

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

NOMBRE DEL PROYECTO:

WEB SERVER DE PARO EXTERNOS DE CLEVER

OPCIÓN QUE CORRESPONDA:

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PRESENTA:

EDNA AISLINN TORRES PÉREZ

NOMBRE DEL ASESOR:

LIC. ELIUT FLORES JIMÉNEZ

APIZACO, TLAX. (octubre de 2025)



Apizaco, Tlaxcala, **18/septiembre/2025**

Oficio No. S.C./OPV/0525/25

**ASUNTO: Liberación de proyecto
para titulación integral**

CECILIA YARELLY ARREGIN LARA
JEFA DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL.

NOMBRE DEL EGRESADO:	TORRES PÉREZ EDNA AISLINN
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	20371003
NOMBRE DEL PROYECTO:	Web Server de paro externos de Clever
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

ATENTAMENTE

*Excelencia en Educación Tecnológica.
Pensar para Servir, Servir para Triunfar®*

MIQUELINA SÁNCHEZ PULIDO
JEFA DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

ccp. Coordinación de carrera
Expediente
MSP/mrol*



ELIUT JIMÉNEZ FLORES

ASESOR INTERNO



2025
Año de
**La Mujer
Indígena**

Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec,
Municipio de Tzompantepec, Tlaxcala, Mex. C.P. 90491
Tel. (241) 4172010 Ext. 115, e-mail: sistemas@apizaco.tecnm.mx
tecnm.mx | apizaco.tecnm.mx





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Apizaco
División de Estudios Profesionales

Tzompantepec, Tlaxcala **21/octubre/2025**

TORRES PÉREZ EDNA AISLINN
EGRESADO(A) DE LA CARRERA DE INGENIERÍA
EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
CON NÚMERO DE CONTROL 20371003.
P R E S E N T E.

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** y cuyo tema es: **WEB SERVER DE PARO EXTERNOS DE CLEVER**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **1 USB** del mismo, apropiadamente empastado, a la brevedad, para presentar su Acto de Recepción Profesional.

Sin otro particular por el momento me despido.

ATENTAMENTE
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

CECILIA YARELY ARREGUÍN LARA
JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

ccp. Archivo

CYAL/rrc



2025
Año de
La Mujer
Indígena

Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec,
Municipio de Tzompantepec, Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010
Ext. 304, e-mail: dep_titulacion@apizaco.tecnm.mx | apizaco.tecnm.mx



Agradecimientos

Primero que nada, agradezco profundamente a mis padres Yoana Pérez Mejía y Miguel Torres Rivera, por su amor incondicional, su apoyo constante y por enseñarme el valor del esfuerzo, la responsabilidad y la perseverancia. Gracias por siempre creer en mí, este logro también es de ustedes.

Con especial cariño agradezco a mis abuelos, Bernardo Pérez George, Elvia Mejía Escobar, Alejandro Torres Arévalo y Teresa Rivera Cabrera por sus palabras sabias, su ejemplo de vida y su amor que ha sido siempre un refugio y una fuente de motivación para seguir adelante, su apoyo ha sido fundamental en este camino.

Agradezco también a Motherson, por brindarme la oportunidad de realizar mi residencia profesional en sus instalaciones, en especial a mantenimiento y a todo el equipo de la Línea 1 de producción, por permitirme ser parte de su entorno de trabajo, compartir sus conocimientos y mostrar siempre una disposición para colaborar.

A mi asesor externo el Ing. José Eduardo Pérez Muñoz con el apoyo del Ing. Kevin Robles y Jaciel Barajas por su guía, su tiempo y sus enseñanzas durante el desarrollo del proyecto. Su experiencia ha sido fundamental para que este trabajo alcanzara los objetivos planteados.

A mi asesor interno el Lic. Eliut Flores Jiménez, por su acompañamiento y orientación durante este proceso, por sus comentarios constructivos y por motivarme a dar siempre lo mejor.

Este proyecto no solo representa el cierre de una etapa académica, sino también un paso firme hacia los sueños que he venido construyendo con esfuerzo, dedicación y el apoyo de quienes más quiero.

Índice

Agradecimientos	1
índice de Ilustraciones	5
Contenido	8
Introducción	8
Generalidades	9
Ubicación.....	9
Logotipo de la empresa	9
SAS Automotive Systems, S.A. de C.V.....	10
Misión	10
Visión	10
Objetivo	11
Valores	11
Problemas a resolver.....	11
Objetivos	13
Objetivo General.....	13
Objetivos Específicos	13
Justificación.....	14
Marco teórico	14
1. Manufactura y Producción en la Industria Automotriz	14
1.1. Conceptos de Producción y Ensamblaje en Línea	15
1.2. Lean Manufacturing y la Reducción de Desperdicios	15
1.3. OEE (Overall Equipment Effectiveness) y su Impacto en la Disponibilidad	16
2. Automatización y Control Industrial	17
2.1. Controladores Lógicos Programables (PLC)	17
2.2. Sensores y Actuadores: La Interacción en el Control Industrial	18
3. La Industria 4.0 y la Conectividad	19
3.1. Fundamentos de la Industria 4.0: Hacia los Sistemas Ciberfísicos.....	19

3.2. Internet de las Cosas Industrial (IIoT) y la Conectividad de la Planta	20
4. Bases de Datos y Gestión de la Información	21
4.1. Sistemas de Gestión de Bases de Datos (SGBD) para la Centralización de Datos	22
4.2. Mapeo Objeto-Relacional (ORM) con SQLAlchemy	22
5. Desarrollo Web para Aplicaciones de Monitoreo.....	23
5.1. Arquitectura Cliente-Servidor y el Monitoreo Remoto	24
5.2. Tecnologías Front-End: La Interfaz de Usuario	25
5.3. Tecnologías Back-End: La Lógica del Servidor	25
6. Protocolos de Comunicación Industrial.....	26
6.1. Comunicación Ethernet/IP para la Interacción en Tiempo Real	27
6.2. OPC UA (Open Platform Communications Unified Architecture) para la Interoperabilidad	28
7. Usabilidad y Diseño de Interfaces de Usuario (UI/UX).....	28
7.1. Diseño de Interfaz de Usuario (UI) para la Usabilidad en el Entorno Laboral.....	29
7.2. Experiencia de Usuario (UX) y el Impacto en la Eficiencia Operativa	30
8. Mejora Continua y Toma de Decisiones Basada en Datos.....	31
8.1. Métodos de Análisis de Causa Raíz (RCA)	31
8.2. Business Intelligence (BI) y Visualización de Datos para la Acción	32
Procedimiento y descripción de las actividades realizadas	33
Análisis de la situación actual	33
Definición de requerimientos del sistema	35
Arquitectura del sistema	36
Instalación y configuración de Siemens TIA Portal V19	38
Instalación y configuración de Python.....	42
Instalación y configuración de Visual Studio Code.....	43
Organización del proyecto en Visual Studio Code	45
Conexión del Web Server con el PLC mediante Snap7	46
Explicación del archivo plc.py	46

Configuración de estaciones y definición de paros en el sistema	50
Procedimiento para agregar una estación	51
Verificación y corrección de paros internos y externos en TIA Portal	53
Arquitectura y funcionamiento del archivo principal app.py.....	54
Configuración inicial del archivo app.py	55
Definición de rutas principales en app.py	58
Funcionamiento de la Base de Datos en el Sistema de Monitoreo	62
Definición y establecimiento de causas en resumen.html	64
Creación del análisis de paros por área.....	66
Descarga de reportes en Excel.....	69
Resultados	72
Conclusiones	80
Conclusiones del proyecto.....	80
Recomendaciones	81
Experiencia personal y profesional adquirida.....	82
Competencias desarrolladas	83
Fuentes de información	84

índice de Ilustraciones

Ilustración 1.- Ubicación	9
Ilustración 2.- Logotipo Motherson	9
Ilustración 3.- Arquitectura cliente-servidor	24
Ilustración 4.- Línea 1 de producción	34
Ilustración 5.- HMI de paros.....	35
Ilustración 6.- Diagrama de arquitectura	38
Ilustración 7.- Logo TIA Portal V19.....	38
Ilustración 8.- Descarga de TIA Portal v19.....	39
Ilustración 9.- Carpeta de TIA Portal v19.....	39
Ilustración 10.- Antivirus y firewalls desactivados.....	39
Ilustración 11.- Privilegios de administrador para TIA Portal v19	40
Ilustración 12.- Opción “typical” en TIA Portal V19.....	40
Ilustración 13.- Instalación final de TIA Portal v19.....	40
Ilustración 14.- Carga del proyecto de línea 1	41
Ilustración 15.- Conexión exitosa del PLC.....	41
Ilustración 16.- Logo de python.....	42
Ilustración 17.- Descarga de python.....	42
Ilustración 18.- Instalar PATH.....	42
Ilustración 19.- Instalación exitosa de Python.....	42
Ilustración 20.- Logo Visual Studio Code	43
Ilustración 21.- Descarga de Visual Studio Code	43
Ilustración 22.- Acuerdo de licencia de Visual Studio Code.....	43
Ilustración 23.- Tareas adicionales de Visual Studio Code.....	44
Ilustración 24.- Sección de Extensiones (Python)	44
Ilustración 25.- Intérprete Python.....	44
Ilustración 26.- Estructura de archivos y carpetas	45
Ilustración 27.- Conexión mediante Snap7	46
Ilustración 28.- Establecer conexión con el PLC	47
Ilustración 29.- Mantener la conexión activa automáticamente con el PLC.....	47
Ilustración 30.- Mantener la conexión activa automáticamente con el PLC.....	47
Ilustración 31.- Lectura de valores booleanos	48
Ilustración 32.- Escritura de valores booleanos	48
Ilustración 33.- Control seguro de la conexión.....	49
Ilustración 34.- Diagrama de como interactúan los métodos	49
Ilustración 35.- Paros Internos	50
Ilustración 36.- Paros Externos	50
Ilustración 37.- Estaciones con reset	51

