si byte enviado es igual a:

X	X	X	X	X	1	1	1

Se leería: el entrenador está funcionando en modo 2, hay flujo de agua porque están abiertas las válvulas solenoides inferior y superior, así como la bomba

Fig. 11: Estructura de los bytes de estado del controlador electrónico del entrenador FPC.

- Parámetros del controlador

  Definen las variables y tipos de datos que el controlador identifica como sus parámetros de operación y que alteran el proceso que se está controlando, estos parámetros son gestionados desde la aplicación.

  Set Point: es la temperatura a la cual se desea mantener el fluido del entrenador

  Temperatura: es la temperatura actual del fluido, en control de procesos se le conoce como Variable de Proceso (Process Variable)

  Proporcional: es la ganancia proporcional del controlador PID

Integral: es la ganancia integral del controlador PID.

Derivativo: es la ganancia derivativa del controlador PID.

Ciclo de trabajo del calentador: es un valor porcentual de la energía que el controlador electrónico está entregando al calefactor del fluido.

- **Comandos del servidor de aplicaciones**

La capa superior del modelo de comunicación la conforma el protocolo de comunicación en su versión 1.2. Está basado en comandos que funcionan en un modelo cliente-servidor web en donde el controlador electrónico del entrenador es el servidor y la aplicación de la plataforma (alojada en el servidor de aplicaciones) es el cliente.

El controlador implementa un *servidor TCP Socket asíncrono* y está codificado para leer cadenas de comandos y datos que deben finalizar con un salto de línea \n.

El servidor de aplicaciones deberá realizar una función de cliente TCP asíncronapor medio de sockets.

A continuación, se muestra la estructura básica de los dos tipos de comandos que forman parte del protocolo.

- **Comandos para leer el estado del controlador**

Sintaxis: Comando\n

Descripción: el servidor de aplicaciones solicita que el controlador le envíe los datos de estado solicitados. El comando debe finalizar con salto de línea \n.

Identificadores de los datos: estos se especifican detalladamente en la Tabla de transacciones de datos.

Comando:

23: **SETPOINT**: el controlador enviará el valor actual del setpoint que tiene establecido, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario.

Ej. 23\n <comando>
35.5 <valor respuesta>

- **Comandos para actualizar el estado del controlador**

Sintaxis: Comando\n

Descripción: el servidor de aplicaciones está indicando que enviará un dato de actualización al controlador. El proceso de la transacción es el siguiente:

a. El servidor de aplicaciones envía el comando <comando>\n, en donde <comando> identifica lo que se va a actualizar, el carácter de fin de línea \n indica la finalización del comando.

b. El controlador recibirá el comando y evaluará si el id corresponde a un campo actualizable, en caso afirmativo enviará una respuesta de acuerdo con la naturaleza de la acción a realizar.

c. Los comandos relacionados con la actualización de variables para el controlador PID (Setpoint, Proporcional, Integral, Derivativo) requieren de dos transacciones TCP, la primera es para indicar que variable se va a actualizar y la segunda para especificar el nuevo valor de la variable.

Identificadores con permisos de escritura: son aquellos en los cuales el servidor de aplicaciones tiene permiso de escritura de acuerdo con la Tabla de transacciones de datos y estructura de los bytes de estado.

Comando:

51: **PROPORCIONAL**, actualización del valor de la variable del control proporcional del controlador PID. Al recibir este comando el controlador esperará un segundo dato desde el servidor de aplicaciones que corresponde al valor proporcional.

Ej. 51\n <comando>
2.0\n <valor>

En el **Anexo 2**, se muestra el documento de referencia: "Protocolo de Comunicación entre Controlador del Entrenador FPC y Servidor de Aplicaciones" en su versión 1.2 en dicho documento puede encontrarse la lista completa de comandos, así como su estructura y sintaxis de uso.

6.4. Diseño de Circuitos de Entrada y Salida E/S del Controlador Electrónico

Los circuitos de entradas y salidas deben cumplir con dos requerimientos esenciales, primeramente deben proporcionar un aislamiento galvánico entre los sensores, actuadores y microcontrolador controlador de entradas y salidas ya que funcionan con una lógica de voltaje diferente, el segundo requerimiento es que no deben deformarse las señales de entrada y salida ya que si bien son digitales si los circuitos no se diseñan apropiadamente se corre el riesgo de tener activaciones y desactivaciones erráticas. Adicionalmente, todos los dispositivos que forman parte de este circuito deben ser de fácil adquisición en el mercado local.

a. Circuito de entrada

Está basado en el optoacoplador PC817 que tiene la función de aplicar un aislamiento galvánico para separar la lógica de 24V de los dispositivos de entrada de la lógica TTL de 5V del controlador ES. El circuito diseñado se muestra en la figura 12.

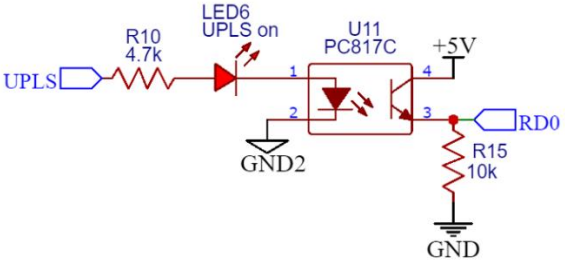


Fig. 12: Circuito de entrada basado en optoacoplador.

El circuito de entrada recibe la señal de estado de un sensor basado en interruptor SPST normalmente abierto. Si el interruptor está cerrado el nivel del voltaje de la entrada es +24V, la corriente de entrada es limitada por la resistencia de 4.7kΩ que polariza en directa al diodo led de 3mm el cual se encenderá y hará circular una corriente de 5mA por el diodo led interno del optoacoplador. El diodo led interno generará una energía luminosa que saturará al transistor NPN interno del optoacoplador que establecerá en el pin 3 un voltaje de 5V se enviará a una de las entradas digitales del microcontrolador PIC18F4550. La resistencia del pull down de 10kΩ que se encuentra conectada a el pin 3 del optoacoplador asegura que cuando el transistor está abierto el voltaje en dicho pin sea 0V.

En el circuito puede también apreciarse que el pin 2 tiene una conexión a tierra distinta a que se tiene en la resistencia de pull down [10] debido a que son fuentes de alimentación diferentes, esto asegura el aislamiento galvánico entre los circuitos con lógica de 24V y de 5V.

El circuito resultante no sólo consigue aislar ambas lógicas de voltaje sin deformar la señal de entrada, sino que también incorpora un indicador de activación (el diodo led) que es una herramienta invaluable cuando se realizan acciones de mantenimiento y reparación.

b. Circuito de salida

Los circuitos de salida están formados por dos etapas, la primera es de baja potencia y tiene una función similar a la de los circuitos de entrada que es proporcionar un aislamiento galvánico de los circuitos sin deformar la señal.

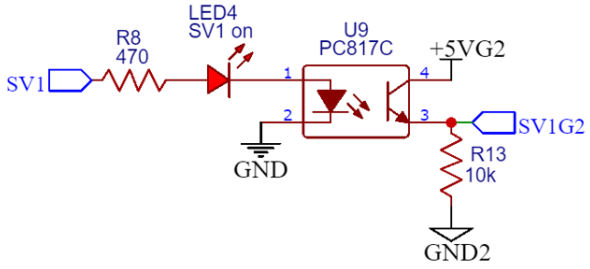


Fig. 13: Circuito de salida basado en optoacoplador.

En la fig. 13, se muestra el circuito de salida en donde puede apreciarse la similitud con el circuito de entrada. En la salida (pin 3), el nivel de voltaje sigue siendo igual que en la entrada (5V) pero ya se encuentra aislado y listo para aplicarse al circuito amplificador que está basado en el circuito integrado ULN2003A.

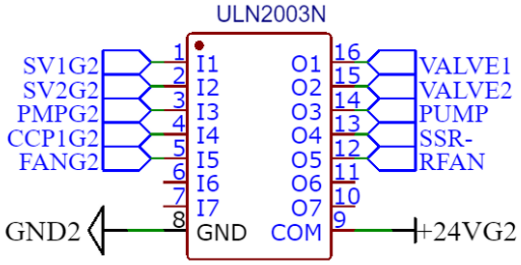


Fig. 14: Circuito amplificador de salida basado en ULN2003A.

Los actuadores de salida son cuatro, tres de ellos son relés de 24V que se utilizan para activar las válvulas eléctricas y el cuarto es un relé de estado sólido (SSR) [11] con entrada DC PWM que se utiliza para aplicar energía a la resistencia calefactora de 600W, cada relé tiene un consumo promedio de 30mA mientras que el relé de estado sólido es un KSI240D25 cuya corriente nominal de entrada es de hasta 25mA. Como la salida del microcontrolador PIC18F4550 no es capaz de generar las corrientes necesarias se utiliza el circuito integrado ULN2003A (Fig. 13) como buffer de corriente ya que a sus entradas pueden activarse con lógica de 5V mientras que sus salidas puedan manejar perfectamente 24V que es compatible con los relés y el relé de estado sólido.

El relé de estado sólido puede manejar corrientes de carga AC de hasta 25A con voltajes de entre 48-280VAC. En el circuito amplificador de salida puede observarse que el pin 13 corresponde a SSR-, esto sígnica que el proceso de regulación de la energía se realiza polarizando a tierra el fotodiodo interno en una conexión conocida como sink-mode [11] (Fig. 15).

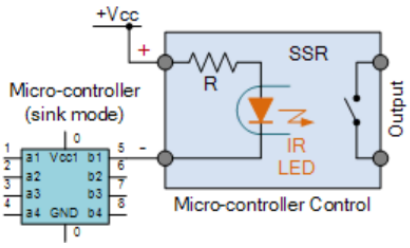


Fig. 15: Control PWM del SSR en conexión Sink con el microcontrolador.

c. Circuito de control de entradas y salidas

El microcontrolador de 8 bits PIC18F4550 es el encargado de controlar entradas y salidas además de comunicarse con el controlador principal ESP32 por medio de un bus SPI. El controlador de entradas y salidas notificará al ESP32 cada vez que se produzca una variación de las entradas y esperará a que este le ordene realizar algún cambio de estado en los actuadores de salida para SV1, SV2 y la bomba de agua. Además, cuando el ESP32 se lo ordena activará la señal PWM de control del SSR y realizará ajustes en el ciclo de trabajo de señal consiguiendo así regular la energía que se entrega al calefactor.

Debido a que el microcontrolador PIC18F4550 no realiza ninguna tarea que requiera precisión en tiempos de ejecución, no fue necesario dotarlo de un circuito oscilador de alta estabilidad basado en cristal de cuarzo. En su lugar se utilizó el oscilador interno de 8Mhz obteniendo buenos resultados en las pruebas de funcionamiento realizadas.

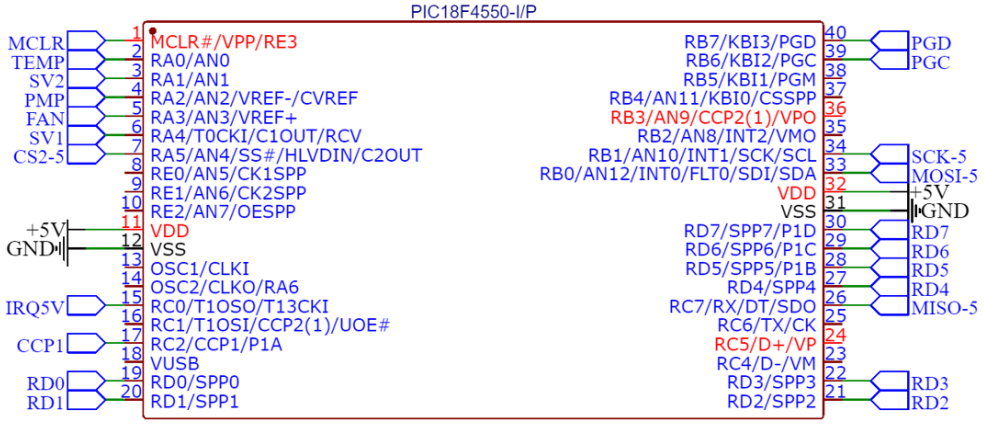


Fig. 16: Controlador de entradas y salidas basado en el PIC18F4550.

En la figura 16, se puede apreciar la definición de pines del microcontrolador y su función operativa en el circuito controlador. El puerto A es de salida digital para los actuadores y el puerto D es de entradas digitales para los interruptores. La salida CCP1 es utilizada para generar y controlar el ciclo de trabajo de la señal PWM para el SSR. Para la conexión del bus SPI se utilizó el módulo nativo del microcontrolador (MISO, MOSI, SCK y CS). Adicionalmente, para efectos de desarrollo se dotó al controlador de las conexiones necesarias para su programación ICSP (MCLR, PGD, PGC) de manera que no fuese necesario retirar el dispositivo del circuito para realizar una reprogramación.

d. Circuito SYSCON

El sistema de control (SYSCON) del controlador está basado en el microcontrolador ESP32 WROOM 32 embebido en la placa de desarrollo ESP32 Dev Kit V1 de 30 terminarles con fuente de alimentación de 3.3V 500mA. Como la placa de desarrollo cuenta con todos los circuitos necesarios para que el microcontrolador funcione, la fase de diseño se simplificó bastante y permitió dirigir esfuerzos en asegurar el acople de niveles de voltaje ya que funciona a 3.3V. El acople se consiguió con el circuito YF08E que dispone de 8 canales de ajuste de niveles.

Si bien para este proyecto no fue necesario su incorporación, se dotó al circuito SYSCON de un puerto de conexión para un encoder rotativo para que en un futuro se puedan introducir órdenes al controlador de forma local, así como de un puerto para la conexión de un display alfanumérico LCD con bus I2C para la visualización de variables y el proceso en general. Previo al diseño de las conexiones entre el ESP32 y el resto de los dispositivos se hizo un análisis en las entradas y salidas disponibles en la placa de desarrollo para garantizar que posteriormente no se tuviesen problemas eléctricos ni de programación. La definición de pines resultante se muestra en la tabla 4.

Tabla 4: Definición de pines de la placa de desarrollo ESP32 Dev Kit V1. La columna checked indica que la conexión fue comprobada para garantizar compatibilidad.

GPIO	Input	Output	Notes	Circuit Application	State	Device	Checked
0	pulled up	OK	outputs PWM signal at boot	x			
1	TX pin	OK	debug output at boot	x			
2	OK	OK	connected to on-board LED				
3	OK	RX pin	HIGH at boot	x			
4	OK	OK					
5	OK	OK	outputs PWM signal at boot				
6	x	x	connected to the integrated SPI flash	x			
7	x	x	connected to the integrated SPI flash	x			
8	x	x	connected to the integrated SPI flash	x			
9	x	x	connected to the integrated SPI flash	x			
10	x	x	connected to the integrated SPI flash	x			
11	x	x	connected to the integrated SPI flash	x			
12	OK	OK	boot fail if pulled high				
13	OK	OK		IRQ	input	From PIC18F4550	
14	OK	OK	outputs PWM signal at boot	x			
15	OK	OK	outputs PWM signal at boot	x			
16	OK	OK					
17	OK	OK					
18	OK	OK		SCK	output	MAX31865/PIC18F4550	
19	OK	OK		MISO	input	MAX31865/PIC18F4550	
21	OK	OK		CS2	output	PIC18F4550	
22	OK	OK		CS1	output	MAX31865	
23	OK	OK		MOSI	output	MAX31865/PIC18F4550	
25	OK	OK		SCL	bidirectional	Display LCD I2C	
26	OK	OK		SDA	bidirectional	Display LCD I2C	
27	OK	OK		DT	input	Rotary Enconder	
32	OK	OK		CLK	input	Rotary Enconder	
33	OK	OK		SW	input	Rotary Enconder	
34	OK		input only				
35	OK		input only				
36	OK		input only				
39	OK		input only				

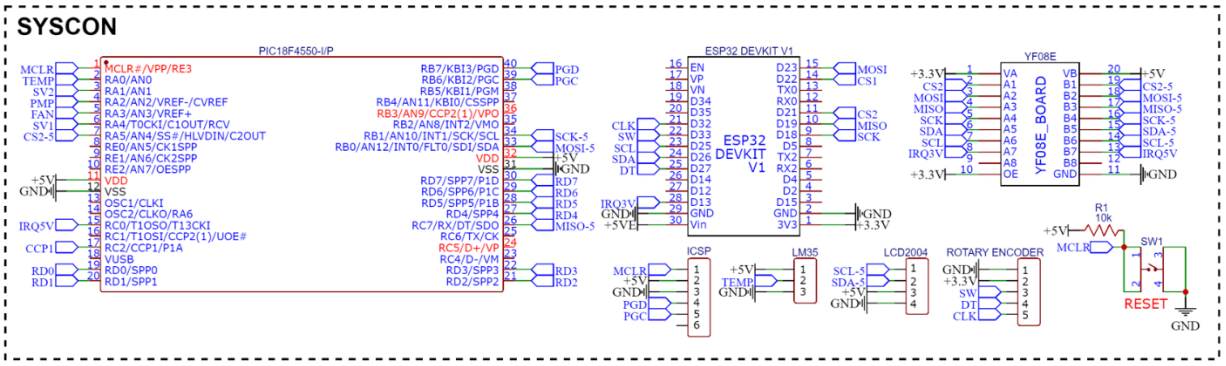


Fig. 17: Circuito SYSCON completo, se incluye el PIC18F4550 ya que en forma parte de los dispositivos lógicos programables del circuito.

En la figura 17, se muestra el circuito completo del SYSCON, se puede apreciar que también se dotó al circuito de una conexión para un sensor de temperatura local basado en LM35 para el monitoreo de la temperatura del circuito controlador en caso de que este sea instalado en un espacio cerrado con temperaturas elevadas, este valor de temperatura se utilizará para activar en ventilador de enfriamiento que dispone de un canal de activación en el circuito de salidas.

e. Convertidor de RTD a SPI

Está basado en el circuito integrado MAX31865 que viene embebido en una placa de desarrollo. El dispositivo traduce las variaciones en la resistencia de un sensor industrial PT100 en valores digitales que se envían a un microcontrolador por medio de un bus SPI. Como este dispositivo funciona con 3.3V se puede conectar directamente al controlador ESP32 y su voltaje de alimentación es proporcionado por la fuente de alimentación embebida en el ESP32 Dev Kit V1 (Fig. 18).