Ilustración 38.- Estaciones sin reset.....	51
Ilustración 39.- Ejemplo de paro con reset	52
Ilustración 40.- Ejemplo de paro sin reset.....	52
Ilustración 41.- Lista de paros en TIA Portal	53
Ilustración 42.- Lista de reset en TIA Portal	53
Ilustración 43.- Importación de librerías.....	55
Ilustración 44.- Configuración de logging	56
Ilustración 45.- Inicialización de la aplicación Flask	56
Ilustración 46.- Configuración de la base de datos con SQLAlchemy.....	56
Ilustración 47.- Definición de la zona horaria	56
Ilustración 48.- Definición de los modelos de datos (Paro)	57
Ilustración 49.- Definición de los modelos de datos (turno).....	57
Ilustración 50.- Creación de tablas y carga inicial	57
Ilustración 51.- Conexión con el PLC.....	58
Ilustración 52.- Ruta página principal.....	58
Ilustración 53.- Ruta estado de estaciones.....	59
Ilustración 54.- Ruta reset de estaciones	59
Ilustración 55.- Ruta resumen de paros.....	60
Ilustración 56.- Ruta guardar causa de paro	60
Ilustración 57.- Ruta página de análisis de datos	61
Ilustración 58.- Ruta Análisis de tiempo.....	61
Ilustración 59.- Ruta exportar Excel.....	62
Ilustración 60.- Creación de registros.....	63
Ilustración 61.- Actualización de registros	63
Ilustración 62.- Asignación de causa y área	64
Ilustración 63.- Causas para paros externos.....	65
Ilustración 64.- Causas para paros externos 2	65
Ilustración 65.- Causas paros internos.....	66
Ilustración 66.- Causas paros de ECHECK.....	66
Ilustración 67.- Obtención de los datos desde el backend	67
Ilustración 68.- Creación de los gráficos con Chart.js	67
Ilustración 69.- Gráfico de paros externos por área	68
Ilustración 70.- Top 3 causas por área seleccionada	68
Ilustración 71.- Inicialización de los gráficos	69
Ilustración 72.- Ruta definida en el servidor Flask.....	70
Ilustración 73.- Historial de paros	70
Ilustración 74.- Resumen total de paros.....	70
Ilustración 75.- Crear el archivo Excel en memoria	71

Ilustración 76.- Nombre del archivo Excel	71
Ilustración 77.- Interfaz principal de los paros externos (Andon).....	73
Ilustración 78.- Interfaz principal de los paros internos (Andon)	74
Ilustración 79.- Resumen de paros externos.....	74
Ilustración 80.- Resumen de paros internos	75
Ilustración 81.- Menú de causas externos empezando por área	75
Ilustración 82.- Causa completa en paro externo	75
Ilustración 83.- Causas ECHECK	76
Ilustración 84.- Causas paros internos.....	76
Ilustración 85.- Paro externo guardado.....	76
Ilustración 86.- Paro interno guardado	76
Ilustración 87.- análisis de paros externos	77
Ilustración 88.- Top 3 causas por Área.....	77
Ilustración 89.- análisis de paros internos	78
Ilustración 90.- Top 3 estaciones por internos (causas).....	78
Ilustración 91.- análisis paros ECHECK.....	79
Ilustración 92.- Descarga de reporte por turno en Excel	79
Ilustración 93.- Reporte de Excel	80

Contenido

Introducción

El presente proyecto de residencia profesional se desarrolló en la empresa SAS Automotive Systems, S.A. de C.V, su planta ubicada en Puebla, México, se dedica principalmente al ensamblaje de cockpits (tableros), colaborando estrechamente con marcas como Volkswagen.

Se trabajo en la Línea 1 de producción, en ella se ensamblan los modelos Jetta y Tayron, la cual cuenta con 36 estaciones de trabajo, así como un área de ECHECK y una sección de Final Inspection (FI). Dentro de su operación se presentan paros de línea, los cuales se clasifican en paros externos (generados mediante el sistema Andon, al jalar una cuerda instalada en las estaciones) y paros internos (registrados mediante el sistema Clever). En total, la línea tiene 32 paros externos y 14 internos, lo que representa un reto significativo en términos de control, registro y análisis de estos eventos.

Previo al desarrollo del proyecto, el registro de los paros se realizaba de forma manual y poco precisa, ya que los supervisores anotaban la información con base en su memoria o sin un seguimiento en tiempo real, lo que dificultaba la identificación de causas y patrones. Ante esta problemática, surgió la necesidad de diseñar e implementar un sistema automatizado de monitoreo en tiempo real, que permitiera llevar un control eficiente sobre las causas, duración e impacto de los paros, apoyando así la toma de decisiones operativas y la mejora continua.

El sistema desarrollado integra tecnologías como el PLC Siemens S7-1500, una interfaz web y una base de datos. El sistema se encarga automáticamente de registrar el inicio, fin y duración real del evento, permitiendo generar estadísticas por turno, área, estación y tipo de paro.

Este proyecto no solo representa un avance en la digitalización y control de procesos en la Línea 1, sino que también contribuye a fortalecer la cultura de mejora continua, aumentar la eficiencia productiva y atender problemáticas relacionadas con la ergonomía laboral y la gestión de incidencias.

Generalidades

Ubicación

Av. San Lorenzo Almecatla, Zona VW, Sanctorem, Pue.

Ubicado en: Parque Industria FINSA Puebla

CP: 72730

Tel: 222 764 8376

Web: careers.motherson.com/es

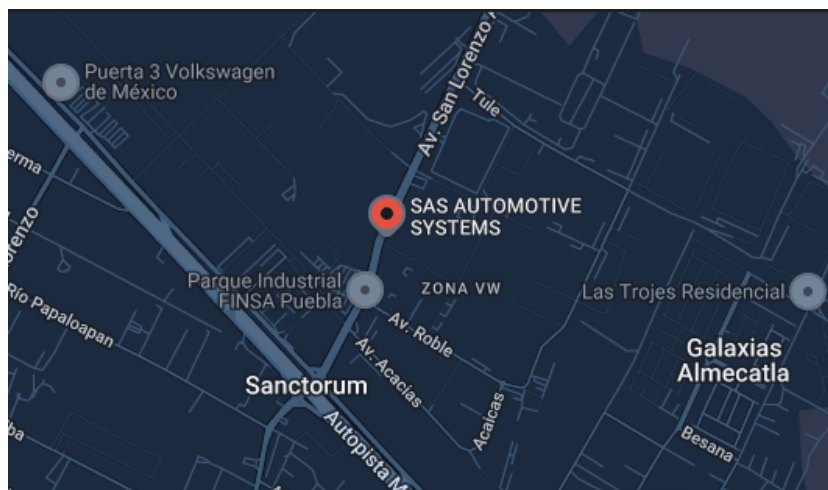


Ilustración 1.- Ubicación

Logotipo de la empresa



Ilustración 2.- Logotipo Motherson

SAS Automotive Systems, S.A. de C.V

Motherson SAS, es uno de los líderes globales en manufactura de componentes y sistemas para la industria automotriz, es especializada en el diseño, desarrollo y producción de soluciones innovadoras que impulsan en el futuro de la movilidad. Con más de cuatro décadas de experiencia, nuestra empresa combina ingeniería de vanguardia, tecnología avanzada y un enfoque centrado en la sostenibilidad para ofrecer componentes críticos, módulos inteligentes y sistemas integrados que mejoran la seguridad, la conectividad y la eficiencia de los vehículos.

SAS Automotive Systems fue fundada en 1996 con un join venture (empresa conjunta) entre Faurecia – Continental.

2004 – Inicio de operaciones de planta Puebla.

2019 – Adquisición total por Faurecia de SAS interior modules.

2022 – SAS Interior modules pasa a formar parte del grupo Forvia: Faurecia-Hella-SAS.

2023 – Motherson compra SAS interior modules.

Misión

Garantizar que encantamos al cliente.

Implicar a los empleados como <<socios>> en el progreso.

Mejorar el valor a largo plazo para los accionistas.

Establecer nuevas normas de buena ciudadanía corporativa.

Preservar el planeta y aspirar a mejorar el impacto ambiental de todo lo que hacemos.

Visión

Ser el proveedor de soluciones sostenibles preferido a nivel mundial.

Objetivo

Juntos aspiramos a encantar a todo aquel que deposita su confianza en nosotros y persigue objetivos aparentemente imposibles, de forma que ofrezcamos oportunidades sostenibles a nuestros socios y estemos orgullosos de formar parte de algo más grande que nosotros mismos.

Valores

- Ser una organización ágil, reactiva y que aprende.
- Mejorar continuamente para lograr normas de primer nivel y la satisfacción total de los clientes.
- Gestionar proactivamente el cambio y adaptarse a él.
- Mantener normas estrictas de ética, integridad y seguridad empresarial.
- Garantizar una cultura y un comportamiento comunes y alimentar el orgullo en toda nuestra organización.
- Reconocer las contribuciones de las personas y beneficios de nuestra diversidad.
- Garantizar el bienestar, la igualdad y las oportunidades para todos.
- Desarrollar competencias de liderazgo más sólidas y un mayor trabajo global en equipo.
- Aumentar constantemente los niveles de competencia en nuestra organización mediante la colaboración y los programas de intercambio de conocimientos.

Problemas a resolver

Durante el desarrollo del proyecto en la Línea 1 de producción de SAS Automotive Systems, S.A. de C.V., se identificaron diversas problemáticas que afectaban directamente la eficiencia operativa, el control de los paros y la calidad de los datos utilizados para la toma de decisiones. A continuación, se describen los principales problemas detectados, priorizados según su impacto en el desempeño de la línea:

Falta de control y registro preciso de las causas de paro: el problema más crítico identificado fue la ausencia de un sistema que permitiera registrar de forma estructurada y en tiempo real las causas específicas de los paros. Esto limitaba gravemente la capacidad de análisis, dificultaba la detección de patrones recurrentes y entorpecía la aplicación de acciones correctivas efectivas.

No se registraba automáticamente el inicio, fin y duración del paro: otro aspecto prioritario fue la falta de precisión en los tiempos reales de paro. La duración se estimaba de manera subjetiva, lo cual afectaba los indicadores de eficiencia y la confiabilidad de los reportes por estación o por turno.

Retrasos al reanudar la línea por procesos manuales: el procedimiento para quitar los paros se realizaba manualmente o mediante comunicación verbal, lo cual ocasionaba pérdida de tiempo y retrasos innecesarios en la reactivación de la línea, impactando directamente en la productividad.

Ausencia de una base de datos estructurada para consulta y análisis: no se contaba con una herramienta digital que centralizara y organizara los datos históricos de los paros. Esta situación impedía generar reportes detallados y limitaba el análisis estratégico de información por estación, causa, área o turno.

Dificultad para identificar las estaciones o turnos más afectados: debido a la falta de trazabilidad por ubicación y horario, no era posible conocer con claridad qué estaciones acumulaban más tiempo de paro, ni en qué turnos se concentraban los mayores incidentes, lo cual complicaba la asignación de recursos y la aplicación de medidas correctivas puntuales.

Dependencia de registros manuales basados en la memoria del supervisor: finalmente, se detectó que los registros se llenaban “a como se acordaban”. Esta práctica generaba inconsistencias, pérdida de información y reducía la confiabilidad de los datos disponibles.

Objetivos

Objetivo General

Diseñar e implementar un sistema de monitoreo en tiempo real para el control de paros externos e internos en la Línea 1 de producción de la empresa SAS Automotive Systems, S.A. de C.V., que permita registrar automáticamente el inicio, fin y duración de los eventos, así como clasificar sus causas, con el fin de mejorar la eficiencia operativa y apoyar la toma de decisiones.

Objetivos Específicos

1. **Analizar el proceso actual de registro y control de paros en la Línea 1**, identificando las principales problemáticas relacionadas con la recopilación de datos, tiempos muertos y causas de paro.
2. **Desarrollar una interfaz web conectada a un PLC Siemens S7-1500**, que permita el monitoreo en tiempo real del estado de cada estación y la visualización automatizada de los paros.
3. **Integrar una base de datos estructurada (SQLAlchemy)** para el almacenamiento automático de los registros de paros, incluyendo estación, tipo de paro, área, causa, inicio, fin y duración.
4. **Habilitar el uso de tablets en planta**, desde las cuales los operadores puedan quitar el paro y seleccionar la causa correspondiente desde un menú desplegable, sin afectar el flujo operativo.
5. **Generar reportes y estadísticas por estación y turno**, que permitan analizar la frecuencia, duración e impacto de los paros, facilitando la identificación de áreas críticas y la mejora continua.
6. **Reducir el tiempo total de paro y eliminar la dependencia de registros manuales**, mediante la automatización del proceso de captura y control de información en planta.

Justificación

El desarrollo del proyecto “**Web Server de paros externos de Clever**” surge de la necesidad de mejorar el registro y seguimiento de los paros en la Línea 1 de producción de Motherson, optimizando la gestión de fallos y reduciendo los tiempos de inactividad. Actualmente, el registro se realiza de forma manual o incompleta, lo que limita la toma de decisiones en tiempo real y dificulta la trazabilidad de los incidentes.

Con la implementación de un **web server conectado a un PLC Siemens S7-1500**, se automatiza y digitaliza el proceso, permitiendo una captura precisa del inicio, fin y duración de cada paro, así como la selección de la causa a través de una lista desplegable y campo para observaciones. El sistema también contempla notificaciones automáticas si no se registra una causa en tiempo determinado, asegurando una respuesta rápida ante fallos.

Este proyecto impacta directamente en la **eficiencia operativa**, al reducir tiempos muertos, mejorar la comunicación entre operadores y supervisores, y fortalecer el control sobre la producción. Además, al integrarse en la red interna de Motherson, garantiza la **seguridad y disponibilidad de la información** en tiempo real.

Marco teórico

1. Manufactura y Producción en la Industria Automotriz

La producción en la industria automotriz se distingue por su enfoque en la **eficiencia de procesos** y el control de calidad riguroso. Para una empresa como SAS Automotive Systems, que produce componentes críticos como los cockpits de modelos de alta demanda como el Jetta y el Tayron, la **eficiencia operativa** no es solo una meta, sino un imperativo comercial. La Línea 1 de producción, al ser la más grande de la planta, es un ecosistema complejo donde la sincronización y el flujo

constante de trabajo son fundamentales. Cualquier interrupción, sin importar cuán breve sea, puede tener efectos negativos en cascada, afectando los tiempos de entrega, aumentando los costos de producción y comprometiendo la satisfacción del cliente.

1.1. Conceptos de Producción y Ensamblaje en Línea

La línea de producción en la que se desarrolla el proyecto opera bajo el modelo de **producción por flujo continuo** o en línea de ensamblaje. Este sistema se caracteriza por la disposición secuencial de las estaciones de trabajo, donde el producto se mueve de una estación a otra para que se le agreguen componentes de forma progresiva. En este modelo, el **takt time** (tiempo de ciclo requerido) es una métrica clave. El takt time se calcula dividiendo el tiempo de producción disponible por la demanda de los clientes. Por ejemplo, si una planta tiene 480 minutos de producción al día y necesita producir 480 unidades, el takt time es de 1 minuto por unidad.

El ritmo de la línea se ajusta a este tiempo de ciclo para asegurar que la producción satisfaga la demanda. Cualquier **desviación** o **paro** de la línea rompe esta cadencia. Los paros pueden ser causados por una variedad de factores, como la falta de materiales, fallas en la maquinaria, problemas de calidad o la necesidad de realizar ajustes. El problema identificado en el proyecto es que estos paros no se registraban de manera precisa, lo que impedía a la gerencia comprender la frecuencia y las causas reales de estas interrupciones. La falta de datos objetivos llevaba a suposiciones y decisiones ineficaces. Al automatizar el registro, el sistema del web server proporciona una visión clara de dónde y por qué ocurren las interrupciones, permitiendo que la empresa tome medidas concretas para mantener el ritmo de producción.

1.2. Lean Manufacturing y la Reducción de Desperdicios

La filosofía de **Lean Manufacturing** (Manufactura Esbelta) es una metodología de gestión que busca maximizar el valor para el cliente mientras se minimizan los **desperdicios** (o muda en japonés). Los paros de producción son un claro ejemplo

de uno de los siete desperdicios principales: el **desperdicio de espera**. Este desperdicio ocurre cuando los empleados, máquinas o procesos deben detenerse y esperar a que una actividad previa se complete, generando tiempo de inactividad que no agrega valor al producto.