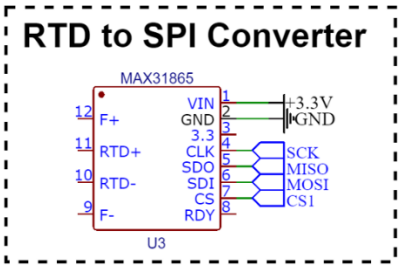


Fig. 18: Circuito SYSCON completo, se incluye el PIC18F4550 ya que forma parte de los dispositivos lógicos programables del circuito.

f. Circuitos de alimentación

Los circuitos de alimentación están basados en una fuente de +12V y 5A marca IMO modelo BPS-1-60-12DC de 60W. A partir de los 12V se utilizan dos reguladores lineales de voltaje 78M05 con encapsulado TO-252 tipo SMD que pueden proporcionar hasta 500mA de corriente.

Se cuenta también con una fuente de alimentación de +24V marca MW modelo DR-120-24 de hasta 120W y 5A. Esta fuente de alimentación es utilizada por el circuito buffer de salida ULN2003A para activar los relés de las válvulas y bomba de agua. Además, un regulador lineal 78M05 es utilizado para proveer 5V para las entradas del buffer de las salidas.

Los consumos de energía de todos los dispositivos que forman parte del controlador electrónico fueron medidos a plena carga para garantizar que los reguladores de voltaje tuviesen al menos un 30% de holgura en su capacidad máxima de corriente y así evitar sobre calentamientos sin necesidad de ventilación mecánica ni grandes disipadores de calor.

Tabla 5. Consumos de corriente de los dispositivos activos del controlador electrónico.

DC Current (A) consumption chart				
Load	Actuator	Current (A)	Source (V)	Comments
LED +3.3V OK	3mm led	4.70E-03	3.3	On
RTD Converter	MAX31865	5.08E-03	3.3	Converting
VALVE1	Input ULN2003A	8.5E-04	5	On
VALVE2	Input ULN2003A	8.5E-04	5	On
PUMP	Input ULN2003A	8.5E-04	5	On
LED UPLS on	3mm led	4.70E-03	5	On
LED DNLS on	3mm led	4.70E-03	5	On
LED LEVEL	3mm led	4.70E-03	5	On
LED FLOW	3mm led	4.70E-03	5	On
LED MODE on	3mm led	4.70E-03	5	On
LED START on	3mm led	4.70E-03	5	On
LED RESET on	3mm led	4.70E-03	5	On
LED EMRG on	3mm led	4.70E-03	5	On
LED +5V OK	3mm led	4.70E-03	5	On
Microcontroller	PIC18F4550	6.00E-03	5	No load
Display	LCD2004 with I2C	5.46E-02	5	Load with Backlight enabled
VALVE1	Output ULN2003A	2.96E-02	24	Active
VALVE2	Output ULN2003A	2.96E-02	24	Active
PUMP	Output ULN2003A	2.96E-02	24	Active
LED +12V OK	3mm led	4.70E-03	12	On
LED VALVE1	3mm led	4.70E-03	5	On
LED VALVE2	3mm led	4.70E-03	5	On
LED PUMP	3mm led	4.70E-03	5	On
UPLS	Optocoupler	5.00E-03	24	Active
DNLS	Optocoupler	5.00E-03	24	Active
LEVEL	Optocoupler	5.00E-03	24	Active
FLOW	Optocoupler	5.00E-03	24	Active
MODE	Optocoupler	5.00E-03	24	Active
START	Optocoupler	5.00E-03	24	Active
RESET	Optocoupler	5.00E-03	24	Active
EMRG	Optocoupler	5.00E-03	24	Active
Microcontroller	ESP32 WROOM32	1.50E-01	5E	No load with WIFI enabled

Total DC Current (A)
0.49
DC Current 12V PSU (A)
0.37
DC Current 5V 78M05 (A)
0.11
DC Current 5V ESP32 78M05 (A)
0.15
DC Current 3.3V AMS1117 (A)
0.01
DC Current 24V (A)
0.12

En la tabla 5, se aprecian los consumos de corriente de la mayoría de los dispositivos electrónicos activos del circuito controlador, como se puede apreciar la corriente total no supera los 0.5A a plena carga, siendo este consumo realmente menor en condiciones normales de operación ya que es muy difícil que todos los dispositivos estén activados al mismo tiempo. La figura 19 muestra el circuito de alimentación completo.

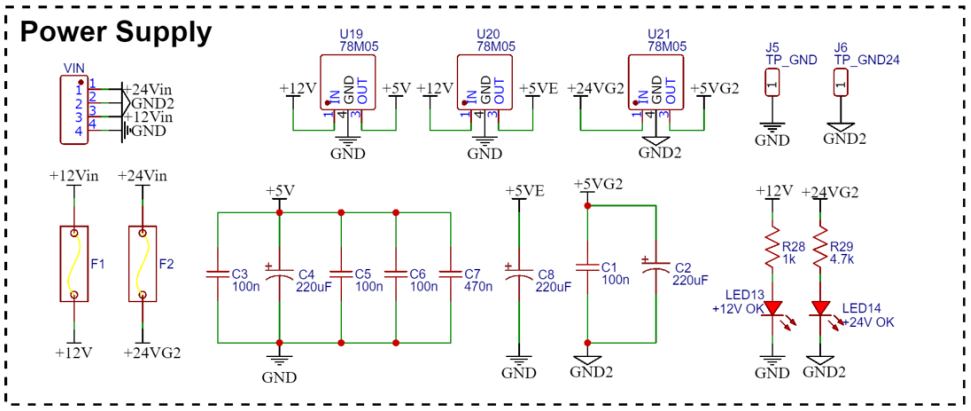


Fig. 19. Circuitos de alimentación del controlador electrónico.

Además de los capacitores electrolíticos de filtrado, el circuito consta de una serie de capacitores de bypass 100nF que son extremadamente importantes debido a que se cuenta con circuitos microcontroladores con altas frecuencias de conmutación que pueden generar ruidos en las señales de entradas y PWM.

6.5. Diseño del Diagrama Esquemático y PCB

Para el diseño del diagrama esquemático y PCB fue especialmente importante la elección del software CAD ya que debía permitir un prototipado rápido y relación directa entre el diagrama esquemático y circuito impreso (PCB), esto requiere de una extensa librería de partes no solo local sino de otras a las cuales pueda accederse a través de Internet. Se eligió el programa en línea EasyEDA [12] que posee un excelente tutorial, una amplia librería de partes y sobre todo acceso gratuito a su herramienta de diseño de diagramas esquemáticos y circuitos impresos. El diagrama esquemático completo puede observarse en el Anexo 3.

Para el diseño del PCB se utilizaron dos capas con diámetros de pista que van desde 0.3mm hasta 2.4mm, los diseños de capa Top, Bottom así como la vista de diseño 3D se muestran en el Anexo 4. Para la elaboración del PCB se utilizaron los servicios de la empresa JLPCB especializada en la fabricación de tarjetas de circuito impreso, así como el montaje de dispositivos SMD, en la figura 20 se muestra el PCB fabricado.

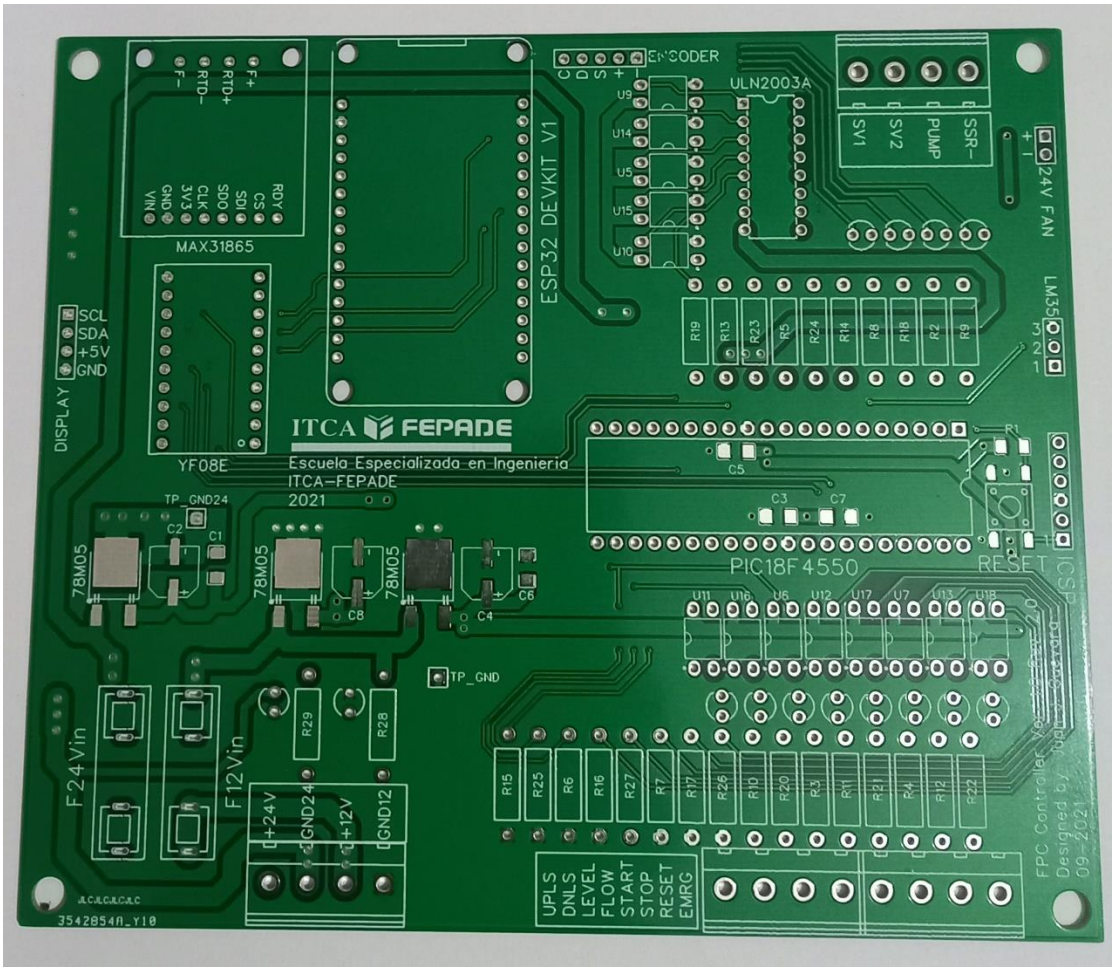


Fig. 20. Tarjeta PCB del controlador electrónico.

6.6. Montaje del Circuito en PCB

El montaje del circuito en PCB fue una tarea relativamente simple debido a la robustes del diseño del PCB, presentó alguna dificultad la soldadura de componentes de montaje superficial.

Para el proceso de soldado de los componentes se utilizó una estación de soldadura con cautín de temperatura regulable la cual se estableció en 400 °C con un estaño de aleación 60/40 y líquido flux no contaminante ni corrosivo.

En la figura 21 se muestra el montaje de todos los dispositivos electrónicos en el PCB.

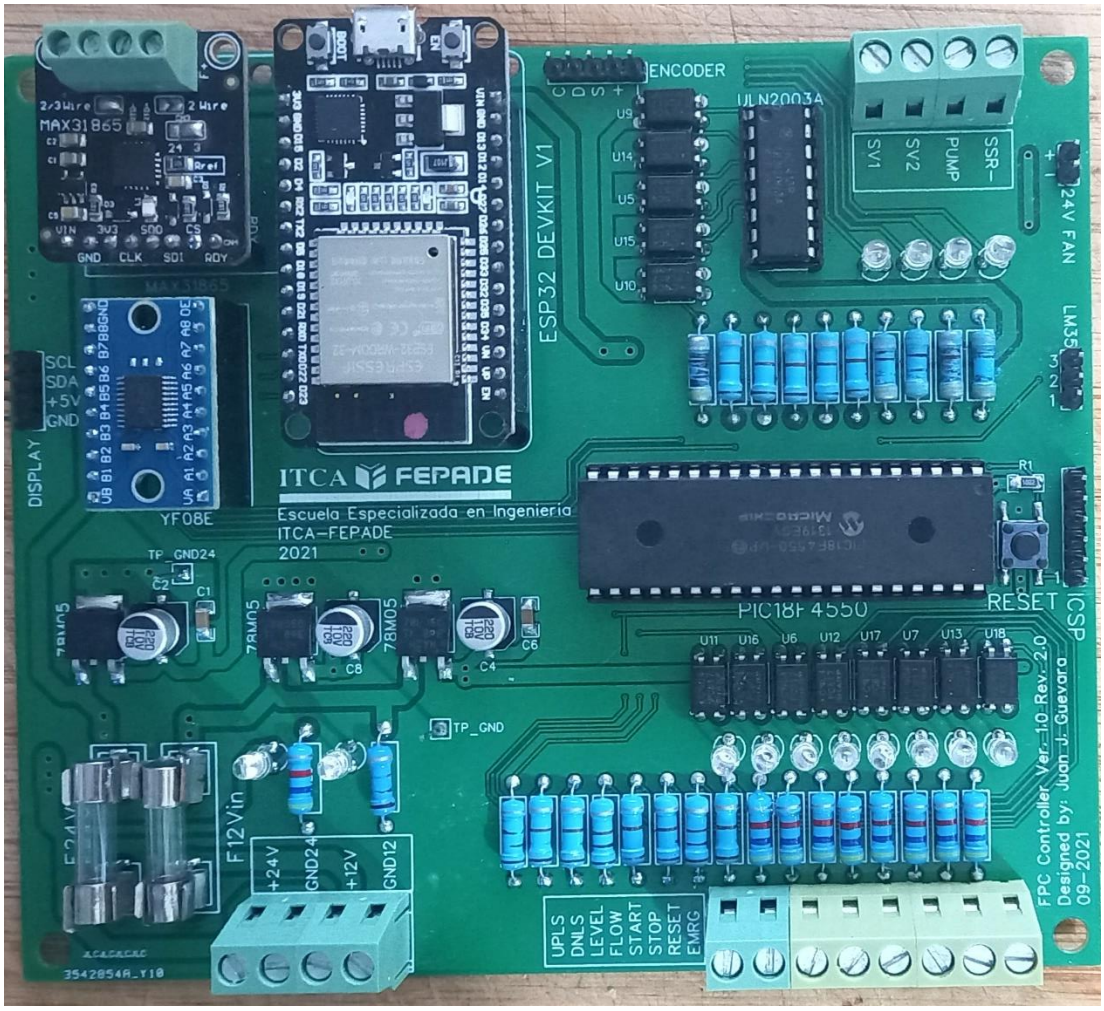


Fig. 21. Tarjeta PCB del controlador electrónico con dispositivos electrónicos soldados.

Para finalizar el trabajo de montaje de componentes, cada circuito electrónico estuvo sujeto a pruebas de funcionamiento individual esto se hizo con el objetivo de garantizar el correcto funcionamiento del hardware del controlador electrónico.

Con todos los procedimientos anteriormente descritos, se dio por finalizado el diseño, montaje y comprobación de funcionamiento del hardware (circuitos electrónicos) del controlador del entrenador FPC con lo cual se pudo pasar a la etapa no menos compleja del diseño del software (firmware).

6.7. Diseño del Firmware del Controlador Electrónico

En el controlador electrónico hay dos microcontroladores: ESP32 y PIC18F4550. Se decidió iniciar el proceso de diseño del firmware del PIC debido a la experiencia que se posee en la programación del dispositivo. El lenguaje de programación seleccionado es el nativo del fabricante Microchip que es XC8 en su versión 2.0 en conjunto con el IDE MPLAB X y programador Pickit 3.

El haber dotado al circuito electrónico de un puerto de programación ICSP permitió agilizar enormemente el proceso de programación del dispositivo, así como la depuración del firmware ya una vez instalado no fue necesario retirar en ningún momento el microcontrolador del PCB.

a. Firmware del controlador de entradas y salidas (PIC18F4550)

Para el diseño del firmware de este controlador se establecieron las siguientes premisas:

1) Todas las acciones de control deben ser sincronizadas por un temporizador interno de 10ms.
2) Para detectar el cambio de estado de los sensores de entrada tipo push button se utilizará la estrategia de lectura basada en estados de máquina [13] por medio interrupciones internas, lo cual constituye la manera más efectiva de efectuar un debouncing por software [14].
3) Se debe evitar en todo momento diseñar estructuras de programación con potenciales riesgos de bloqueo del procesador (tiempos muertos que impiden que otras tareas se lleven a cabo).

Tabla 6. Segmento de código que muestra parte de las rutinas de interrupción que a su vez son gobernadas por el temporizador de 10ms.

```
void __interrupt() high_isr(void){

    INTCONbits.GIEH = 0;

    //Check for 10ms timer flag
    if(INTCONbits.TMR0IF) {
        //Check if it's time to read and update SPST sensors
        Sensors_Update_SPST();

        /*Check if it's time to read ADC (every 1 second)
         * Argument(0) means that AN0 is selected.
         */
        ADC_Read(0);

        //Check if a button sensor is pressed
        Sensors_Update_Buttons();

        //Load count value
        Timer_Load_Count(tmr0lValue, tmr0hValue);

        INTCONbits.TMR0IF = 0;  //Initialize Timer0 interrupt
    }
```

En la tabla 6, se muestra un segmento de código que evita el bloqueo de los distintos procesos que el microcontrolador debe manejar. Dado que todas las funciones se ejecutan a distintos intervalos de tiempo no se interfieren entre ellas produciendo una sensación efectiva de multitarea. Esta metodología de programación es la que se utiliza en los Sistemas Operativos de Tiempo Real (RTOS) [15] que son muy utilizados en sistemas embebidos.

4) Para garantizar legibilidad y rápida depuración, cada dispositivo sensor, actuador, señal, etc. que esté sujeto a configuración y/o programación deberá contar con su fichero de código correspondiente.

5) El método main() del código deberá limitarse a invocar funciones de inicialización, implementar el bucle infinito y realizar el llamado de las funciones correspondientes cuando la bandera de interrupción apropiada sea activada.