En la planta de SAS Automotive Systems, el tiempo perdido por los paros no registrados o mal documentados representaba un desperdicio significativo. Los supervisores y líderes de línea perdían tiempo valioso llenando formatos de papel de forma estimada, en lugar de centrarse en la resolución de los problemas de fondo. El proyecto del web server aborda directamente este problema al eliminar la necesidad de este registro manual. Al automatizar la detección y el registro de los paros, el sistema permite que los supervisores se concentren en la **toma de decisiones** para resolver la causa raíz de la interrupción. Esta automatización es un paso crucial hacia la adopción completa de los principios de Lean Manufacturing, ya que proporciona los datos necesarios para un ciclo de **mejora continua**. Los datos precisos revelan patrones, lo que permite a la empresa identificar las áreas de mayor desperdicio y enfocar sus esfuerzos de mejora de manera más efectiva.

1.3. OEE (Overall Equipment Effectiveness) y su Impacto en la Disponibilidad

El **OEE** es el estándar global para medir la **efectividad de la producción** de una máquina o una línea. Es una métrica compuesta por tres factores que se multiplican entre sí: **disponibilidad, rendimiento y calidad**.

1. **Disponibilidad:** Mide el porcentaje de tiempo que una máquina está realmente disponible para funcionar en comparación con el tiempo que debería estarlo. Se ve afectada por los **paros planificados y no planificados**. Los paros externos e internos que tu proyecto registra caen directamente en esta categoría.
2. **Rendimiento:** Mide la velocidad a la que la máquina opera en comparación con su velocidad de diseño.
3. **Calidad:** Mide la cantidad de productos aceptables producidos en comparación con la cantidad total producida.

El proyecto del web server tiene un impacto directo en el factor de **disponibilidad**. Antes, el cálculo de la OEE se basaba en datos subjetivos, lo que dificultaba un análisis preciso de las pérdidas de tiempo. Con el nuevo sistema, los datos de los paros son precisos, lo que permite calcular la disponibilidad con una fiabilidad total. Esto proporciona una base sólida para que la gerencia pueda tomar decisiones estratégicas. Por ejemplo, si el análisis de los datos del sistema muestra que una máquina específica está deteniendo la producción con frecuencia debido a fallos recurrentes (disponibilidad baja), se puede justificar una inversión en **mantenimiento predictivo** o en la sustitución de un componente. Por el contrario, si los datos revelan que la mayoría de los paros son causados por la falta de materiales, se puede optimizar la cadena de suministro. El sistema convierte la disponibilidad, antes un dato estimado, en una métrica **medible y accionable**, lo que es fundamental para la mejora continua.

2. Automatización y Control Industrial

La **automatización** es un pilar fundamental en la manufactura moderna, permitiendo a las empresas alcanzar niveles de precisión, velocidad y fiabilidad que serían imposibles con la mano de obra humana. En el sector automotriz, la automatización es la fuerza motriz detrás de la producción en masa de componentes complejos como los cockpits. El control industrial, por su parte, es el conjunto de sistemas que gestionan y dirigen los procesos productivos. La implementación de un sistema de monitoreo en tiempo real, como el desarrollado en este proyecto, se basa en la integración de la automatización física con los sistemas de control digital.

2.1. Controladores Lógicos Programables (PLC)

Los **Controladores Lógicos Programables (PLC)** son computadoras industriales diseñadas para controlar procesos de fabricación. A diferencia de las computadoras personales, los PLC están contruidos para ser extremadamente robustos y fiables, capaces de operar en entornos industriales hostiles con alta fiabilidad y en tiempo

real (Bolton, 2015). Su función principal es recibir señales de entrada de sensores, procesar la información de acuerdo con una lógica programada y enviar señales de salida a los actuadores para controlar la maquinaria. El PLC es el "cerebro" de la Línea 1 de producción de SAS Automotive Systems, ya que es el encargado de secuenciar las operaciones de ensamblaje, activar y desactivar los equipos y, crucialmente para este proyecto, gestionar los estados de paro.

En el contexto del proyecto "Web server de paro externos de clever", la elección del **Siemens S7-1500** es estratégica. Este PLC de última generación no solo ofrece un alto rendimiento en el procesamiento de tareas de control, sino que también cuenta con capacidades avanzadas de comunicación. Su integración con redes industriales permite que otros sistemas, como el servidor web, se conecten a él para leer el estado de la máquina y las variables del proceso. Esta comunicación es la piedra angular del proyecto, ya que el servidor web no adivina el estado de la línea de producción, sino que lo obtiene directamente de la fuente más precisa y fiable: el PLC. La comunicación directa con el PLC elimina por completo el error humano en el registro de los eventos, garantizando que el inicio y el fin de cada paro se capturen con una precisión de milisegundos, lo cual es vital para el cálculo exacto de la OEE.

2.2. Sensores y Actuadores: La Interacción en el Control Industrial

El concepto de **sensores y actuadores** es fundamental para entender cómo funciona la automatización. Los **sensores** son dispositivos que detectan cambios o eventos en el entorno físico (como la presencia de una pieza, una luz de paro encendida o la presión en un circuito) y convierten esa información en una señal eléctrica que el PLC puede interpretar. Los **actuadores**, por otro lado, son dispositivos que convierten una señal eléctrica del PLC en una acción física, como encender o apagar un motor, mover un pistón hidráulico o detener una banda transportadora (Hughes, 2011).

El proyecto de residencia se distingue porque no requiere la instalación de nuevos sensores físicos. En su lugar, aprovecha la información que el propio PLC ya

procesa. Cada vez que ocurre un paro en una estación, ya sea por una emergencia (paro interno) o por la intervención del operador (paro externo), el PLC activa una **variable de estado** o una **bandera digital** que indica la condición de paro. El servidor web está programado para monitorear continuamente estas variables a través del protocolo de comunicación industrial. Cuando el servidor detecta que la variable de paro cambia a "Activo", registra la hora de inicio del evento. Cuando la variable vuelve a "Inactivo" (ya sea porque el supervisor quitó el paro desde la tablet o porque el problema se resolvió), el servidor registra la hora de finalización. Esta interacción virtual entre el servidor y el PLC, mediada por las variables de estado, es lo que permite un sistema de registro automatizado, eliminando la necesidad de que los supervisores llenen formatos en papel o utilicen cronómetros manuales.

3. La Industria 4.0 y la Conectividad

La **Industria 4.0** representa un cambio de paradigma en la manufactura, impulsado por la integración de tecnologías digitales en los procesos productivos. Este concepto, que se ha popularizado como la **cuarta revolución industrial**, busca crear fábricas inteligentes y completamente interconectadas, donde los sistemas ciberfísicos, el Internet de las Cosas (IoT) y la computación en la nube trabajan de la mano para optimizar la producción (Kagermann, Wahlster, & Helbig, 2013). El proyecto de residencia, al conectar la línea de producción a un servidor web para el monitoreo de paros, se alinea perfectamente con los principios de esta revolución, transformando un proceso analógico y manual en uno digital y automatizado.

3.1. Fundamentos de la Industria 4.0: Hacia los Sistemas Ciberfísicos

La **Industria 4.0** se basa en cuatro principios de diseño clave que guían la transformación digital en las fábricas:

1. **Interconexión:** Se refiere a la capacidad de las máquinas, dispositivos, sensores y personas de comunicarse entre sí a través de redes como el

Internet de las Cosas Industrial (IIoT). El proyecto ejemplifica este principio al establecer una conexión directa y constante entre el PLC de la línea de producción y el servidor web, creando una red de datos en tiempo real.

2. **Transparencia de la Información:** Implica la creación de una copia virtual o digital del mundo físico. En el contexto de los paros, esto se traduce en la capacidad de ver en una pantalla de computadora o tablet la hora exacta de inicio, fin y duración de un paro, sin necesidad de registros manuales en papel. Esta "copia digital" proporciona un contexto vital para la toma de decisiones.
3. **Asistencia Técnica:** Consiste en ayudar a los operarios y supervisores en la toma de decisiones. El sistema no solo muestra los datos de los paros, sino que también los presenta de una forma comprensible (en dashboards y gráficos), lo que facilita que los supervisores entiendan la situación de la línea y tomen decisiones rápidas y bien informadas.
4. **Decisiones Descentralizadas:** Se refiere a la capacidad de los sistemas ciberfísicos para tomar decisiones por sí mismos en tiempo real, sin la intervención de un supervisor. Aunque el proyecto aún requiere la intervención humana para asignar la causa de un paro, la automatización del registro de inicio y fin es un paso significativo hacia una mayor descentralización en el control de la producción.

El proyecto es un claro ejemplo de un **sistema ciberfísico (CPS)**, ya que integra los procesos físicos de la línea de ensamblaje con los procesos digitales del servidor web. Esta sinergia es lo que permite una gestión más eficiente y proactiva de la producción.