Tabla 7. Estructura del método main() del firmware del controlador ES.

```
void main(void) {

    //Internal oscillator
    OSCCON = 0x72;

    //Configure IO ports
    Ports_Init();

    //SPI device is slave
    SPI_Init_Slave();

    //Configure ADC
    ADC_Init();

    //Innitialice for input sensors
    Sensors_Init();

    //innitial actuator values
    Actuators_Init();

    //PWM Inntial configuration
    PWM_Init(period); //Frequency is: 1KHZ

    //ActuatorsStateBk is used to update PORTS when ActuatorsState has
changed
    ActuatorsStateBk = ActuatorsState;


    //Set the 10ms timer
    Timer_Init(tmr0lValue, tmr0hValue);

    //Global and peripheral interrupts are enabled
    INTCONbits.GIEH = 1;
    INTCONbits.GIEL = 1;


    while(1){}
}
```

En la tabla 7 se muestra la aplicación de esta premisa de diseño, se observa que dentro del método main() únicamente se invocan funciones de inicialización y el bucle infinito while(1) está vació, con esto se garantiza que todas las acciones es gestionen a través de interrupciones y del temporizador de 10ms.

La tabla 8 muestra información sobre los ficheros de código que constituyen el firmware del controlador.

Tabla 8. Archivos de código y función como parte del firmware del controlador ES.

Fichero	Función
Main_Firmware.c	Método main(), bucle infinito, invocación de funcionas de inicialización, temporizador general e invocación de rutinas de atención a interrupciones.
actuators.c	Función de inicialización de puertos relacionados con actuadores, rutinas de enmascaramiento y actualización del estado de los actuadores.
adc.c	Función de inicialización del módulo y canal para la conversión de señales de analógico a digital para el sensor de temperatura LM35, rutina de lectura AD y conversión de datos.
conbits.c	Definición de bits de configuración del microcontrolador y variable global de frecuencia de reloj.
ports.c	Rutina de inicialización de los puertos digitales de entrada y salida del microcontrolador
pwm.c	Función de inicialización del módulo CCP1, rutina de activación de la señal PWM, rutinas de actualización del ciclo de trabajo de la señal PWM.
sensors.c	Función de inicialización de sensores interruptores push y SPST, rutinas de actualización de estado de los interruptores.
spi.c	Función de inicialización del bus SPI
timer.c	Función de inicialización de temporizador de 10ms

Finalmente, para la efectiva comunicación a través del Bus SPI del controlador ES y el ESP32 se diseñó una estructura de comandos que se muestra en la tabla 9.

Tabla 9. Comandos de comunicación entre controlador ES y ESP32 a través del bus SPI.

SPI Commands Chart		
Command	Masked	Action
1	No	Send sensores state
2	No	Send actuators state
3	No	Send PWM state
4	No	Send PWM state
5	No	Send Temperature of the controller
6	No	Activate PWM
7	No	Deactivate PWM
8	No	Deactivate SV1
9	No	Activate SV1
10	No	Deativate SV2
11	No	Activate SV2
12	No	Deactivate PUMP
13	No	Activate PUMP
14	No	Deactivate FAN
15	No	Activate FAN
>=128	Yes	Update PWM duty cicle

b. Firmware del controlador principal del SYSCON ESP32

El controlador principal del SYSCCON ESP32 fue seleccionado para garantizar la comunicación entre el servidor de aplicaciones en donde se encuentra el software de gestión y control Web del entrenador FPC. Además, este dispositivo debe comunicarse de forma efectiva con el controlador ES por medio del bus SPI. En síntesis, es el elemento más importante del circuito controlador del entrenador FPC. Debido a la diversidad de tecnologías de comunicación con los que debe interactuar, se evaluaron distintos framewoks de programación buscando principalmente un lenguaje que sea abierto, multiplataforma, de propósito general y que tuviese una versión para microcontroladores como el ESP32 WROOM 32.

El único lenguaje de programación que cumple con estas condiciones es Python ya que además de ser un lenguaje de prototipado rápido (la curva de aprendizaje es muy baja)[16], cuenta con MicroPython que es una implementación de Python 3.4 con todas las prestaciones de las versiones más recientes[17] y dispone de un firmware específicamente diseñado para la familia de controladores ESP32. El firmware es cargado en el microcontrolador utilizando la herramienta esptool de Python por medio de comandos específicos de consola sin necesidad de contar con herramientas especiales de programación (Como por ejemplo el programador Pickit 3 para el PIC18F4550) facilitando el proceso de desarrollo y depuración [18].

Una vez seleccionado el lenguaje de programación se definieron las siguientes premisas de diseño del código:

1) Debido a que se utilizarán sockets TCP como estrategia de comunicación entre el SYSCON y la aplicación Web será necesario crear un firmware con un enfoque 100% asíncrono, esto para garantizar que las funciones que bloqueen procesos (Toda conexión a red por su naturaleza bloquea a otros procesos) se ejecuten de forma parcial evitando que se tengan retardos.

   Para cumplir esta premisa de diseño, el código está casi en su totalidad estructurado en tareas asíncronas que están soportadas por Python. Se hizo un estudio de la documentación de Python[19] y MicroPython[20] relacionada con las corutinas y tareas para diseñar un script de código que intercambia comunicación asíncrona por medio de sockets TCP, en la tabla 10 se aprecia un segmento de código(script) que ejecuta la aplicación Web del lado del servidor de aplicaciones.

Tabla 10. Script de comunicación que implementa un socket TCP que es ejecutado por el cliente (Aplicación Web del servidor).

```
import asyncio
import sys

srvIP = 'Dirección IP'
srvPort = Puerto

async def tcp_echo_client(message):
    reader, writer = await asyncio.open_connection(
        srvIP, srvPort)

    print(f'Send: {message!r}')
    writer.write(message.encode())

    data = await reader.read(100)
    print(f'Received: {data.decode()!r}')

    print('Cerrando la conexion')
    writer.close()

argumento = str(sys.argv[1]) + '\n'
asyncio.run(tcp_echo_client(argumento))
```

En el código mostrado, se puede observar la que la función *tcp_echo_client()* es asíncrona porque en su definición se específica dicha característica por medio de *async def* que está disponible en la librería *asyncio* inicialmente importada. El escript es esencialmente un eco, es decir envía una consulta y obtiene una respuesta por parte del servidor. También se puede observar que el argumento de la función finaliza siempre con un salto de línea `'\n'` que es un requerimiento establecido en el protocolo de comunicación.

Por su parte, el controlador ESP32 implementa un servidor TCP basado en sockets también asíncrono que se muestra en la tabla 11.

Tabla 11. Clase Server que se ejecuta en el controlador ESP32, es asíncrono e implementa un servidor TCP basado en sockets.

```
import usocket as socket
import uasyncio as asyncio
import uselect as select

class Server:

    def __init__(self, host='direccion', port=puerto, backlog=5,
timeout=20):
        self.host = host
        self.port = port
        self.backlog = backlog
        self.timeout = timeout

    async def run(self):
        print('Esperando conexión del cliente...')
        self.cid = 0
        self.server = await asyncio.start_server(self.run_client,
self.host, self.port, self.backlog)
        while True:
            await asyncio.sleep(100)

    async def run_client(self, sreader, swriter):
        self.cid += 1
        print('Conexión establecida del cliente ', self.cid)
        try:
            while True:
                try:
                    res = await asyncio.wait_for(sreader.readline(),
self.timeout)

                except asyncio.TimeoutError:
                    res = b''
                if res == b'':
                    raise OSError
                print('Recibido: {0!r}'.format(res.decode()))

                prueba = res.decode()
                prueba = prueba.replace('\n','')
                print(prueba, len(prueba))

                swriter.write('¡Copiado!')
                await swriter.drain()  # Echo back
        except OSError:
            pass
        print('Cliente {} desconectado.'.format(self.cid))
        await sreader.wait_closed()
        print('Socket del Cliente {} cerrado.'.format(self.cid))

    async def close(self):
        print('Cerrando el servidor')
        self.server.close()
        await self.server.wait_closed()
        print('Servidor cerrado.')

server = Server()
try:
    asyncio.run(server.run())
except KeyboardInterrupt:
```

```
    print('Detenido por medio del KB')  # This mechanism doesn't work on
Unix build.
finally:
    asyncio.run(server.close())
    _ = asyncio.new_event_loop()
```

Este es un código escrito en MicroPython pero puede apreciarse que es muy similar a Python. Todas las funciones definidas son asíncronas. La función *run_client()* es la encargada de "escuchar" las peticiones que vienen del cliente (la aplicación Web) y devolver una respuesta apropiada. El código mostrado es el prototipo de prueba, la codificación final posee una estructura de identificación de comando y envío de respuesta apropiada (ver Protocolo de Comunicación).

2) La clase Server que implementa el servidor asíncrono TCP, debe utilizarse para definir las tareas y tiempos de ejecución de cada una de ellas. En la tabla 12, se puede observar el prototipo de la función *async def run()*, en esta función se definen las tareas y sus tiempos de ejecución.

Tabla 12. Estructura de la función run() que se encarga de definir las tareas asíncronas y sus tiempos de refresco.

```
async def run(self):
        print('Esperando conexión del cliente...')
        self.cid = 0

        """
        Aquí se agregan todas las tareas asíncronas
        """
        # Tareas de estado de máquina
        asyncio.create_task(estado_maquina(estados_tms))
        asyncio.create_task(estado_start(estados_tms))
        asyncio.create_task(estado_stop(estados_tms))
        asyncio.create_task(estado_reset(estados_tms))

        # Tareas de servicio
        asyncio.create_task(leerTemperaturaPT100(pt100_tms))
        asyncio.create_task(updateFromSlave(spi_slv_state_ts))
        asyncio.create_task(actualizarSensores(spi_slv_sensors_tms))
        asyncio.create_task(busSPI(spi_slv_tms))
        asyncio.create_task(actualizarPID(pid_sample_ts))

        # Función para atender solciitudes desde el cliente TCP
        self.server = await asyncio.start_server(self.run_client,
self.host, self.port, self.backlog)
        while True:
            await asyncio.sleep(100)
```

Las tareas asíncronas se dividen en dos tipos:

- **Tareas de estado de máquina**, que representan los cuatro posibles estados del entrenador FPC. Es decir, representan lo que el entrenador está haciendo en ese momento y deben ejecutarse en una secuencia lógica e imperturbable.
- **Tareas de servicio**, están relacionadas con la lectura de sensores, modificación de actuadores, lectura de temperatura, manejo del controlador PID y del bus SPI. Estas tareas proporcionan información para los procesos de los estados de máquina.

Todas las tareas asíncronas son definidas con un tiempo de actualización cuyo valor es experimental. Esto significa que durante la etapa comprobación general del funcionamiento de todo el sistema serán ajustados hasta obtener los mejores resultados. Los tiempos son valores que representan milisegundos (tms) y segundos (ts).

3) Debe poseer funciones y variables que inicialicen y protejan al entrenador y controlador.

En el análisis inicial del funcionamiento del entrenador FPC se determinó que el problema con mayores probabilidades de producirse es el de sobrecalentamiento del agua, evaporación y daño eléctrico de la resistencia calefactora de 600W. Para evitar esto, siempre que se esté ejecutando un proceso de control que active al calefactor se monitorea la temperatura y se apagará el controlador cuando esta supere los 40°C.

Otro problema que puede producirse está relacionado con el flujo del fluido. Debido a que el entrenador no dispone de un control de la potencia de flujo de la bomba de agua, es posible que mediante la activación de perturbaciones (activación del flujo del fluido) se vacíe por completo el tanque 1 haciendo que la bomba trabaje en vacío dañándola con el tiempo. Si bien las cámaras de video mostrarán en tiempo real el nivel del fluido en el tanque 1 al usuario haciendo que este desactive el modo perturbación. Se ha creado una función que corre en segundo plano para desactivar automáticamente el flujo cuando este tenga un nivel demasiado bajo.

Otras posibles fallas como el derrame del fluido no son posibles al desarrollar prácticas de laboratorio ya que la cantidad de fluido total nunca supera la capacidad de los tanques. Si es posible que se derrame el líquido debido a algún fenómeno natural como un terremoto que produzca la rotura de una conexión de las tuberías o de los tanques. Sin embargo, y debido a que, si bien el entrenador será accedido remotamente, siempre será necesario el monitoreo de un operario o responsable del laboratorio que revise antes del inicio las prácticas de laboratorio el estado del entrenador y garantice las condiciones de operación mecánicas y eléctricas.

6.8. Diseño de Aplicativo Web de Plataforma de Teleingeniería

La aplicación desarrollada es de tipo multiplataforma, es decir, que puede ser cargada desde cualquier sistema operativo y dispositivo que se requiera. Para el desarrollo de esta se utilizaron las siguientes tecnologías:

1) Servidor de tipo LAMP

LAMP, es un acrónimo de Linux, Apache, MySQL, PHP. Es una pila popular para crear y desplegar aplicaciones web dinámicas. En esta pila (stack), Linux sirve como el sistema operativo para la aplicación web. MySQL se utiliza como base de datos. Apache se utiliza como servidor web. PHP se utiliza para procesar contenido dinámico. En algunas otras variantes de esta pila, Perl se utiliza en lugar de PHP o Python. Sin embargo, para nuestro caso, vamos a instalar PHP. En otras pilas se suele utilizar MongoDB como base de datos.

2) PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación interpretado que se utiliza para la generación de páginas web de forma dinámica. Este código se ejecuta al lado del servidor y se incrusta dentro del código HTML. Cabe destacar que es un lenguaje de código abierto, gratuito y multiplataforma.

3) **MySQL**

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo,12 y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

4) **Laravel**

Laravel es uno de los frameworks de código abierto más fáciles de asimilar para PHP. Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

5) **NodeJS**

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, por ejemplo, servidores web.

6) **Vue.JS**

Vue es un framework progresivo para construir interfaces de usuario. A diferencia de otros frameworks, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas Single-Page Applications cuando se utiliza en combinación con herramientas modernas y librerías de apoyo.

En el anexo 8, se muestra la guía de instalación de la aplicación Web de la plataforma de Teleingeniería.

6.9. Comprobación y Puesta a Punto de la Comunicación entre la Aplicación Web y el Controlador Electrónico

Inicialmente, se realizó al montaje de todos los dispositivos de hardware que en su conjunto conforman la plataforma de Teleingeniería. Para realizar este trabajo se rediseñó el diagrama de control y potencia del entrenador cuyo diseño final se muestra en el anexo 4. A partir de esta disposición se realizaron las conexiones en el tablero de control tal como se muestra en la figura 22.

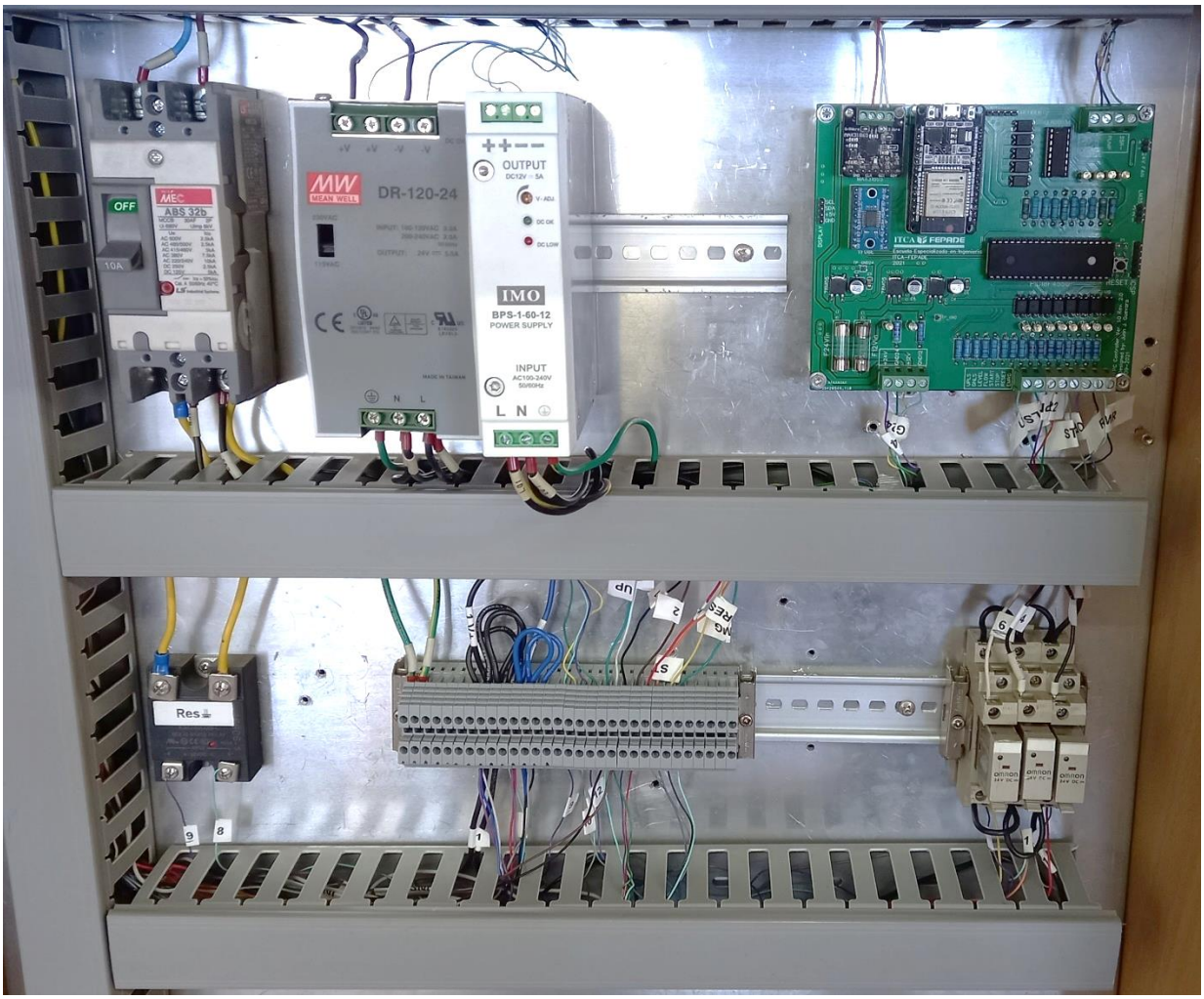


Fig. 22: Conexiones de potencia en el tablero de control del entrenador FPC.

Como se puede apreciar, la incorporación del controlador electrónico (esquina superior derecha) simplificó muchísimo las conexiones y redujo la cantidad de dispositivos entrenador original.

El hardware del servidor lo constituye una minicomputadora Raspberry Pi 4 de 4GB de memoria RAM y 32GB de almacenamiento en memoria SD, para su montaje se diseñó un case de protección y para las cámaras brazos de soporte ajustables, se muestran en la figura 23. Mientras que el montaje final del entrenador se muestra en la figura 24.

Fig. 23: Case de protección para el servidor basado en Raspberry Pi y Brazos ajustables para las cámaras de video.

Fig. 24: Vista general del montaje final del entrenador FPC.

Una vez finalizada la instalación de los dispositivos del entrenador FPC se procedió a realizar la configuración de las direcciones IP del controlador electrónico y servidor Raspberry PI. Para ambos casos se coordinó con la dirección de informática la asignación de IP mediante reserva por dirección MAC en la VLAN de los laboratorios del edificio B, concretamente en el laboratorio B202 que es el lugar en donde se ha realizado todo el trabajo de investigación, pruebas y montaje.

Una vez comprobado el acceso en las subredes de datos del campus de ITCA-FEPADE sede central, se solicitó informática el acceso a la plataforma de Teleingeniería desde fuera del campus a través de una IP pública consiguiendo resultados satisfactorios. La tabla 13 muestra las direcciones IP asignadas a los dispositivos.

Tabla 13: Direcciones IP de acceso a los recursos de la plataforma de Teleingeniería.

Dispositivo o recurso	Observaciones	Dirección IP
Raspberry Pi 4	Servidor de Streaming para las video cámaras y aplicación de gestión y prácticas del entrenador	172.16.48.101
ESP32 WROOM32	Controlador SYSCON del entrenador FPC	172.16.48.102
IP pública para acceso fuera de campus de ITCA-FEPADE		190.5.143.147

Habiendo conectado la plataforma de Teleingeniería a la red de datos se comprobó el funcionamiento de los módulos de la aplicación de gestión y prácticas (fig. 25).

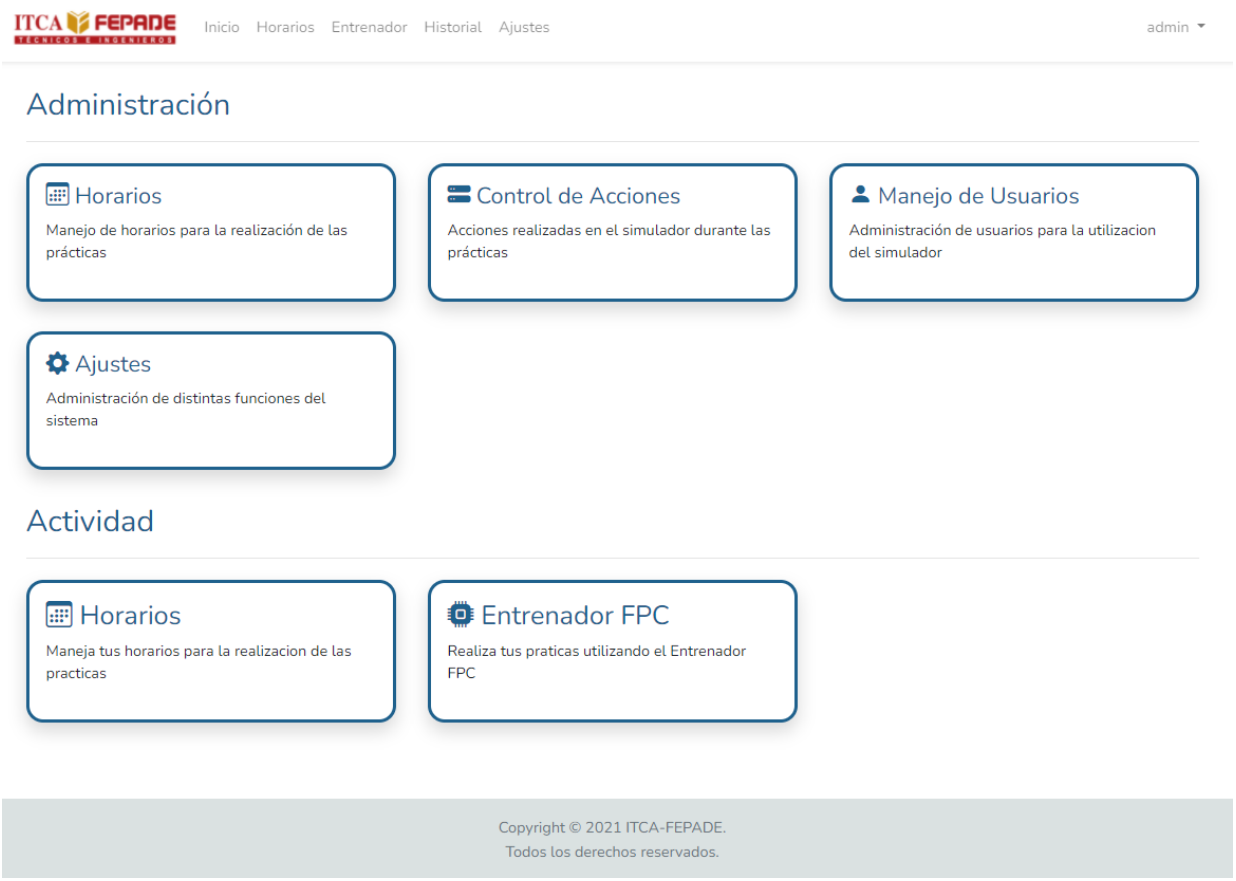


Fig. 25: Módulos de gestión y prácticas de la aplicación Web de la plataforma de Teleingeniería.

En la figura 26, se observa el funcionamiento del módulo de prácticas de la aplicación. Puede observarse la activación de las cámaras de video.

Para que los usuarios del módulo de prácticas tuvieran una visión clara de que las órdenes que se proporcionen de forma remota se estén reflejando en el entrenador, se decidió que una de las cámaras mostrase el tablero de control local donde se observa la activación/desactivación de las válvulas solenoides de las tuberías y la bomba de agua del entrenador, así como el estado de los sensores capacitivos que detectan el nivel del fluido en el tanque 2. Mientras que, la segunda cámara mostraría una vista general de las tuberías y tanques del entrenador en donde se pueda apreciar el flujo del fluido.

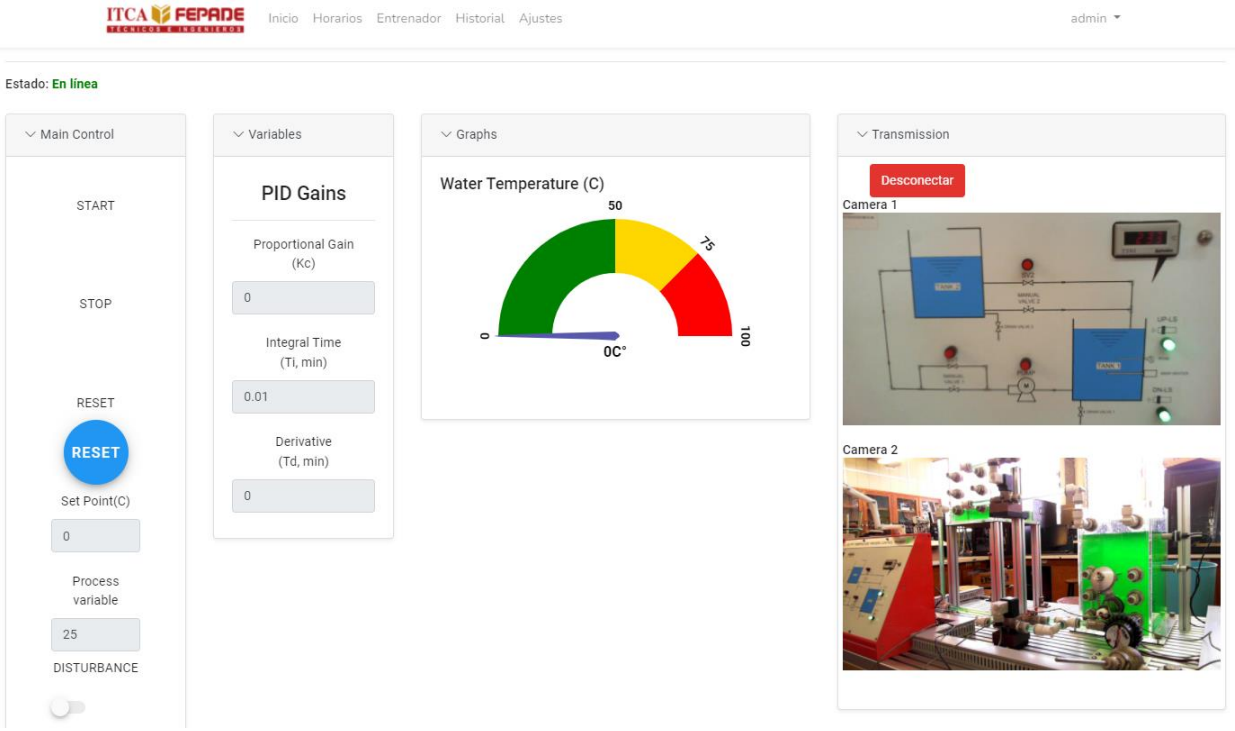


Fig. 26: Módulo de prácticas de la plataforma de Teleingeniería.

En la figura 26, puede apreciarse que el módulo de prácticas está conformado por tres columnas de información. La columna de la izquierda denominada *Main Control* muestra el estado de máquina del entrenador, la columna a la derecha la columna *Variables* permite la asignación y modificación de las variables de control del proceso de forma dinámica, la columna *Graphs* es la zona de las gráficas de estado del proceso y finalmente la columna *Transmission* muestra las cámaras de video.

Para la puesta a punto de la plataforma, se diseñaron dos documentos de referencia para regir la comunicación entre el controlador electrónico del entrenador FPC y la aplicación Web.

a) Diagrama de estados de máquina del entrenador FPC, en este documento que es utilizado en aplicaciones de control industriales, se muestran los tres estados de la máquina (entrenador) que son RESET, START y STOP.

   En estado RESET la máquina realiza un vaciado del tanque 1 hasta desactivar el sensor capacitivo DNLS (Down sensor) mediante la activación de la bomba de agua y la apertura de la válvula SV1. Al finalizar este proceso el usuario recibe en el módulo de prácticas de la aplicación un mensaje que le indica que el RESET se ha aplicado y le permitirá enviar la orden START.

   En el estado START, inicialmente se realiza un llenado del tanque 1 a través de la apertura de la válvula SV2. El fluido cae por gravedad desde el tanque 2 al tanque 1. La válvula SV2 se cierra hasta que el sensor capacitivo de nivel superior UPLS (up sensor) se activa. En este momento se empiezan

a aplicar el calentamiento del fluido a partir de los valores de SET POIN y variables P, I y D definidas por el usuario que está haciendo la práctica.

El estado STOP puede ser aplicado solamente si se está en estado START y lo hace desactivando remotamente el controlador PID del entrenador FPC. En este momento el entrenador cierra su ciclo y se queda esperando a recibir la orden para aplicar el estado RESET para así repetir todo el proceso nuevamente.

El diagrama de estado de máquina del entrenador FPC se muestra en el anexo 6.

b) Diagrama de flujo y estado de máquina del controlador y la aplicación, en donde se describe una secuencia más completa del proceso de comunicación entre la aplicación y el controlador electrónico por medio de un diagrama de flujo. Este diagrama es un insumo importante para los programadores de la aplicación Web ya que acerca más el estado de máquina del proceso industrial a una estructura de programación de una aplicación. El diagrama completo se muestra en el anexo 7.

6.10. COMPROBACIÓN GENERAL DE FUNCIONAMIENTO

Las pruebas generales de funcionamiento se hicieron para detectar potenciales errores sobre todo de comunicación entre la aplicación Web y el controlador electrónico. También tiene como objetivo mejorar el firmware de los microcontroladores del controlador en un proceso de mejora continua.

Esta comprobación se realiza mediante el desarrollo de prácticas de laboratorio remotas ya sea dentro de la red de datos de ITCA-FEPADE como fuera del campus. Entre otras cosas, este proceso ha permitido ajustar los tiempos de refresco de lectura de la temperatura de la RTD PT100, de actualización del controlador PID para reducir lecturas erróneas que afecten el control del proceso. Permitieron, mejorar las rutinas de protección del entrenador FPC para reducir los potenciales de daños por pérdidas de conexión o comandos no recibidos desde la aplicación.

Es importante aclarar que los resultados obtenidos en esta etapa de la investigación han servido para elaborar las recomendaciones que se describen más adelante y también es un proceso que aún continua como todo desarrollo de un sistema que integra hardware y software.

7. RESULTADOS

El resultado más importante es sin duda la plataforma de Teleingeniería, pero con el diseño y montaje de ésta, explicado en el apartado metodología de la investigación, se han obtenido una serie de resultados que han enriquecido el conocimiento, no solo de los docentes investigadores, sino también de los estudiantes que colaboraron con esta actividad, la cual establece un nuevo estado de la técnica para futuros proyectos en donde se integren elementos de hardware y software a través de redes de datos.

A continuación, se describen de los resultados más significativos.

1. Diagrama de bloques del controlador electrónico (Anexo 1), que puede y debe servir como referencia para la creación de otros controladores para el control de procesos de tipo industrial.
2. Protocolo de comunicación (Anexo 2), que es una referencia de comunicación asíncrona entre dos plataformas (Aplicaciones Web y circuitos electrónicos) que utilizan tecnologías diferentes y que con la implementación de la Industria 4.0 y las clases a distancia cada vez están más unidas.
3. Diagrama esquemático del controlador electrónico del entrenador FPC (Anexo 3) que define un estado del arte como punto de partida para el diseño de otros circuitos controladores que manejen procesos industriales similares.

4. Diseño de Circuito Impreso de doble cara del controlador electrónico con software CAD gratuito, (Anexo 4) que demuestra que se puede hacer un diseño profesional incluyendo vistas 3D del resultado final, sin incurrir en adquisición de software con licenciamiento elevado.
5. Diagramas de estado de máquina (Anexo 6) y de flujo-estado (Anexo 7) para plataformas que servirá como referencia de un modelo de comunicación entre un circuito de control electrónico industrial y una aplicación Web que se ejecuta en un servidor remoto.

Adicionalmente, forman parte de los resultados de este proyecto de investigación:

6. El código fuente de la aplicación Web para la gestión y desarrollo de prácticas de laboratorio.
7. El código fuente del firmware del controlador SYSCON ESP32 y del controlador ES PIC18F4550.
8. Manual de servicio de la aplicación Web de la plataforma de Teleingeniería.
9. Guías de práctica de laboratorio para nivel básico, intermedio y avanzado.

8. CONCLUSIONES

1) Es posible diseñar un controlador electrónico de bajo costo que cumpla con los requerimientos de diseño de una máquina destinada a realizar un control de proceso industrial. Una solución de hardware como la diseñada en este proyecto de investigación puede llegar a tener un costo hasta cinco veces superior al utilizado, con la desventaja que por lo general se trabaja con una plataforma cerrada y el agregar más prestaciones a la solución requiere siempre de una mayor inversión.
2) Es posible conectar dinámicamente una aplicación que se ejecuta en un servidor remoto de forma eficiente y efectiva con un controlador electrónico, siempre que se implemente un enfoque de comunicación y programación asíncrono.
3) Los sockets TCP son especialmente útiles cuando se diseña una solución en la cual se debe intercambiar información en tiempo real y la conexión debe mantenerse activa por mucho tiempo.
4) Se pueden realizar prácticas de laboratorio a distancia en entrenadores que tradicionalmente se manipulan únicamente de forma presencial. La plataforma de Teleingeniería diseñada es única en el país, ya que no es un simulador, sino que todas las acciones de control que realiza el usuario se hacen en el entrenador físico, pero a distancia. Además, la incorporación del video en tiempo real mejora la experiencia del usuario, ya que puede observar de forma efectiva que todas sus acciones se ejecutan directamente en el entrenador como si estuviera en una práctica presencial.

9. RECOMENDACIONES

1) Como la plataforma de Teleingeniería que integra diversos sistemas y tecnologías está sujeta a un proceso de depuración y mejora continua, por lo que se deben realizar ajustes, correcciones de errores y adecuaciones que mejoren la experiencia del usuario.
2) La plataforma de Teleingeniería diseñada demuestra que es posible modificar la circuitería electrónica de los entrenadores de control de procesos industriales que se usan en prácticas de laboratorio presenciales y hacer que estos sean gestionados a distancia. Por lo que se sugiere expandir las prestaciones de la plataforma existente, incorporando módulos de software y hardware para los demás entrenadores de control de procesos con los que cuenta el laboratorio J101 de ITCA-FEPADE.

3) La aplicación Web de la plataforma de Teleingeniería es multiplataforma y está diseñada para que pueda migrarse fácilmente a otros servidores. Se sugiere que esta sea instalada en un servidor de Nube o Hosting Web que proporcione un ancho de banda más alto y redundancia a fallos por desconexiones.

4) Las cámaras de video utilizadas son Webcam, por lo que la definición y enfoque está limitado. Se recomienda evaluar y utilizar cámaras IP de alta definición e inclusive aumentar el número de cámaras para mejorar la experiencia del usuario. Esto trae consigo el requerimiento de aumento del ancho de banda de la red de datos, ya que se implementa un streaming de video que maneja grandes cantidades de información.

5) En las gráficas del módulo de prácticas del entrenador en la aplicación Web, se pierde continuidad cuando se actualiza la información. No se pudo realizar una modificación del código de las librerías de las gráficas, ya que hacerlo requería un tiempo considerable. Se recomienda hacer una evaluación del código de las librerías de las gráficas y de ser necesario diseñar unas librerías a medida.

6) El protocolo de comunicación si bien es funcional, puede ser mejorado para reducir el tamaño de los paquetes de datos que se envían por la red. Se sugiere estudiar el protocolo y hacer las mejoras que se consideren necesarios para asegurar un flujo de información mas liviano, sin menoscabo de la calidad de ésta.

10. GLOSARIO

Aislamiento Galvánico. Es un método de protección que sirve para separar dos circuitos sin que haya contacto alguno entre ambos y a su vez poder transferir la energía de un lado a otro.