3.2. Internet de las Cosas Industrial (IIoT) y la Conectividad de la Planta

El **Internet de las Cosas Industrial (IIoT)** es un componente esencial de la **Industria 4.0**. Se refiere a la red de dispositivos, sensores, actuadores y máquinas conectadas que se comunican entre sí para recopilar, intercambiar y analizar datos en el entorno de fabricación (Gilchrist, 2016). El proyecto aprovecha la tecnología

IIoT al convertir la línea de producción en una "cosa inteligente" que puede generar y transmitir datos sobre su estado de operación.

Antes del proyecto, el estado de paro de la línea de producción era una "información aislada", conocida solo por el operario y el supervisor de área. Con la implementación del servidor web, esta información se digitaliza y se convierte en un dato accesible para todos los niveles de la organización. El PLC actúa como un nodo dentro de la red IIoT, transmitiendo la información de paro al servidor. Este servidor, a su vez, procesa y almacena los datos en una base de datos para su posterior análisis. Esta conectividad permite una **transparencia total** sobre lo que sucede en el piso de producción, eliminando los silos de información y facilitando una colaboración más efectiva entre los equipos de producción, mantenimiento y calidad. En esencia, el proyecto convierte datos analógicos y subjetivos en información digital, lo que es un paso fundamental para cualquier planta que busque modernizarse y adoptar las tecnologías de la Industria 4.0.

4. Bases de Datos y Gestión de la Información

La información es el activo más valioso de cualquier sistema de monitoreo en tiempo real. Para que los datos recopilados por el web server sean útiles y sirvan para la toma de decisiones, deben ser almacenados, organizados y gestionados de forma eficiente. La transición de un registro en papel a un sistema digital requiere una infraestructura de gestión de datos robusta, que garantice la integridad, la accesibilidad y la seguridad de la información. El proyecto se basa en el uso de una base de datos relacional y una herramienta de mapeo objeto-relacional para simplificar esta gestión.

4.1. Sistemas de Gestión de Bases de Datos (SGBD) para la Centralización de Datos

Un **Sistema de Gestión de Bases de Datos (SGBD)** es un software diseñado para la creación, mantenimiento y manipulación de bases de datos. Su principal función es proporcionar un entorno estructurado donde la información pueda ser almacenada de manera lógica, lo que facilita su posterior recuperación y análisis (Silberschatz, Korth, & Sudarshan, 2011). En un entorno de producción como el de SAS Automotive Systems, la centralización de los datos de paro es crucial. Antes del proyecto, la información se encontraba dispersa en múltiples formatos de papel, lo que dificultaba la consolidación y el análisis a largo plazo. Por ejemplo, era casi imposible saber cuántos paros por "falta de material" ocurrieron en toda la línea durante el último trimestre.

El SGBD del proyecto, que se implementa en el servidor, actúa como un repositorio central para todos los registros de paro. Cada evento de paro se convierte en una fila en la tabla de la base de datos, con campos específicos para registrar la estación, la hora de inicio, la hora de finalización, la duración real y la causa. Esta estructura de datos relacional no solo garantiza que la información esté organizada, sino que también permite la ejecución de consultas complejas. Por ejemplo, un supervisor puede fácilmente generar un reporte que muestre la duración promedio de los paros por cada estación o identificar el área con el mayor número de interrupciones en un turno específico. La **centralización de datos** elimina la redundancia, reduce el riesgo de pérdida de información y proporciona una fuente única de verdad para la toma de decisiones.

4.2. Mapeo Objeto-Relacional (ORM) con SQLAlchemy

La interacción entre el lenguaje de programación del servidor (Python) y la base de datos relacional puede ser compleja, ya que requieren lenguajes y paradigmas diferentes (la programación orientada a objetos versus el lenguaje de consulta estructurado SQL). Un **Mapeo Objeto-Relacional (ORM)** es una herramienta que

actúa como un puente entre ambos. Un ORM permite a los desarrolladores interactuar con los datos de una base de datos utilizando objetos de su lenguaje de programación nativo, sin tener que escribir sentencias SQL (Date, 2004). El ORM traduce automáticamente las operaciones de objeto a sentencias SQL y viceversa.

El proyecto del web server utiliza **SQLAlchemy**, un ORM de Python. Esta herramienta simplifica la lógica de negocio del servidor de varias maneras. En lugar de escribir una consulta SQL como INSERT INTO paros (estacion, inicio, fin) VALUES ('Estacion 1', '...', '...'), el desarrollador simplemente crea un objeto de Python que representa un paro y lo guarda en la base de datos. Esta abstracción no solo hace que el código sea más legible y fácil de mantener, sino que también mejora la **seguridad**. El ORM se encarga de escapar y validar los datos de entrada, previniendo ataques comunes como la inyección SQL. La combinación de una base de datos estructurada con la facilidad de uso de un ORM es lo que permite que el sistema del web server sea robusto, escalable y seguro para gestionar grandes volúmenes de datos de paro a lo largo del tiempo.

5. Desarrollo Web para Aplicaciones de Monitoreo

La interfaz web es la cara del proyecto, el punto de contacto directo entre los datos del PLC y los supervisores de la línea de producción. El desarrollo de esta aplicación no solo implica la creación de una plataforma funcional, sino también el diseño de una experiencia de usuario que sea intuitiva y eficiente en un entorno industrial. La elección de una arquitectura **cliente-servidor** y de tecnologías de código abierto como **Python**, **JavaScript**, **HTML** y **CSS** se justifica por su versatilidad, escalabilidad y la capacidad de crear una solución accesible desde cualquier dispositivo.

5.1. Arquitectura Cliente-Servidor y el Monitoreo Remoto

La aplicación de monitoreo de paros se basa en la **arquitectura cliente-servidor**, un modelo distribuido donde las tareas se dividen entre los "proveedores de servicios" (el servidor) y los "solicitantes de servicios" (los clientes). En este proyecto:

- **El servidor:** Es la computadora donde se ejecuta el código de Python que gestiona la lógica de la aplicación. Se comunica con el PLC para obtener los datos de paro y con la base de datos para almacenar y recuperar la información. Es el motor que procesa todas las solicitudes.
- **El cliente:** Es el navegador web que se ejecuta en las tablets utilizadas por los supervisores. El cliente solicita información al servidor (por ejemplo, el estado actual de la línea) y envía comandos (como la selección de una causa de paro).

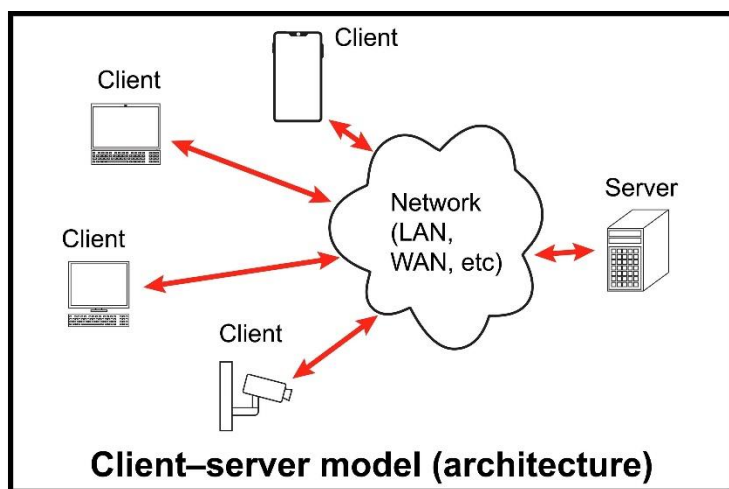


Ilustración 3.- Arquitectura cliente-servidor

Este modelo de arquitectura es ideal para el proyecto por su flexibilidad. Permite a los supervisores acceder a la información en tiempo real desde cualquier lugar de la planta, siempre y cuando tengan acceso a la red Wi-Fi de la compañía. Elimina la necesidad de estaciones de trabajo fijas y facilita un monitoreo remoto que agiliza la toma de decisiones y reduce los tiempos de reacción ante un paro. La separación

de responsabilidades entre el servidor y el cliente también mejora la escalabilidad, ya que el sistema puede soportar múltiples clientes concurrentes sin una degradación significativa del rendimiento.

5.2. Tecnologías Front-End: La Interfaz de Usuario

El **front-end** es la parte de la aplicación que el usuario ve y con la que interactúa. En este proyecto, el front-end se construye utilizando tres tecnologías principales que trabajan en conjunto:

- **HTML (HyperText Markup Language):** Proporciona la estructura del contenido web. Define los elementos de la página, como los títulos, las tablas que muestran los paros, los botones para interactuar y los formularios para la entrada de datos.
- **CSS (Cascading Style Sheets):** Se encarga del diseño visual y la apariencia de la interfaz. Permite definir los colores, las tipografías, los márgenes y la disposición de los elementos. Es fundamental para crear una interfaz que no solo sea funcional, sino también visualmente clara y fácil de leer en las pantallas de las tablets, con botones grandes y un contraste de colores adecuado.
- **JavaScript:** Es el lenguaje de programación que añade interactividad y dinamismo a la interfaz. JavaScript se encarga de que la página se actualice en tiempo real sin tener que recargarla. Cuando un paro ocurre, el front-end usa JavaScript para solicitar los nuevos datos al servidor y actualizar la tabla de paros al instante. También valida los datos de entrada del usuario y maneja la lógica de la interfaz, como ocultar o mostrar ciertos elementos según el estado de la línea.