Bus SPI. El SPI es un protocolo de comunicación síncrona de 4 hilos, entre dispositivos electrónicos presentado por Motorola en 1982, que ha ganado bastante aceptación en la industria como sistema de comunicación de muy corta distancia, normalmente dentro la placa de circuito impreso. Es un protocolo de transmisión que permite alcanzar velocidades muy altas y que se diseñó pensando en comunicar un microcontrolador con distintos periféricos y que funciona a full dúplex (Una forma retorcida de decir que puede enviar y recibir datos al mismo tiempo). SPI utiliza una solución síncrona, porque utiliza unas líneas diferentes para los datos y el Clock.

Controlador Lógico Programable. PLC o Controlador Lógico Programable son dispositivos electrónicos muy usados en Automatización Industrial. Un PLC controla la lógica de funcionamiento de máquinas, plantas y procesos industriales, procesan y reciben señales digitales y analógicas y pueden aplicar estrategias de control. Programmable Logic Controller o Controlador lógico programable. Se trata de un equipo electrónico, que, tal como su mismo nombre lo indica, se ha diseñado para programar y controlar procesos secuenciales en tiempo real.

Framework. Un framework es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código.

Optoacoplador. Un optoacoplador también llamado optoaislador, es un circuito electrónico que funciona como un interruptor aislado ópticamente. Es decir, que permite una conexión eléctricamente aislada entre dos circuitos que operan a distintos voltajes. Está construido por un led y un circuito de control activado por luz infrarroja.

Señal PWM. La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

Set Point. Set Point SP o Consigna es el valor deseado de la variable de proceso, es decir, la consigna. Es el valor al cual el control se debe encargar de mantener la PV

Streaming de video. Es un tipo de tecnología multimedia que envía contenidos de vídeo y audio a su dispositivo conectado a Internet. Esto le permite acceder a contenidos (TV, películas, música, pódcast) en cualquier momento que lo desee, en un PC o un móvil, sin someterse a los horarios del proveedor.

SYSCON. Es el nombre que recibe a circuito de control lógico de un aparato electrónico y que integra tanto al microcontrolador, circuito oscilador, de RESET y algunos casos memoria generalmente de tipo EEPROM. El SYSCON controla el funcionamiento de todas las etapas eléctricas y mecánicas de una máquina.

11. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Arbildo López, "El control de procesos industriales y su influencia en el mantenimiento", pp. 35–49, may 2011.

[2] K. Ogata, *Ingeniería de Control Moderna*, Quinta Ed., vol. 1. Madrid: Pearson, 2009.

[3] A. Roca, *Control Automático de Procesos Industriales*, vol. 1. Diaz de Santos, 2014.

[4] "What is an Embedded System? Definition and FAQs | OmniSci". https://www.omnisci.com/technical-glossary/embedded-systems (consultado el 18 de enero de 2022).

[5] "Transmission Control Protocol", Internet Engineering Task Force, Request for Comments RFC 793, sep. 1981. doi: 10.17487/RFC0793.

[6] N. H. Tollervey, *Programming with MicroPython*, Primera. Estados Unidos: O'REILLY, 2017.

[7] "Chipsets | Espressif Systems". https://www.espressif.com/en/products/socs (consultado el 27 de enero de 2022).

[8] "MQTT - The Standard for IoT Messaging". https://mqtt.org/ (consultado el 23 de febrero de 2022).

[9] I. Hübschmann, "A Complete Guide to REST APIs in IoT", *Nabto*, el 1 de marzo de 2021. https://www.nabto.com/rest-api-iot-guide/ (consultado el 23 de febrero de 2022).

[10] E. Electronics *et al.*, "Pull Up and Pull Down Resistor", *Circuit Digest*, el 29 de noviembre de 2018. https://circuitdigest.com/tutorial/pull-up-and-pull-down-resistor (consultado el 24 de febrero de 2022).

[11] "DC Solid State Relays", *celduc® relais*, el 4 de enero de 2021. https://www.celduc-relais.com/en/dc-solid-state-relays/ (consultado el 24 de febrero de 2022).

[12] "EasyEDA Tutorial". https://docs.easyeda.com/en/FAQ/Editor/index.html (consultado el 24 de febrero de 2022).

[13] T. (Tommy) Gartlan, "Debouncing Push-Buttons Using a State Machine Approach", *EEWeb*, el 6 de junio de 2018. https://www.eeweb.com/debouncing-push-buttons-using-a-state-machine-approach/ (consultado el 24 de febrero de 2022).

[14] B. says, “Ultimate Guide to Switch Debounce (Part 8)”, *EEJournal*, el 16 de abril de 2020. https://www.eejournal.com/article/ultimate-guide-to-switch-debounce-part-8/ (consultado el 24 de febrero de 2022).

[15] “Why RTOS and What is RTOS?”, *FreeRTOS*. https://www.freertos.org/about-RTOS.html (consultado el 26 de febrero de 2022).

[16] “An Overview of Packaging for Python — Python Packaging User Guide”. https://packaging.python.org/en/latest/overview/ (consultado el 27 de febrero de 2022).

[17] “MicroPython language and implementation — MicroPython 1.18 documentation”. https://docs.micropython.org/en/latest/reference/index.html (consultado el 27 de febrero de 2022).

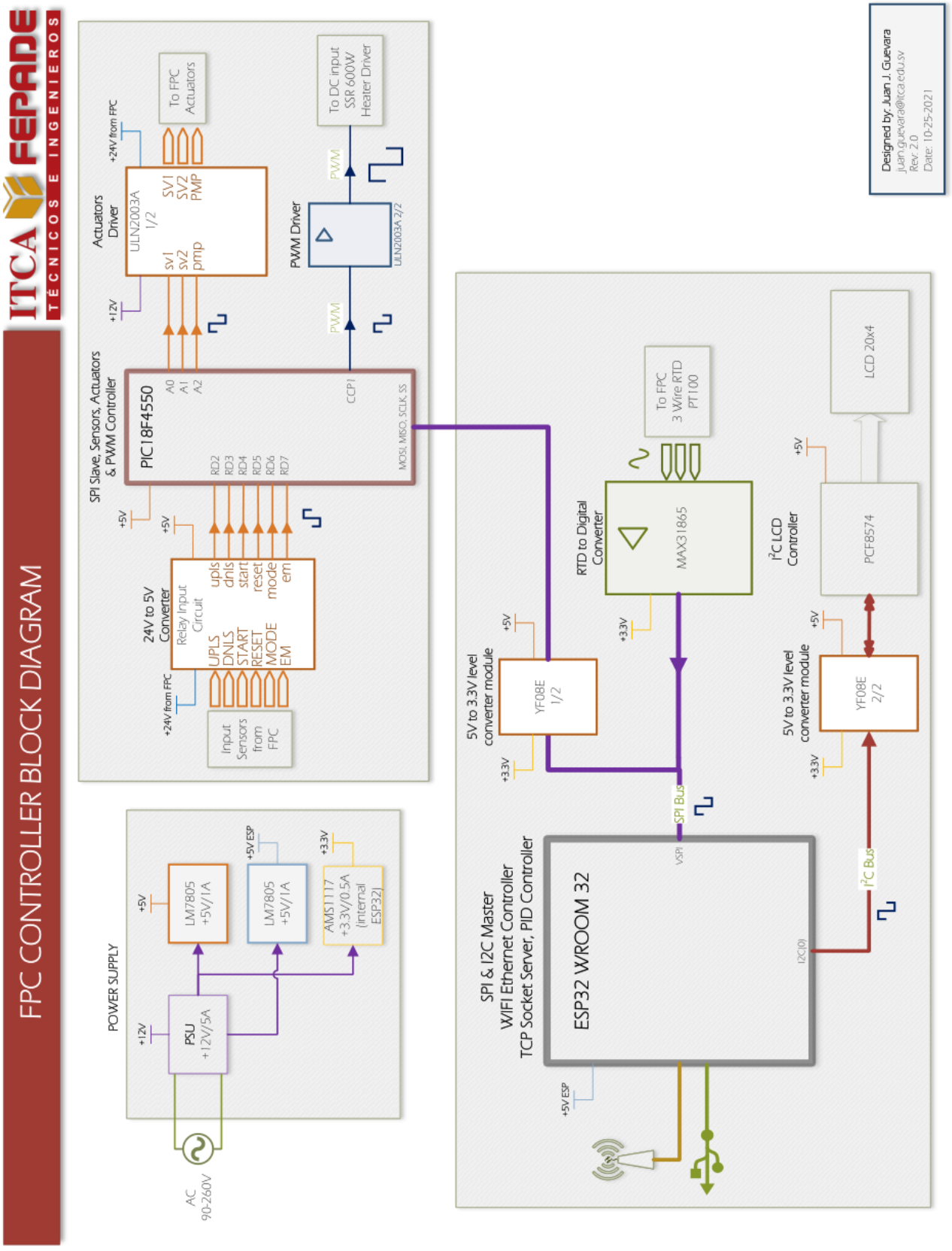
[18] “1. Getting started with MicroPython on the ESP32 — MicroPython 1.18 documentation”. https://docs.micropython.org/en/latest/esp32/tutorial/intro.html#esp32-intro (consultado el 27 de febrero de 2022).

[19] “Corrutinas y Tareas — documentación de Python - 3.10.2”. https://docs.python.org/es/3/library/asyncio-task.html (consultado el 27 de febrero de 2022).

[20] P. Hinch, *Asynchronous programming in MicroPython*. 2022. Consultado: el 27 de febrero de 2022. [En línea]. Disponible en: https://github.com/peterhinch/micropython-async/blob/5c5e6ba44a4b88a382329c178ff8d018e6ae73b5/v3/docs/TUTORIAL.md

12. ANEXOS

12.1. ANEXO 1. DIAGRAMA A BLOQUES DEL CONTROLADOR ELECTRÓNICO DE LA PLATAFORMA DE TELEINGENIERÍA



ITCA FEPADE
TÉCNICOS E INGENIEROS

Escuela de Ingeniería Eléctrica y Electrónica

Proyecto

Diseño e implementación de Plataforma de Teleingeniería
para prácticas en tiempo real de laboratorio a distancia
de control de procesos industriales.
Aplicación en Laboratorio de Electrónica de ITCA-FEPADE Sede Central.

Protocolo de Comunicación entre Controlador del Entrenador FPC y Servidor de Aplicaciones

Elaborador por:

Juan J. Guevara

juan.guevara@itca.edu.sv

Noviembre de 2021

Tabla de transacciones de DATOS con el Servidor de Aplicaciones

ESTADO DE SENSORES DE ENTRADA					
NOMBRE	**TIPO DE DATO**	**DESCRIPCION**	**RANGO DE VALORES**	**ENCAPSULAMIENTO**	**ACCION-DEFAULT**
UPLS	BIT	Sensor capacitivo de nivel superior de H2O	1/0	Byte de estado	NA
DNLS	BIT	Sensor capacitivo de nivel inferior de H2O	1/0		
LEVEL	BIT	Entrenador en modo de ajuste de nivel	1/0		
FLOW	BIT	Entrenador en modo normal de operación	1/0		
START	BIT	Inicio de funcionamiento	1/0		
STOP	BIT	Detiene el funcionamiento	1/0		
RESET	BIT	Modo de operación del entrenador que establece las condiciones de inicio (es un RESET) o acción de control	1/0		
EMERGENCY	BIT	Detención forzada del entrenador, se deben detener todas las acciones sin aplicar un reinicio. Requiere acción humana en sitio	1/0		

ACTUADORES / SALIDAS					
NOMBRE	**TIPO DE DATO**	**DESCRIPCION**	**DESCRIPCION**	**ENCAPSULAMIENTO**	**ACCION-DEFALUT**
SOL VALVE 1	BIT	Válvula de paso inferior	1/0	Byte de estado	NA
SOL VALVE 2	BIT	Válvula de paso superior	1/0		
PUMP	BIT	Bomba de H2O	1/0		

PARAMETROS PARA EL CONTROLADOR					
NOMBRE	**TIPO DE DATO**	**DESCRIPCION**	**DESCRIPCION**	**ENCAPSULAMIENTO**	**PERMISO-DEFAULT**
TEMPERATURA (Variable de Proceso)	FLOAT	Temperatura actual en grados Celsius de H2O medida por el sensor PT100	0.00 a 100.00	Cadena	R-0.00
SETPOINT	FLOAT	Es la temperatura en grados Celsius a la cual se desea calentar el H2O	0.00 a 100.00	Cadena	RW-0.00
PROPORCIONAL	FLOAT	Valor de la ganancia proporcional del controlador PID, un P = 0.00 indica que está apagado	0.00 a 1000.00	Cadena	RW-0.00
INTEGRAL	FLOAT	Valor de la ganancia integral del controlador PID, un I = 0.000 indica que está apagado	0.000 a 1000.00	Cadena	RW-0.000
DERIVATIVO	FLOAT	Valor de la ganancia derivativa del controlador PID, un D = 0.000 indica que está apagado	0.000 a 1000.00	Cadena	RW-0.000
CICLO DE TRABAJO DEL CALENTADOR	FLOAT	Corresponde al porcentaje de energía que se está aplicando al calentador de agua del entrenador, es un valor float de entre 0.0 y 100.0	0.0 a 100.0	Cadena	R-0
REFRESCO	FLOAT	Tiempo estimado de actualización de parámetros (en segundos) entre controlador y servidor de aplicaciones	0.1 a 30	Cadena	RW-1.0

Estructura de los bytes de estado del controlador

LEYENDA			
R	**W**	**-**	**0/1**
Lectura	Escritura		Valor por defecto

*Si un bit es R, solo podrá ser leído por el servidor de aplicaciones

**Si un bit es RW, podrá ser leído y modificado por el servidor de aplicaciones

***El valor por defecto se establece al iniciar el controlador o cuando se produce un RESET

SENSORES DE ENTRADA

R-0	R-0	R-0	RW-0	RW-1	RW-0	R-0	R-0
EMERGENCY	**RESET**	**STOP**	**START**	**FLOW**	**LEVEL**	**DNLS**	**UPLS**
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Ejemplo: si byte enviado es igual a:

0	0	0	1	1	0	0	0

Se leería: el entrenador está funcionando (START = 1) en modo flujo (FLOW = 1)

ACTUADORES

					RW-0	RW-0	RW-0
NA	**NA**	**NA**	**NA**	**NA**	**PUMP**	**SOLV2**	**SOLV1**
bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**

Ejemplo: si byte enviado es igual a:

X	X	X	X	X	1	1	1

Se leería: el entrenador está funcionando en modo 2, hay flujo de agua porque están abiertas las válvulas solenoides inferior y superior, así como la bomba

Comandos del servidor de aplicaciones

Estos son los comandos que el servidor de aplicaciones deberá enviar al controlador. El controlador implementa un servidor TCP Socket asíncrono y está codificado para leer cadenas que deben finalizar con un salto de línea \n.

El servidor de aplicaciones deberá realizar una función de cliente TCP por medio de sockets, se sugiere que estos sean asíncronos.

Comandos para leer el estado del controlador

Sintaxis: Comando\n

Descripción: el servidor de aplicaciones solicita que el controlador le envíe los datos de estado solicitados. El comando debe finalizar con salto de línea \n.

Identificadores de los datos: estos se especifican detalladamente en la Tabla de transacciones de datos.

- 20: **byte de estado de actuadores**, el controlador enviará el byte de estado de los sensores. El servidor de aplicaciones deberá aplicar un enmascaramiento para separar los bits e interpretar el estado de acuerdo con la estructura de los bytes de estado.
  Ej. 20\n <comando>
  9 <byte respuesta>

- 21: **Ciclo de trabajo del controlador**, el controlador enviará el valor actual del ciclo de trabajo del calentador que tiene en ese momento, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario. Esta cadena representa un valor porcentual entre 0.0 y 100.0
  Ej. 21\n <comando>
  52.1 <valor respuesta>

- 22: **Temperatura del agua**, el cliente solicita la temperatura actual del sensor PT100. Se envía una cadena con precisión de dos decimales que representa la temperatura en grados Celsius.
  Ej. 22\n <comando>
  30.25 <valor respuesta>

- 23: **SETPOINT**: el controlador enviará el valor actual del setpoint que tiene establecido, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario.
  Ej. 23\n <comando>
  35.5 <valor respuesta>

- 24: **PROPORCIONAL**, el controlador enviará el valor actual de la variable proporcional que tiene establecida, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario.
  Ej. 24\n <comando>
  10.0 <valor respuesta>

- 25: **INTEGRAL**, el controlador enviará el valor actual de la variable integral que se tiene establecida, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario.
  Ej. 25\n <comando>
  0.01 <valor respuesta>

- 26: **DERIVATIVO**, el controlador enviará el valor actual de la variable derivativa que se tiene establecida, se enviará una cadena que el servidor deberá convertir en su valor real float si fuese necesario.
  Ej. 26\n <comando>
  0.001 <valor respuesta>

- 100**: Consulta de Estado de Máquina**, el servidor de aplicaciones consulta al controlador electrónico sobre el estado de máquina actual. Las respuestas pueden ser:

Valor de respuesta	Estado
200	Se terminó de aplicar RESET
201	Se está aplicando RESET
202	Se terminó de aplicar STOP
203	Se está aplicando STOP
204	Se terminó de aplicar START
205	Se está aplicando START
206	El controlador está EN LINEA o LISTO para trabajar

Comandos para actualizar el estado del controlador

Sintaxis: Comando\n

Descripción: el servidor de aplicaciones está indicando que enviará un dato de actualización al controlador. El proceso de la transacción es el siguiente:

1. El servidor de aplicaciones envía el comando ***<comando>\n***, en donde <comando> identifica lo que se va a actualizar, el carácter de fin de línea ***\n*** indica la finalización del comando.
2. El controlador recibirá el comando y evaluará si el **id** corresponde a un campo actualizable, en caso afirmativo enviará una respuesta de acuerdo con la naturaleza de la acción a realizar.
3. Los comandos relacionados con la actualización de variables para el controlador PID (Setpoint, Proporcional, Integral, Derivativo) requieren de dos transacciones TCP, la primera es para indicar que variable se va a actualizar y la segunda para especificar el nuevo valor de la variable.

Identificadores con permisos de escritura: son aquellos en los cuales el servidor de aplicaciones tiene permiso de escritura de acuerdo con la Tabla de transacciones de datos y estructura de los bytes de estado.

- 10: **Activar perturbación**, si la temperatura en el tanque 2 se ha establecido, se activarán las válvulas SV1, SV2 y la bomba.
  Ej. 10\n <comando>
  OK <respuesta>

- 11: **Desactivar perturbación**, se desactivarán las válvulas SV1, SV2 y la bomba.
  Ej. 11\n <comando>
  OK <repuesta>

- 12: **Inicio (START),** Si la variable P es superior a 0, entonces se activará el controlador PID y se activará el módulo PWM del controlador.
  Ej. 12\n <comando>
  OK <respuesta>

- 13: **Detención (STOP)**, se desactiva el controlador PID y el módulo PWM.
  Ej. 13\n <comando>
  OK <respuesta>

- 14: **Inicialización (RESET)**, se desactiva el controlador PID, el módulo PWM y se limpian las variables: SP, P, I, D y Temperatura en Tanque 2.
  Ej. 14\n <comando>
  OK <respuesta>

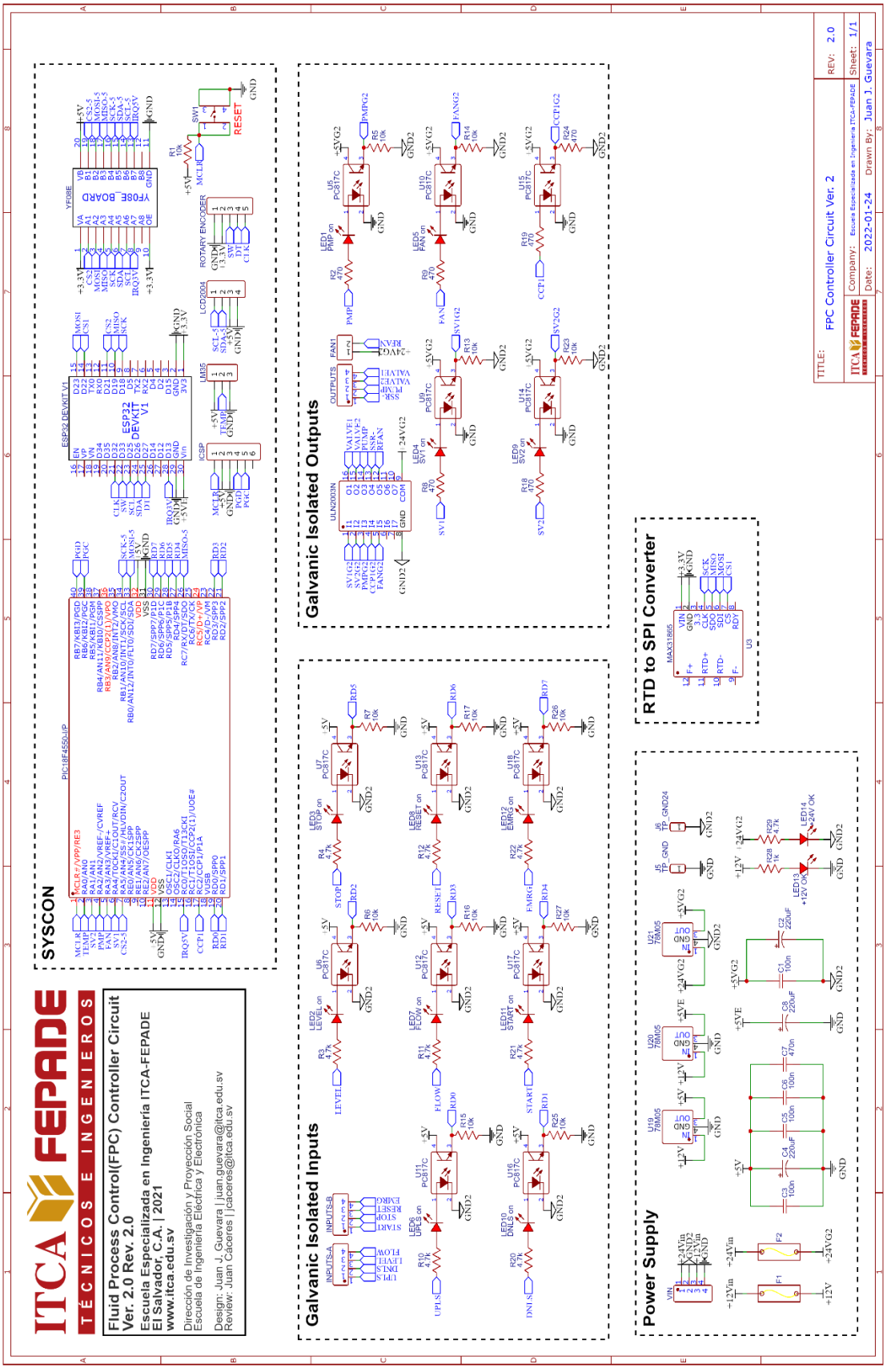
- 50: **SETPOINT**, actualización del valor del set point del controlador PID. Al recibir este comando el controlador esperará un segundo dato desde el servidor de aplicaciones que corresponde al valor del set point.
  Ej. 50\n <comando>
  35.5\n <valor>

- 51: **PROPORCIONAL**, actualización del valor de la variable del control proporcional del controlador PID. Al recibir este comando el controlador esperará un segundo dato desde el servidor de aplicaciones que corresponde al valor proporcional.
  Ej. 51\n <comando>
  2.0\n <valor>

- 52: **INTEGRAL**, actualización del valor de la variable del control integral del controlador PID. Al recibir este comando el controlador esperará un segundo dato desde el servidor de aplicaciones que corresponde al valor proporcional.
  Ej. 52\n <comando>
  0.015\n <valor>

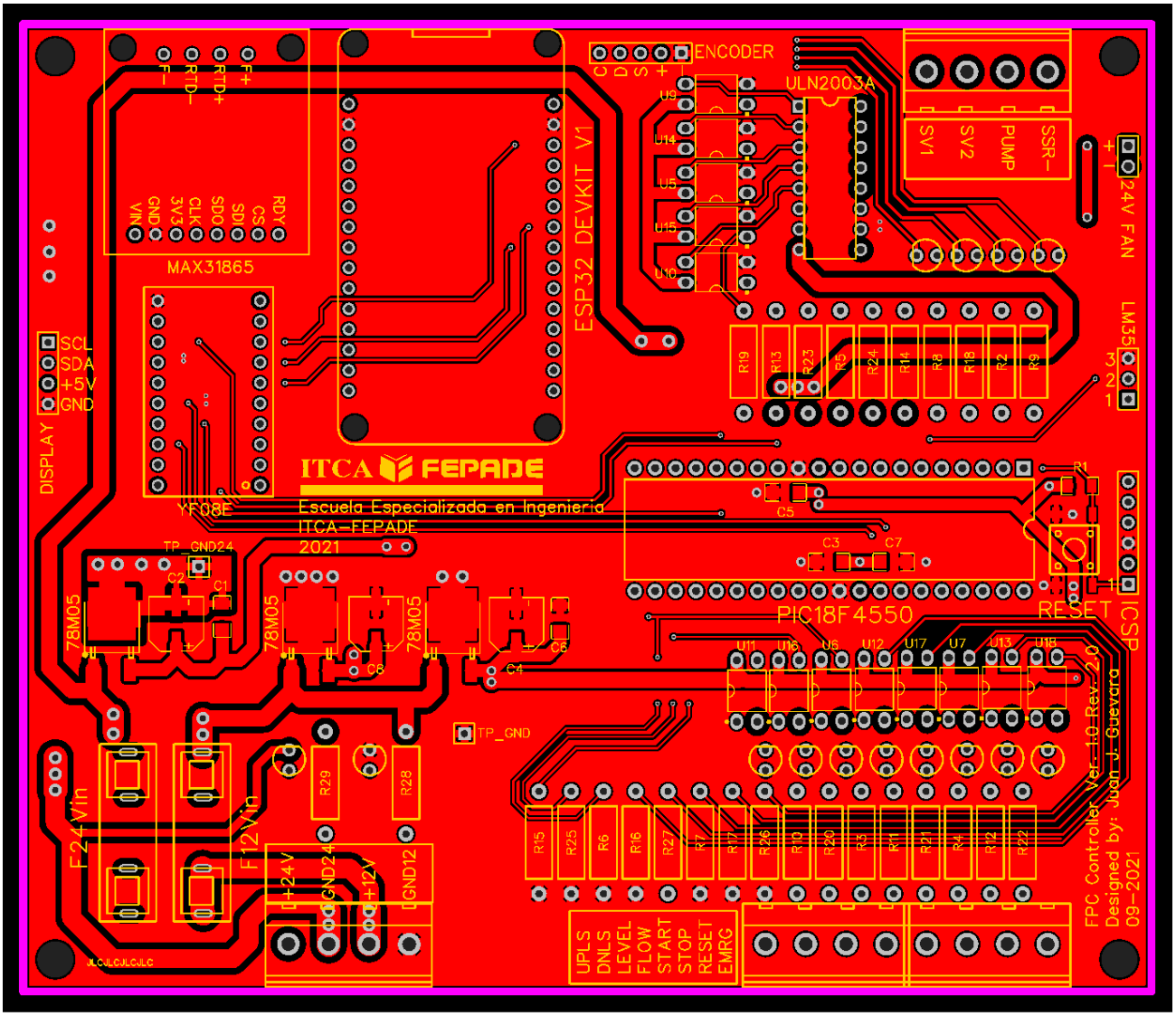
- 53: **DERIVATIVO**, actualización del valor de la variable del control derivativo del controlador PID. Al recibir este comando el controlador esperará un segundo dato desde el servidor de aplicaciones que corresponde al valor derivativo.
  Ej. 53\n <comando>
  0.001\n <valor>