5.3. Tecnologías Back-End: La Lógica del Servidor

El **back-end** es la lógica que se ejecuta en el servidor y maneja todas las operaciones que no son visibles para el usuario. Para este proyecto, el lenguaje de programación principal del back-end es **Python**. La elección de Python se justifica

por varias razones: es un lenguaje de programación de alto nivel, de fácil lectura y con una vasta comunidad de desarrolladores y una gran cantidad de librerías especializadas.

El back-end de Python es el cerebro detrás de las operaciones del sistema. Sus responsabilidades incluyen:

- **Comunicación con el PLC:** Utiliza una librería especializada para comunicarse con el PLC a través del protocolo de red y obtener el estado de la línea en tiempo real.
- **Gestión de la Base de Datos:** Se conecta a la base de datos a través de SQLAlchemy y maneja las operaciones de almacenamiento de nuevos registros de paro y la recuperación de datos para la generación de reportes.
- **Lógica de Negocio:** Procesa las solicitudes del cliente (por ejemplo, el supervisor selecciona una causa de paro desde la tablet) y ejecuta las acciones correspondientes. También se encarga de la lógica para calcular la duración exacta de los paros y clasificar si son internos o externos.
- **API (Application Programming Interface):** El back-end expone una API REST para que el front-end pueda solicitar y enviar datos de forma estructurada.

En conjunto, el back-end y el front-end forman un sistema cohesivo que automatiza el proceso de registro de paros, mejorando la eficiencia operativa y proporcionando información valiosa para la toma de decisiones.

6. Protocolos de Comunicación Industrial

En el corazón de cualquier sistema de monitoreo en tiempo real, especialmente uno que integra el mundo físico de la planta con el mundo digital del software, se encuentra la **comunicación industrial**. Los **protocolos de comunicación** son las reglas y estándares que permiten a dispositivos de distintos fabricantes intercambiar

datos de manera fiable y eficiente. La elección de un protocolo adecuado es crucial para asegurar la latencia mínima, la integridad de los datos y la escalabilidad del sistema. El proyecto del web server se basa en el uso de un protocolo estándar de la industria para establecer la conexión entre el PLC Siemens y el servidor, garantizando una transferencia de datos precisa y en tiempo real.

6.1. Comunicación Ethernet/IP para la Interacción en Tiempo Real

Ethernet/IP (Ethernet Industrial Protocol) es un protocolo de red industrial que utiliza la infraestructura de red Ethernet estándar para la transferencia de datos de control y de información en un entorno de fabricación. Es uno de los protocolos industriales más utilizados, conocido por su capacidad para manejar tanto datos de control en tiempo real (críticos para la operación) como datos de información (para la supervisión y la gestión). A diferencia de los protocolos de red de oficina, Ethernet/IP está diseñado para garantizar una comunicación fiable y de baja latencia, incluso en entornos con mucho ruido eléctrico y con múltiples dispositivos conectados (Holbrook & Al-Akaidi, 2014).

En el contexto del proyecto del web server, Ethernet/IP es el "idioma" que utilizan el PLC y el servidor para hablar entre sí. El servidor web, que corre en una computadora, está programado para conectarse al PLC Siemens S7-1500 a través de la red Ethernet de la planta. Utilizando una librería de Python compatible con el protocolo, el servidor puede leer las variables de memoria del PLC que indican el estado de cada estación de la línea de producción. Esta comunicación bidireccional no solo permite al servidor registrar el inicio y el fin de cada paro, sino que también posibilita a los supervisores enviar una señal al PLC para "quitar un paro" desde la tablet. La implementación de este protocolo asegura que los datos se transfieran de manera segura y casi instantánea, un requisito fundamental para un sistema de monitoreo en tiempo real.

6.2. OPC UA (Open Platform Communications Unified Architecture) para la Interoperabilidad

OPC UA es un protocolo de comunicación industrial que va más allá de la simple transferencia de datos. Es un estándar de interoperabilidad que permite la comunicación segura y fiable entre dispositivos de diferentes fabricantes, sistemas de control (como los PLC), aplicaciones de supervisión (como SCADA) y sistemas de gestión empresarial (como ERP) (Mahdavi, 2013). Su principal ventaja es que es independiente de la plataforma, lo que significa que puede ejecutarse en cualquier sistema operativo y en cualquier dispositivo. OPC UA no solo transfiere datos, sino que también incluye metadatos (información sobre los datos), lo que permite a los sistemas comprender el significado de la información que reciben.

Si bien el proyecto inicial se centra en la comunicación directa con el PLC a través de Ethernet/IP, la arquitectura del sistema está diseñada con la escalabilidad en mente. La compatibilidad futura con OPC UA permitiría que el sistema de monitoreo se integre sin problemas con otros sistemas de la planta, como el software de mantenimiento, el sistema de calidad o una plataforma de Business Intelligence más avanzada. Por ejemplo, el sistema de mantenimiento podría recibir automáticamente una notificación cuando se registre un número elevado de paros internos en una estación, facilitando un mantenimiento predictivo. La adopción de este estándar demuestra una visión a largo plazo para el proyecto, convirtiéndolo en un componente de un ecosistema digital más amplio dentro de la fábrica.

7. Usabilidad y Diseño de Interfaces de Usuario (UI/UX)

El éxito de una herramienta de software, especialmente en un entorno industrial, no depende únicamente de su funcionalidad técnica. Una interfaz robusta y una lógica de programación impecable no servirán de mucho si los usuarios finales (los supervisores y líderes de línea en este caso) encuentran el sistema difícil de usar,

confuso o lento. La **Usabilidad** y el **Diseño de Experiencia de Usuario (UX)** son disciplinas que se enfocan en hacer que la interacción entre el ser humano y el sistema sea lo más fluida, eficiente y satisfactoria posible. Para el proyecto del web server, estos principios fueron cruciales, ya que el sistema se usa en el piso de producción, donde la rapidez y la claridad son de suma importancia.

7.1. Diseño de Interfaz de Usuario (UI) para la Usabilidad en el Entorno Laboral

El **Diseño de Interfaz de Usuario (UI)** se centra en la apariencia y la disposición de los elementos de la pantalla que permiten al usuario interactuar con la aplicación. En el contexto del proyecto, el diseño de la UI para las tablets de los supervisores se enfocó en principios clave de usabilidad industrial:

- **Claridad y Simplicidad:** La interfaz debe ser simple y directa. La pantalla principal muestra solo la información crítica: el estado de cada estación (en paro o en operación), el tiempo de paro transcurrido y la opción para registrar una causa. Los botones para "quitar paro" deben ser grandes y fáciles de presionar, incluso para usuarios que llevan guantes de seguridad o que están operando el dispositivo rápidamente.
- **Consistencia:** El uso de colores, tipografías y la ubicación de los elementos debe ser consistente en toda la aplicación. Por ejemplo, un color específico, como el rojo, siempre debe indicar un paro activo, mientras que un color como el verde indica que la línea está en operación. Esta consistencia visual reduce la curva de aprendizaje y minimiza los errores de operación (Nielsen & Budiu, 2013).
- **Retroalimentación:** La UI debe proporcionar una retroalimentación inmediata al usuario. Cuando un supervisor selecciona una causa de paro, la interfaz debe mostrar una confirmación visual para asegurar que la acción se ha registrado correctamente.

La implementación de estos principios de UI en el proyecto asegura que la herramienta sea fácil de usar para el personal de la planta, que no siempre tiene experiencia previa con software de monitoreo.

7.2. Experiencia de Usuario (UX) y el Impacto en la Eficiencia Operativa

La **Experiencia de Usuario (UX)** es un concepto más amplio que la UI. Abarca la totalidad de la experiencia del usuario al interactuar con el sistema, incluyendo sus emociones, percepciones y reacciones (Norman, 2013). Una buena UX en el contexto del proyecto significa que el supervisor se siente empoderado y no frustrado por la herramienta. La UX se ve influenciada por varios factores:

- **Eficiencia:** El tiempo es crucial en la línea de producción. El sistema debe responder de manera casi instantánea cuando se presiona un botón. Un retraso de solo unos segundos puede llevar a la frustración y a que los usuarios vuelvan a los métodos de registro manuales. La arquitectura del sistema, que utiliza un back-end de Python rápido y una comunicación directa con el PLC, garantiza esta eficiencia.
- **Satisfacción:** Si el supervisor siente que la herramienta le ayuda a resolver problemas y a ser más productivo, la adopción del sistema será exitosa. El hecho de que el sistema automatice el registro de los paros y les permita a los supervisores concentrarse en la resolución de problemas de fondo (en lugar de perder tiempo documentando) contribuye a una experiencia positiva.