12.3. ANEXO 3. DIAGRAMA ESQUEMÁTICO DEL CONTROLADOR ELECTRÓNICO

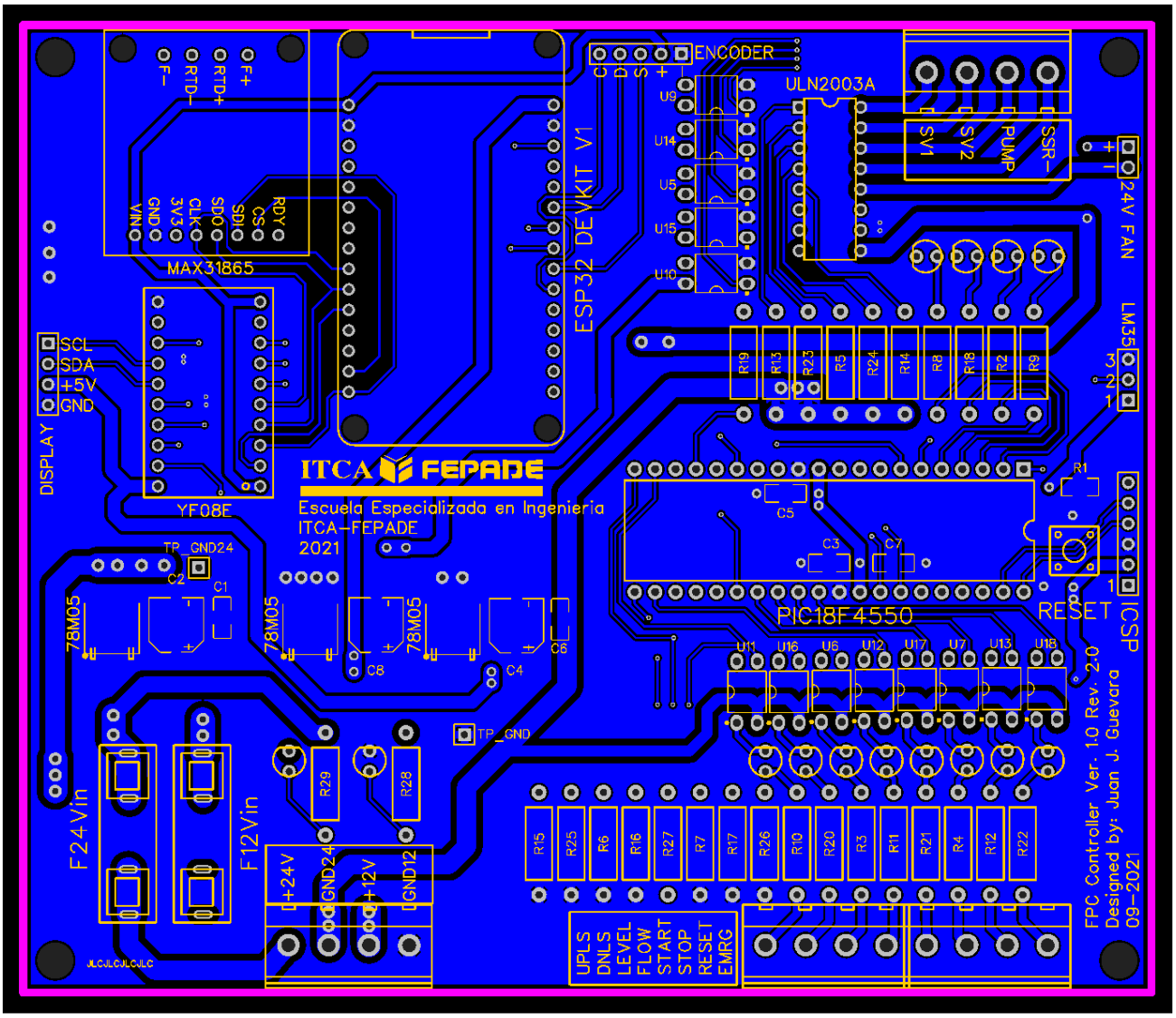


12.4. ANEXO 4. DISEÑO DEL PCB DEL CONTROLADOR ELECTRÓNICO

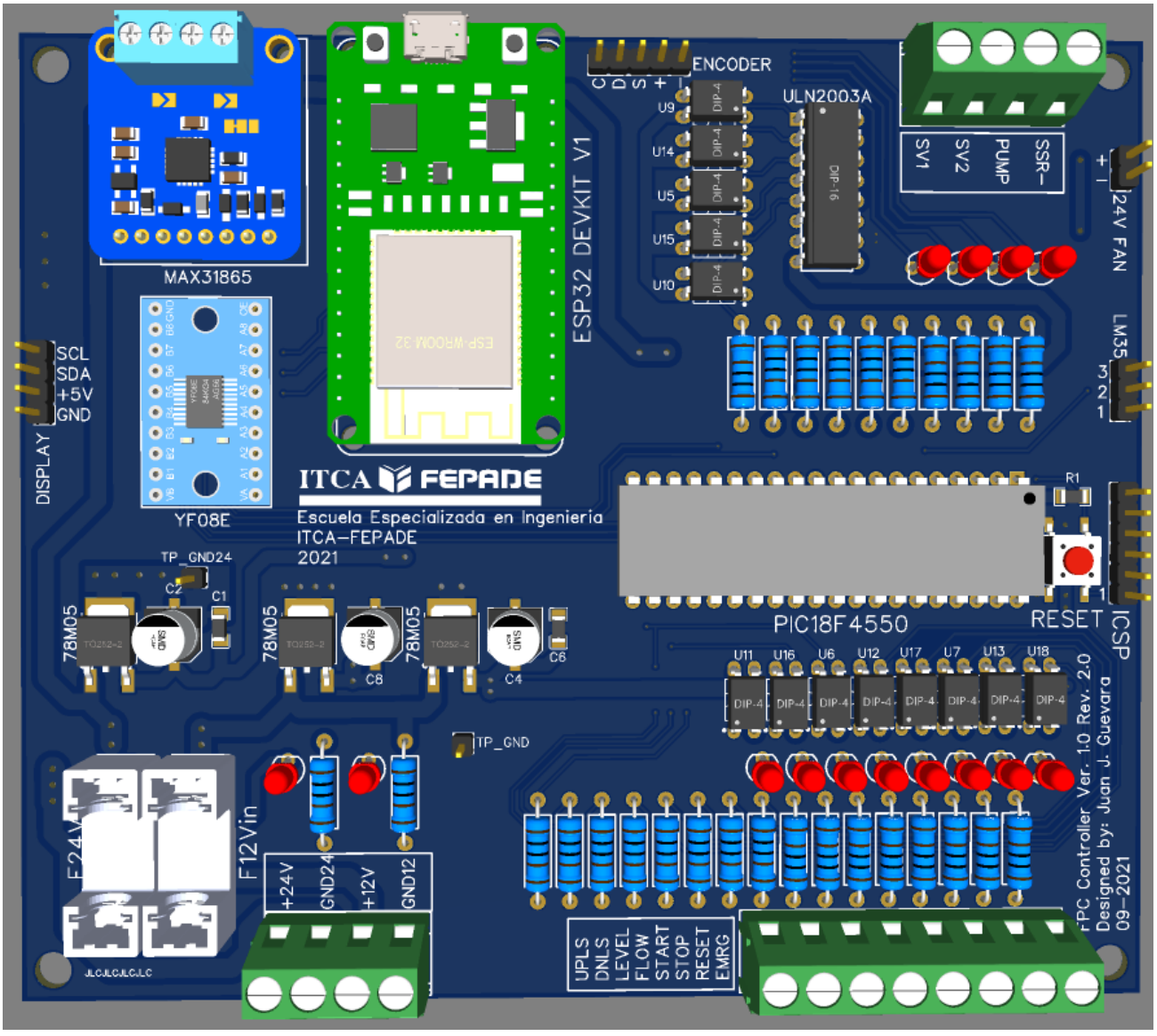
Capa TOP



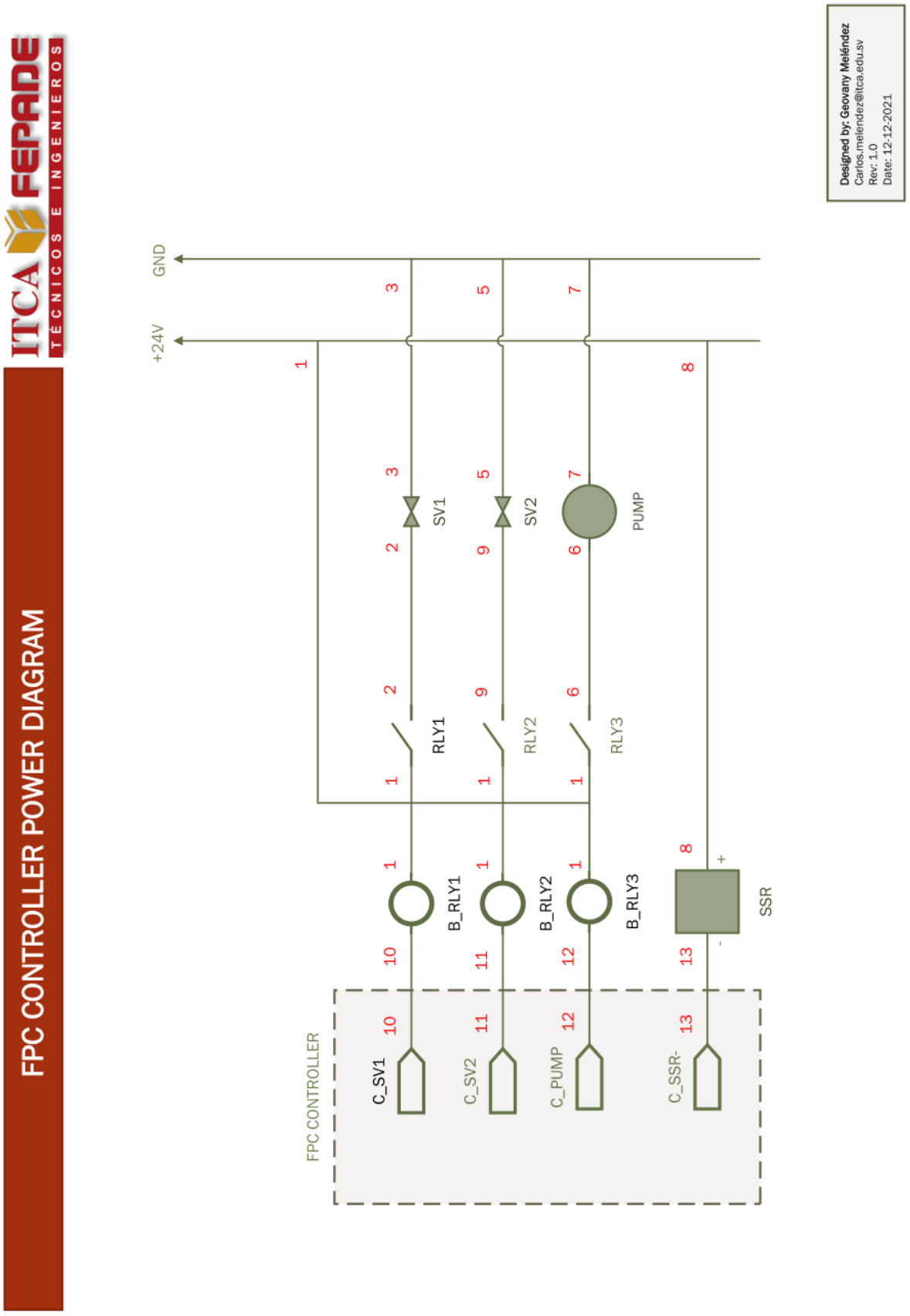
Capa Bottom


ENCODER
ULN2003A
ESP32 DEVKIT V1
MAX31865
YF08E
DISPLAY
SCL
SDA
+5V
GND
ITCA FEPADE
Escuela Especializada en Ingeniería
ITCA-FEPADE
2021
PIC18F4550
RESET
ICSP
24V FAN
LM35
78M05
F24Vin
F12Vin
+24V
GND24
+12V
GND12
UPLS
DNLS
LEVEL
FLOW
START
STOP
RESET
EMRG
FPC Controller Ver. 1.0 Rev. 2.0
Designed by: Juan J. Guevara
09-2021

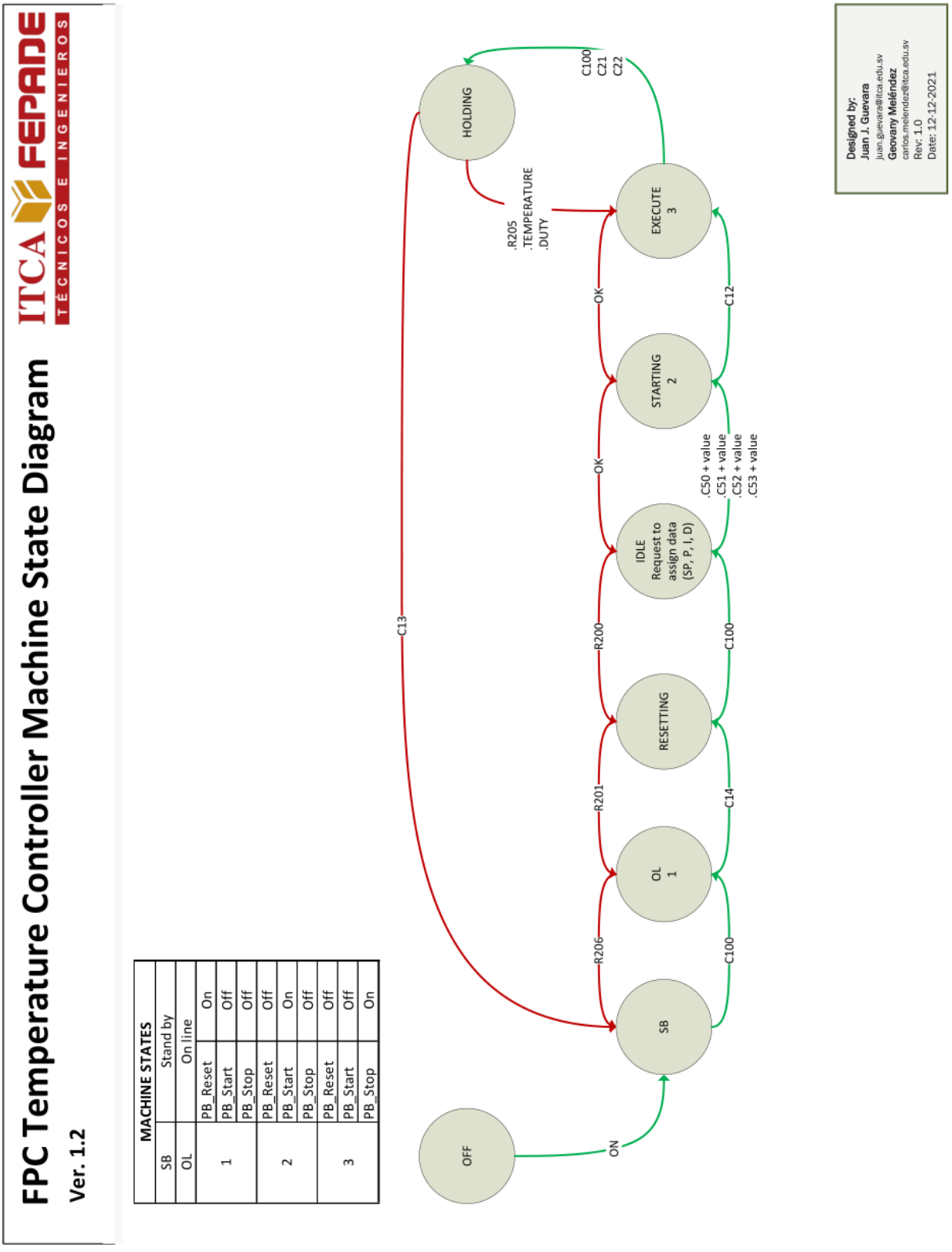
Vista 3D del diseño



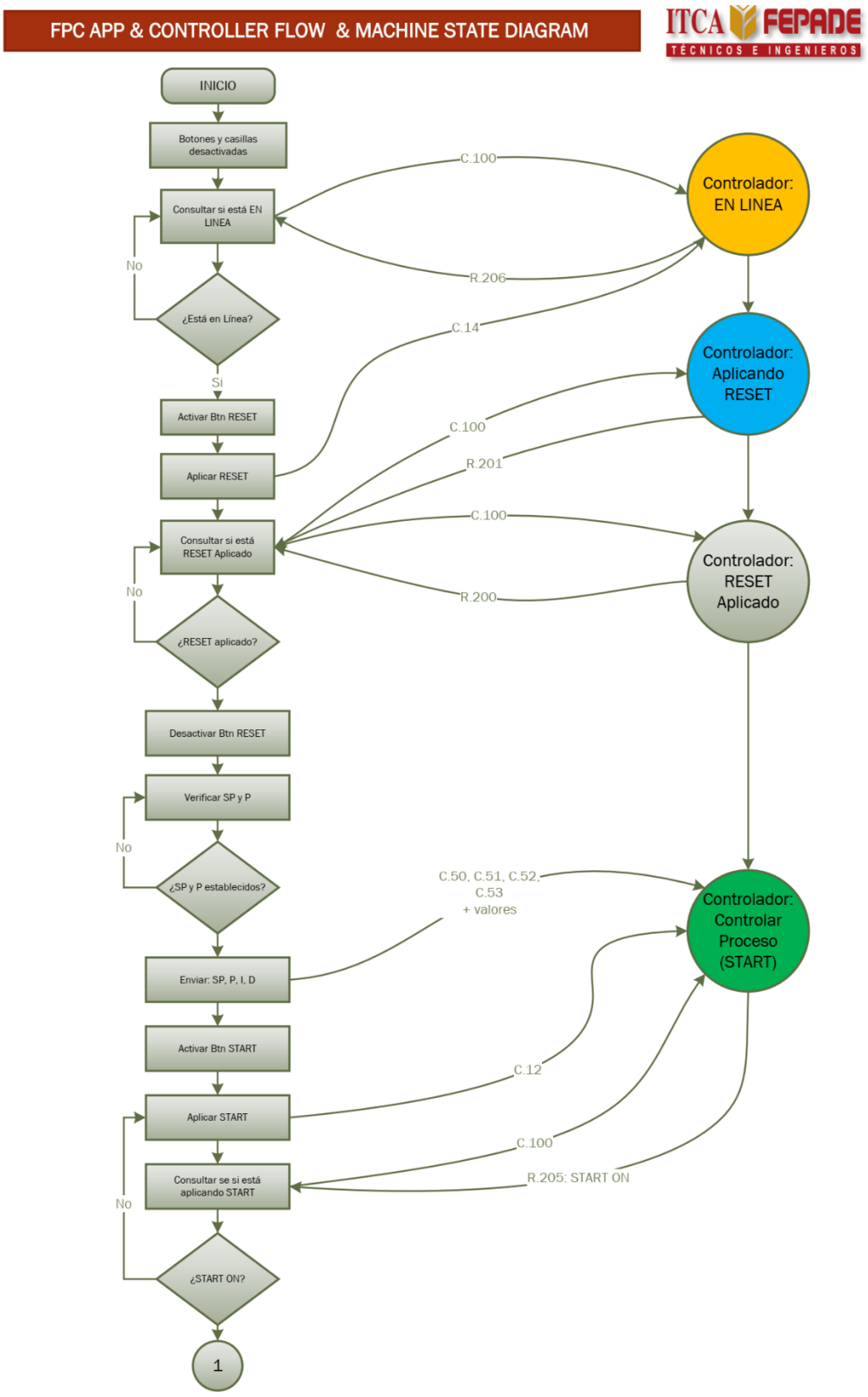
12.5. ANEXO 5. DIAGRAMA DE CONTROL Y POTENCIA DEL ENTRENADOR FPC

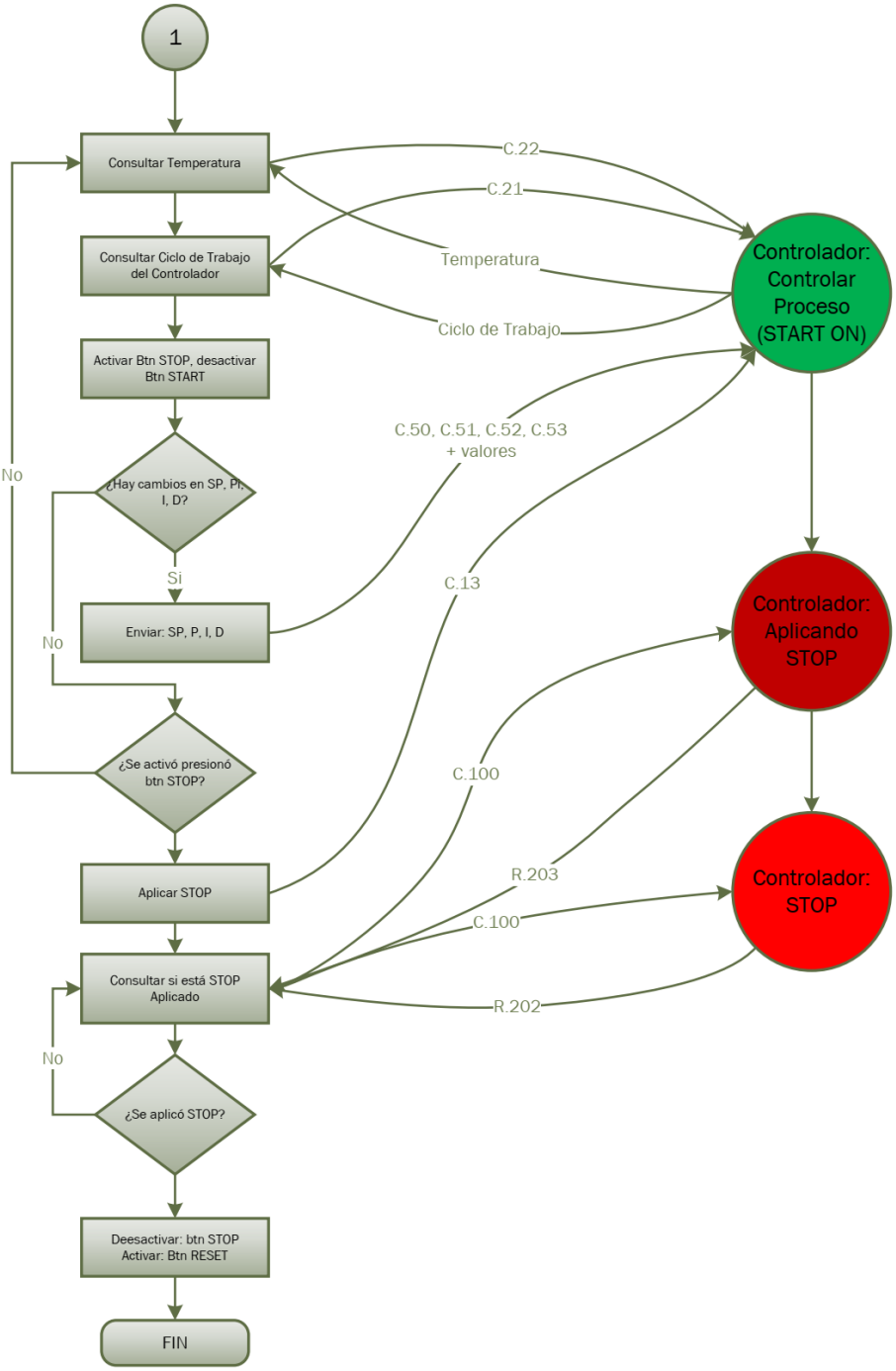


12.6. ANEXO 6. DIAGRAMA ESTADO DE MÁQUINA DEL CONTROLADOR (ENTRENADOR) FPC



12.7. ANEXO 7. DIAGRAMA FLUJO Y ESTADO DE LA APLICACIÓN Y CONTROLADOR ELECTRÓNICO





Designed by: Juan J. Guevara
juan.guevara@itca.edu.sv
Rev: 1.0
Date: 12-12-2021

12.8. ANEXO 8. GUÍA DE INSTALACIÓN DE LA APLICACIÓN WEB DE LA PLATAFORMA DE TELEINGENIERÍA

PROCESO DE DESARROLLO DE LA APLICACIÓN

La aplicación requiere de la instalación y configuración de los siguientes servicios:

12.9. REQUERIMIENTOS DEL SERVIDOR

Como parte fundamental para el correcto funcionamiento de la aplicación desarrollada, se establecieron los siguientes requerimientos mínimos que debe tener el servidor, el cual está montado en una Raspberry PI 4:

- Sistema Operativo: Raspberry Pi OS
- 4 GB de RAM.
- Procesador ARM Broadcom BCM2711 Quad Core Cortex A72 @ 1.5GHz.
- 16 GB de espacio libre en el disco duro.
- Medios de instalación (USB).
- Conexión a internet.
- Uso de la terminal del sistema operativo.
- Privilegios de administrador.

Requerimientos mínimos del cliente:

- Acceso a internet.
- PC de escritorio, laptop, dispositivo móvil o tableta.
- Navegador Web (Optimizado para Chrome o Mozilla).

12.10. CONFIGURACIÓN DEL SERVIDOR

El servidor por preparar es del tipo LAMP, es un acrónimo de **L**inux, **A**pache, **M**ySQL, **P**HP. Es una pila popular para crear y desplegar aplicaciones web dinámicas. En esta pila (stack), Linux sirve como el sistema operativo para la aplicación web. MySQL se utiliza como base de datos. Apache se utiliza como servidor web. PHP se utiliza para procesar contenido dinámico. En algunas otras variantes de esta pila, Perl se utiliza en lugar de PHP o Python. Sin embargo, para nuestro caso, vamos a instalar PHP. En otras pilas se suele utilizar MongoDB como base de datos.

Instalación de Apache.

Apache será el servicio que administrará la parte web del aplicativo en el servidor Raspberry Pi, para ello se detallan los pasos empleados para su instalación:

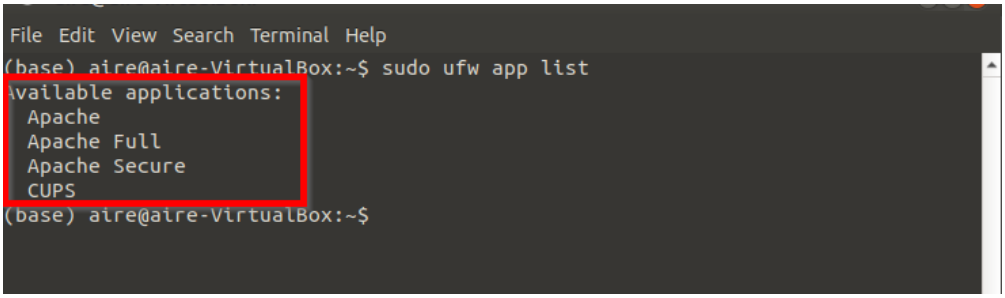
1. Abrir la terminal.