En última instancia, un buen diseño de UI y UX convierte al web server en una herramienta valiosa para el equipo de producción, no solo en un sistema de registro de datos. La facilidad de uso es lo que asegura que el sistema sea utilizado de manera consistente, garantizando que los datos recopilados sean completos y precisos para su posterior análisis.

8. Mejora Continua y Toma de Decisiones Basada en Datos

El valor final del proyecto del "Web server de paro externos de clever" no reside en su capacidad técnica para registrar datos de forma automática, sino en la manera en que esos datos se transforman en conocimiento accionable para impulsar la **mejora continua**. Los datos brutos son solo el punto de partida. Es a través del **análisis y la visualización de datos** que los supervisores y la gerencia pueden identificar patrones, detectar problemas recurrentes y tomar decisiones informadas que optimicen los procesos productivos. Este enfoque estratégico convierte al sistema en una herramienta de gestión esencial, alineada con las filosofías de calidad y eficiencia de la manufactura moderna.

8.1. Métodos de Análisis de Causa Raíz (RCA)

El **Análisis de Causa Raíz (RCA)** es un método estructurado para resolver problemas al identificar las causas subyacentes de un evento o un problema, en lugar de simplemente tratar los síntomas (Anderson & Fagerhaug, 2000). En la Línea 1 de producción, antes del proyecto, un paro era simplemente un evento que debía ser resuelto para que la línea volviera a funcionar. Con el sistema de monitoreo, cada paro se convierte en un registro detallado que puede ser investigado.

El sistema proporciona los datos necesarios para aplicar técnicas de RCA como los **"5 Porqués"** o el **Diagrama de Ishikawa (o de espina de pescado)**. Por ejemplo, si los datos del servidor web muestran que la "Estación 5" tiene el mayor número de paros por "falla de máquina", un supervisor puede usar estos datos para iniciar un análisis de los 5 Porqués:

- **¿Por qué se detuvo la máquina?** Porque falló el motor.
- **¿Por qué falló el motor?** Porque se sobrecalentó.
- **¿Por qué se sobrecalentó?** Porque el sistema de enfriamiento estaba sucio.
- **¿Por qué el sistema de enfriamiento estaba sucio?** Porque no se le dio mantenimiento preventivo a tiempo.

- **¿Por qué no se le dio mantenimiento preventivo?** Porque el plan de mantenimiento no lo incluía.

Este análisis, que antes se basaba en la memoria y la suposición, ahora puede fundamentarse en los datos exactos del sistema, lo que permite a la gerencia atacar la causa raíz del problema (la falta de un plan de mantenimiento) y no solo los síntomas (el motor fallando).

8.2. Business Intelligence (BI) y Visualización de Datos para la Acción

El **Business Intelligence (BI)** es un conjunto de estrategias y tecnologías para analizar datos de negocio y presentar información procesable a los usuarios finales (Sharda, Delen, & Turban, 2020). El proyecto del web server integra elementos básicos de BI a través de la **visualización de datos**. En lugar de simplemente mostrar una lista de registros, la interfaz web convierte los datos de paro en **gráficos, tablas dinámicas y dashboards** que son fáciles de interpretar.

Los líderes y supervisores pueden ver gráficos de barras que muestran la frecuencia de paros por estación, gráficos circulares que ilustran la distribución de los paros entre causas externas e internas, y tablas que resumen la duración total de los paros por turno. Esta visualización de datos permite:

- **Identificar tendencias:** Detectar si hay un aumento en los paros en un turno específico o si una causa particular está volviéndose más común.
- **Evaluar el impacto:** Cuantificar el tiempo de inactividad de cada estación en términos de horas o minutos, lo que permite priorizar los esfuerzos de mejora en las áreas de mayor impacto.
- **Tomar decisiones proactivas:** Basándose en los datos históricos, se pueden implementar planes de mantenimiento predictivo, reasignar personal o ajustar los procesos para evitar futuros paros.

En resumen, el sistema del web server no es solo un registrador de eventos, sino una **herramienta de inteligencia de negocio** que empodera a los usuarios con la

información que necesitan para optimizar la producción, reducir los costos y fomentar una cultura de mejora continua en la planta. La información, que antes estaba fragmentada y en papel, ahora se presenta de forma clara y accesible, sirviendo como la base para un crecimiento sostenible y una mayor eficiencia operativa.

Procedimiento y descripción de las actividades realizadas

El desarrollo del proyecto “Web Server de paro externos de Clever” se llevó a cabo siguiendo una serie de etapas que permitieron abordar de manera estructurada la problemática detectada en la Línea 1 de producción de la planta.

Análisis de la situación actual

Como primera fase del proyecto, se realizó un análisis de la situación actual, observando el flujo de trabajo y la distribución de la Línea 1, compuesta por 36 estaciones, un área de ECHECK y una sección de Final Inspection (FI). Durante esta etapa, se documentaron los métodos existentes para la notificación de paros, identificando dos tipos principales:

- Paros externos (Andon): Activados manualmente por los operadores mediante un cordón instalado en la mayoría de las estaciones.
- Paros internos (Clever): Generados por un sistema vinculado a condiciones específicas del proceso, generalmente relacionadas con problemas técnicos o de calidad que requieren intervención especializada.

Se detectaron deficiencias importantes en el registro de causas y tiempos de paro, particularmente en los paros externos, debido a que el registro dependía de procesos manuales y de la memoria de los operadores, lo que derivaba en datos incompletos o inexactos.

La línea cuenta con dos interfaces HMI, en las que se visualiza el estado de las estaciones tanto para paros externos como internos, lo que permite identificar de forma rápida en qué estación ocurre el paro y su tipo. Los paros externos pueden ser retirados directamente desde el HMI para que la línea continúe su avance, gracias a la conexión de este sistema con el PLC Siemens S7-1500. En cambio, los paros internos (Clever) únicamente pueden visualizarse, ya que están asociados a problemas de proceso que requieren atención y resolución especializada antes de reanudar la producción.

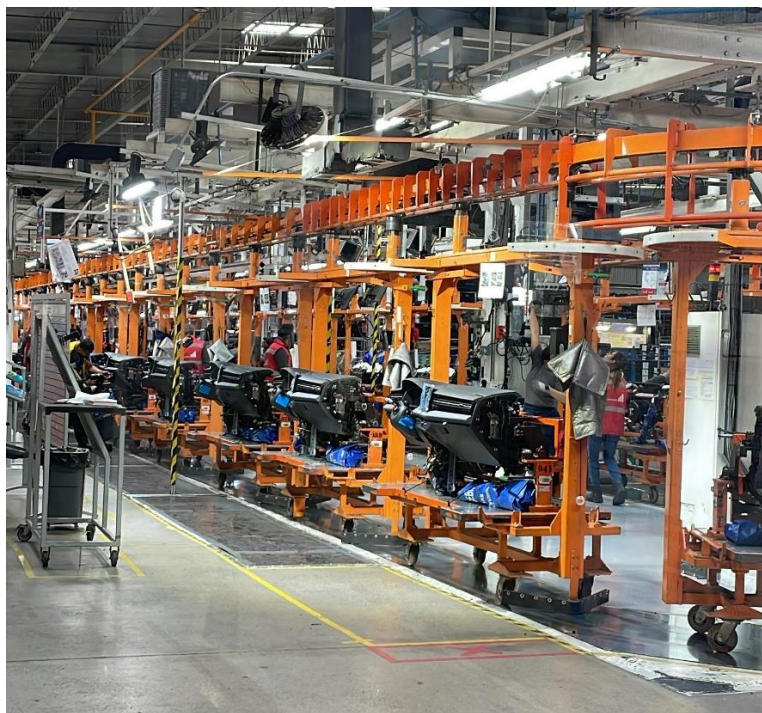


Ilustración 4.- Línea 1 de producción

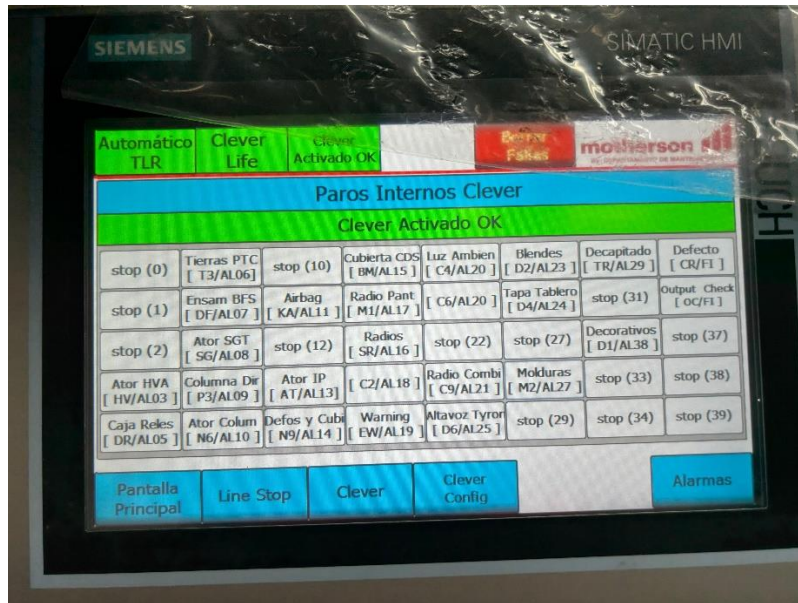


Ilustración 5.- HMI de paros

Definición de requerimientos del sistema

Posteriormente, se procedió a la definición de los requerimientos del sistema para asegurar su correcto funcionamiento, los cuales se clasifican en funcionales y no funcionales.