2. Actualizar el administrador de paquetes:
   ```
   sudo apt update
   ```

3. Instalar Apache2:
```
sudo apt install apache2
```

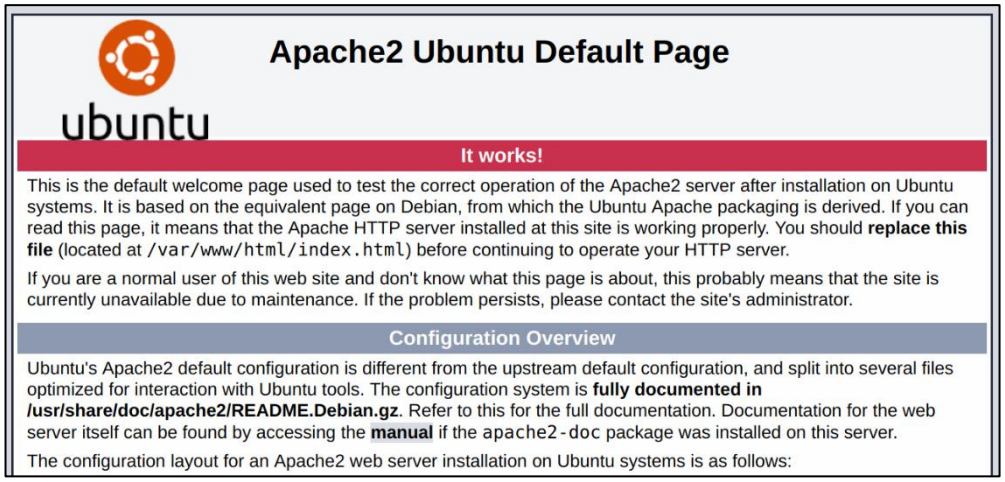
4. Mostrar las configuraciones disponibles:
```
sudo ufw app list
```

Verificar que se muestra una pantalla como esta:



5. Seleccionar “Apache Full” con el siguiente comando:
```
sudo ufw app info "Apache Full"
```

6. Al acceder desde un navegador ya sea local con la URL “http://localhost” o con la direccion IP del servidor verificar que se muestra la siguiente pantalla que indica que Apache ya está funcionando dentro de su sistema.



Instalación de PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación interpretado que se utiliza para la generación de páginas web de forma dinámica. Éste código se ejecuta al lado del servidor y se incrusta dentro del código HTML. Cabe destacar que es un lenguaje de código abierto, gratuito y multiplataforma.

1. Abrir la terminal.

2. Instalar las dependencias de PHP:
```
sudo apt install php libapache2-mod-php
```

3. En la mayoría de los casos, se desea modificar la forma mediante la cual Apache sirve archivos cuando un directorio es solicitado. En este momento, si un usuario solicita un directorio del servidor, Apache buscará, en primera instancia, un archivo llamado index.html. Se requiere que el servidor web relacione a los archivos PHP sobre cualquier otro archivo. Para lo cual se hace que el Apache busque el archivo index.php en primer lugar. Para llevar a cabo esto se debe ingresar el siguiente comando desde la terminal:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

4. Sustituir todo por las siguientes líneas:

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl
index.xhtml index.htm
</IfModule>
```

5. Cerrar la edición presionando Ctrl + X y luego 'Y' para guardar los cambios.

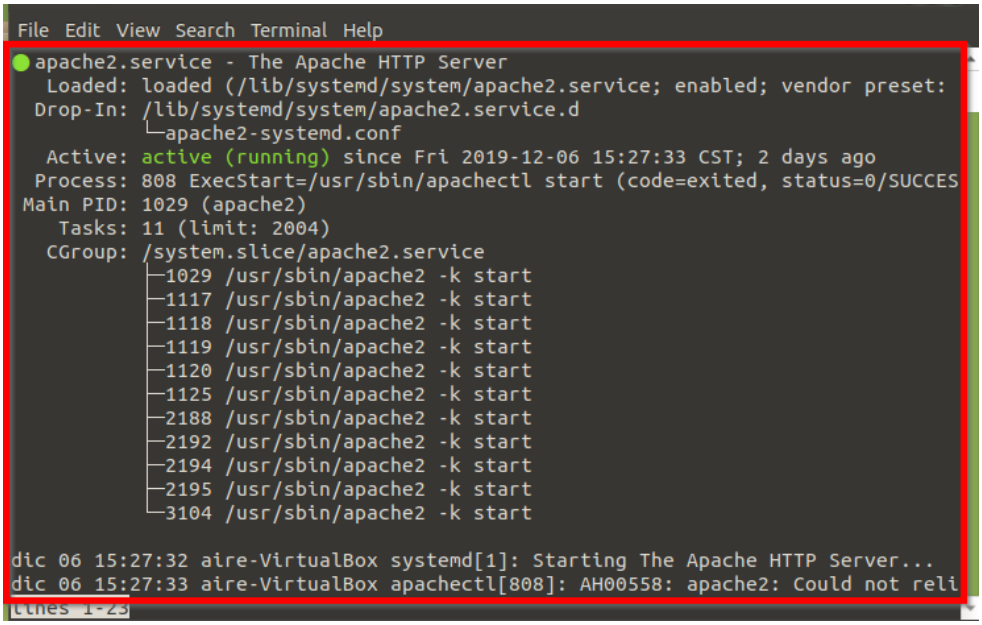
6. El siguiente paso es reiniciar el servidor Apache:

```
sudo systemctl restart apache2
```

7. Verificar el status para estar seguros de que el servidor se está ejecutando.

```
sudo systemctl status apache2
```

8. Mostrará algo como esto:



9. Instalar PHP-CLI:

```
sudo apt install php7.4-cli
```

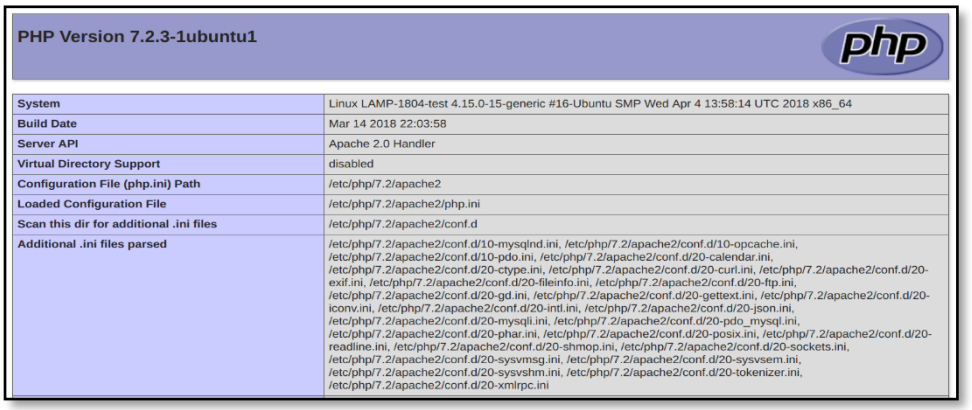
10. Verificación del funcionamiento de PHP dentro del servidor. Crear un nuevo archivo PHP con el siguiente comando:

```
sudo nano /var/www/html/info.php
```

11. Ingresar las siguientes líneas de código que nos mostrarán toda la información de PHP que instalamos previamente.

```
<?php
phpinfo();
?>
```

12. Digitar desde un navegador “http://IP-SERVIDOR/info.php” y nos debería de mostrar la siguiente información:



PHP Version 7.2.3-1ubuntu1

System	Linux LAMP-1804-test 4.15.0-15-generic #16-Ubuntu SMP Wed Apr 4 13:58:14 UTC 2018 x86_64
Build Date	Mar 14 2018 22:03:58
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-intl.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-xmlrpc.ini

Instalación de MySql

1. En la consola de comandos ingresar:

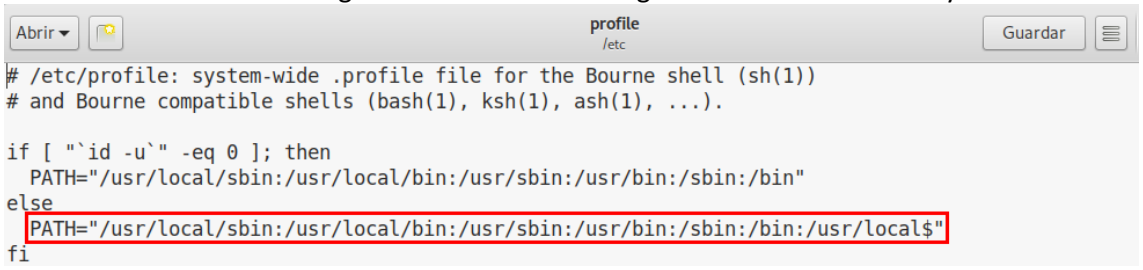
```
wget http://repo.mysql.com/mysql-apt-config_0.8.13-1_all.deb
```

Al momento en que realizas la instalación va a solicitar que introduzcas la contraseña para el usuario root, por lo que debes proporcionarla (no olvidar esta clave). De cualquier forma, puedes configurar tus credenciales de acceso con algunos comandos.

2. Editar el siguiente archivo:

```
gedit /etc/profile
```

3. Cambiar la línea marcada según se muestra en la imagen. Guardar los cambios y cerrar el archivo.



4. Ejecutar el siguiente comando:

```
root@servidor-web:/etc# dpkg -i mysql-apt-config_0.8.13-1_all.deb
```

5. Verificar que el servicio este activo tecleando.

```
service mariadb status
```

6. Si el servicio debe ser detenido teclear:

```
service mariadb stop
```

7. Si se desea iniciar de nuevo:

```
service mariadb start
```

8. Si se desea reiniciar:

```
service mysql restart
```

Instalación de Laravel

Laravel es uno de los frameworks de código abierto más fáciles de asimilar para PHP. Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

1. Asegurarse de haber iniciado sesión como usuario con privilegios de superusuario:
   ```
   sudo apt update && sudo apt upgrade
   ```

2. Instalar curl en el caso de que no se encuentre en nuestro Sistema:
   ```
   sudo apt install curl
   ```

3. Instalar composer con el siguiente comando:
   ```
   curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin -filename=composer
   ```

4. Verificar la versión de composer:
   ```
   composer –version
   ```

5. Debería de ver algo similar a esto:
   ```
   composer version 2.0.8
   ```

6. Crear el proyecto de laravel con el siguiente comando:
   ```
   composer create-project --prefer-dist laravel/laravel my_app
   ```

7. El comando anterior buscará todos los paquetes php necesarios. El proceso puede tardar unos minutos y, si tiene éxito, el final de la salida debería tener el siguiente aspecto:
   ```
   Package manifest generated successfully
   @php artisan key:generate –ansi
   Application key set successfully
   ```

8. Laravel está instalado en el sistema, el siguiente paso es abrir la carpeta del proyecto y ejecutar el servidor; para esto ingresar los siguientes comandos en la terminal:

```
cd my_app
php artisan server
```

9. La salida sería algo parecido a esto:

```
Laravel development server started: <http://127.0.0.1:8000>
```

10. Como último paso abrir el navegador e ingresa la siguiente URL y esperar a que cargue el proyecto en pantalla:

```
http://IP-SERVIDOR:8000
```

Instalación y configuración de Laravel con Apache (Producción)

Para llevar el proyecto de Laravel a producción con Apache2 es necesario realizar las siguientes configuraciones.

1. Actualizar las dependencias

```
sudo apt-get install update
sudo apt-get dist-upgrade
```

2. Mover la carpeta del proyecto a /var/www, que en este caso se encuentra en el escritorio:

```
sudo cp -r /home/aire/Desktop/invernadero /var/www/
```

3. Habilitar el modo rewrite de apache, para esto entramos a la terminal e ingresamos el siguiente comando:

```
sudo a2enmod rewrite
```

4. Cambiar configuracion del archivo de configuración por defecto de Apache

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

5. En este archivo cambiar las siguientes configuraciones:

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/invernadero/public

    <Directory /var/www/ invernadero/public>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride all
        Allow from all
        RewriteEngine On
        RewriteBase /var/www/ invernadero/public
    </Directory>
```

```
        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example, the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

6. Reiniciar el servidor apache:
   ```
   sudo service apache2 restart
   ```

7. Asignar los permisos a la carpeta storage:
   ```
   sudo chown -R www-data: storage
   sudo chown -R 775 storage
   ```

8. Cambiar configuración del archivo. env, que se encuentra en la ruta */var/www/electrica*:
   ```
   sudo nano .env
   ```

9. Cambiar las siguientes configuraciones:
   ```
   APP_ENV = production
   APP_DEBUG = false
   ```

10. Copiar el archivo .htaccess e index.php a la carpeta raíz del proyecto:
    ```
    cd /var/www/electrica/public
    cp .htaccess /var/www/electrica
    cp index.php /var/www/electrica
    ```

11. Modificar el archivo .htaccess alojado en */var/www/*electrica
    ```
    sudo nano .htacces
    ```

12. Eliminar todo el contenido y agregar las siguientes líneas:
    ```
    <IfModule mod_rewrite.c>
        RewriteEngine on
        RewriteRule ^(.*)$ public/$1 [L]
    </IfModule>
    ```

13. Renombrar el archivo server.php a index.php
    ```
    sudo mv server.php index.php
    ```

14. Modificar .htacces en la carpeta public:
    ```
    <IfModule mod_rewrite.c>
    ```

```
    <IfModule mod_negotiation.c>
        Options -MultiViews
    </IfModule>

    RewriteEngine On

    # Redirect Trailing Slashes...
    RewriteRule ^(.*)/$ /$1 [L,R=301]

    # Handle Front Controller...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>
```

15. Ingresar a la URL del proyecto para verificar que se ejecuta correctamente.

    NOTA: Ejecutar las siguientes configuraciones ÚNICAMENTE si al ingresar a "localhost" le muestra el pseudocódigo del index.php:

16. Deshabilitar mpm event
    ```
    a2dismod mpm_event
    ```

17. Habilitar php7.4
    ```
    a2enmod php7.4
    ```

18. Reiniciar apache2
    ```
    systemctl restart apache2
    ```

19. Verificar que el proyecto esté funcionando correctamente por medio de un navegador digitando la url del sistema.

Instalación de NodeJS

1. Utilizar la consola y escribir el siguiente comando:
   ```
   curl -sL
   https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.s
   h -o install_nvm.sh
   ```

2. Ejecute la secuencia de comandos con bash:
   ```
   bash install_nvm.sh
   ```

3. Instalará el software en un subdirectorio de su directorio de inicio en ~/.nvm. También agregará las líneas necesarias a su archivo ~/.profile para utilizarlo. Para obtener acceso a la funcionalidad nvm, deberá cerrar sesión e iniciarla de nuevo u obtener el archivo ~/.profile para que su sesión actual registre los cambios:
   ```
   source ~/.profile
   ```

4. Con nvm instalado, puede instalar versiones aisladas de Node.js. Para obtener información sobre las versiones de Node.js disponibles, escriba lo siguiente:
   ```
   nvm ls-remote
   ```

5. La versión LTS actual en el momento en que se redactó este artículo es la 8.11.1. Puede instalarla escribiendo lo siguiente:
   ```
   nvm install 13.14.0
   ```

6. Normalmente, nvm aplicará un cambio para utilizar la versión más reciente instalada. Puede indicar a nvm que utilice la versión que acaba de descargar escribiendo lo siguiente:
   ```
   nvm use 13.14.0
   ```

7. Cuando instale Node.js utilizando nvm, el ejecutable se llamará node. Puede ver la versión que el shell utiliza actualmente escribiendo lo siguiente:
   ```
   node –v
   ```

8. Si dispone de varias versiones de Node.js, es posible ver cuál está instalada escribiendo lo siguiente:
   ```
   nvm ls
   ```

9. Establecer como predeterminada una de las versiones, escribir lo siguiente:
   ```
   nvm alias default 8.11.1
   ```

10. Esta versión se seleccionará de forma automática cuando se genere una nueva sesión. También es posible hacer referencia a ella con el alias, como se muestra:
    ```
    nvm use default
    ```

Instalación de PM2.

1. Correr el siguiente comando para instalar el gestor de procesos PM2 de forma global:
   ```
   npm -g install pm2
   ```

2. Iniciar PM2 al iniciar el servidor.
   ```
   pm2 startup systemd
   ```

3. Configuraciones de PM2.
   ```
   systemctl start pm2-root
   systemctl enabled pm2-root
   ```

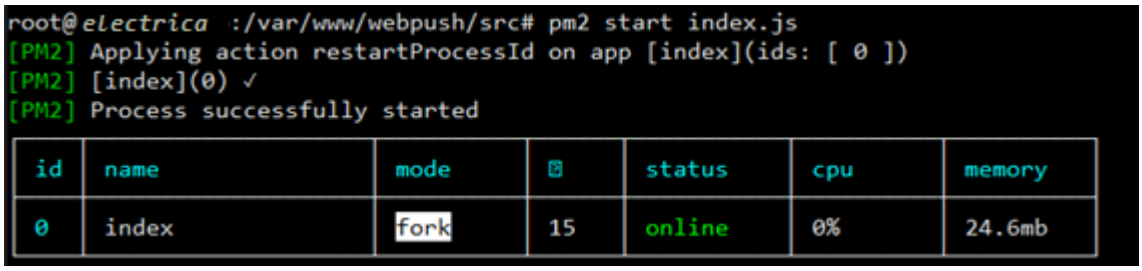
Instalación del servidor de notificaciones push.

1. Clonar el proyecto del repositorio.
   ```
   https://github.com/itcafepade/webpush
   ```

2. Instalar las dependencias del proyecto.
   ```
   npm install
   ```

3. Iniciar el proceso con PM2:
   ```
   pm2 start src/index.js
   ```

4. El servicio se encuentra activado y funcionando correctamente.



ITCA FEPADE
TÉCNICOS E INGENIEROS

SEDE CENTRAL Y CENTROS REGIONALES EL SALVADOR

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

1. SEDE CENTRAL SANTA TECLA
Km. 11.5 carretera a Santa Tecla, La libertad.
Tel.: (503) 2132-7400

2. CENTRO REGIONAL SANTA ANA
Final 10a. Av. Sur, Finca Procavia.
Tel.: (503) 2440-4348

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

4. CENTRO REGIONAL SAN MIGUEL
Km. 140 carretera a Santa Rosa de Lima.
Tel.: (503) 2669-2298

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

www.itca.edu.sv