1. Requerimientos Funcionales

Son las funciones específicas que el sistema debe realizar:

- Registrar en tiempo real el inicio, fin y duración de cada paro.
- Clasificar el tipo de paro en externo (Andon) o interno (Clever).
- Permitir a los operadores reportar la causa del paro mediante una lista desplegable y un cuadro de texto para detalles adicionales.
- Guardar automáticamente la información de los paros en una base de datos SQLite.
- Permitir la eliminación de paros externos desde la interfaz web conectada al PLC Siemens S7-1500 para reanudar la producción.

- Mostrar en el HMI y en la interfaz web las estaciones donde ocurre el paro y su tipo.
- Generar reportes y estadísticas filtradas por estación, tipo de paro, área y turno.
- Bloquear botones de interfaz web para reanudar la línea si no se registra la causa después de 3 paros.

2. Requerimientos No Funcionales

Definen las condiciones bajo las cuales el sistema debe operar:

- Disponibilidad: El sistema debe estar operativo en tiempo real durante los tres turnos de producción.
- Usabilidad: La interfaz web debe ser intuitiva y de fácil navegación para los operadores y supervisores.
- Compatibilidad: El sistema debe integrarse a la red interna de la planta y ser accesible desde tablets y computadoras.
- Rapidez: La actualización de datos debe reflejarse en menos de 2 segundos después de un evento de paro.
- Escalabilidad: El sistema debe permitir su adaptación a otras líneas de producción en el futuro.

El cumplimiento de estos requerimientos garantiza que el sistema no solo resuelva las deficiencias actuales en el registro y control de paros, sino que también mejore la eficiencia operativa, la trazabilidad de incidentes y la toma de decisiones en la planta.

Arquitectura del sistema

Se basa en una arquitectura cliente-servidor, donde un PLC Siemens S7-1500 actúa como punto central de adquisición de datos, comunicándose con un servidor web que procesa, almacena y distribuye la información en tiempo real.

Componentes principales:

1. PLC Siemens S7-1500

- Recibe señales de las estaciones de trabajo relacionadas con paros externos (Andon) y paros internos (Clever).
- Controla la reanudación de la línea para paros externos.
- Envía los datos al servidor mediante comunicación industrial (Profinet o TCP/IP).

2. Servidor Web

- Desarrollado en Python y JavaScript, con una base de datos SQLite para almacenamiento.
- Procesa la información recibida del PLC y la presenta en una interfaz web accesible desde tablets y computadoras.

3. Interfaz Web

- Muestra en tiempo real el estado de cada estación.
- Permite registrar causas de paro, clasificarlas y eliminarlas en caso de paros externos.
- Genera reportes y estadísticas por turno, estación y tipo de paro.

4. HMI (Human Machine Interface)

- Ubicada en puntos estratégicos de la línea para visualizar en tiempo real el origen y tipo de paro.
- Sincronizada con el PLC para mostrar información precisa.

5. Red Interna de Planta

- Garantiza que la comunicación entre PLC, servidor y dispositivos de usuario sea segura y sin interrupciones.

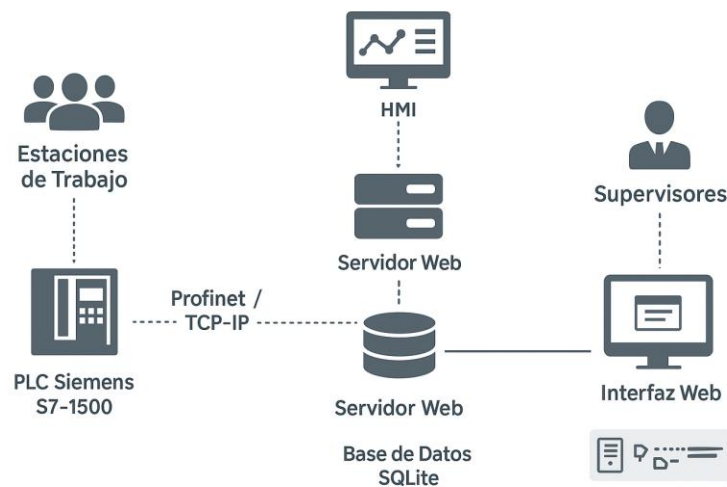


Ilustración 6.- Diagrama de arquitectura

Instalación y configuración de Siemens TIA Portal V19



Ilustración 7.- Logo TIA Portal V19

Se instalo y configuro el software Siemens Totally Integrated Automation Portal (TIA Portal) versión 19, el cual permite la programación, configuración y diagnostico de controladores lógicos programables (PLC), así como la integración de interfaces HMI y dispositivos de red.

Requerimientos previos

Antes de iniciar la instalación, se verifica los siguiente para garantizar la correcta operación del software:

- Sistema operativo compatible: Windows 10/ Windows 11 (64 bits).
- Memoria RAM recomendada: 16 GB.
- Espacio en disco disponible: 60 GB mínimo.
- Procesador: Intel Core i5 / i7 de 8ª generación o superior.

- Permisos de administrador para la instalación.
- Licencia válida de TIA Portal V19.

Procedimiento de instalación

1. Preparación del entorno

Se descargó la versión oficial de TIA Portal V19 desde la plataforma de Siemens.



Ilustración 8.- Descarga de TIA Portal v19

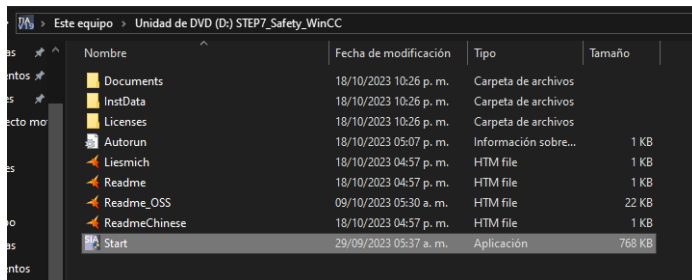


Ilustración 9.- Carpeta de TIA Portal v19

Los archivos se extrajeron en una carpeta dedicada para evitar conflictos con otros programas.

Se desactivo temporalmente el antivirus y el firewall para prevenir interrupciones durante la instalación.

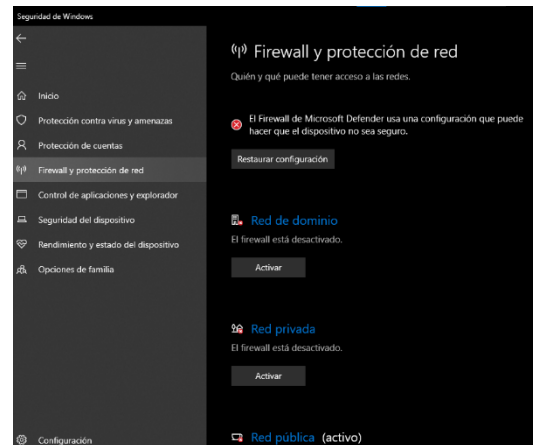
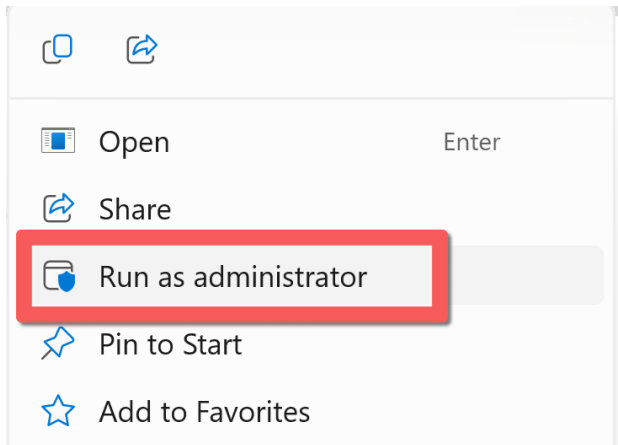


Ilustración 10.- Antivirus y firewalls desactivados

2. Instalación del software



Se ejecutó el instalador con privilegios de administrador.

Ilustración 11.- Privilegios de administrador para TIA Portal v19

Se seleccionó la opción “typical” para incluir:

- Herramientas de configuración del PLC Siemens s7-1500.
- Integración con HMI y dispositivos de comunicación.
- Librerías estándar y controladores de hardware.

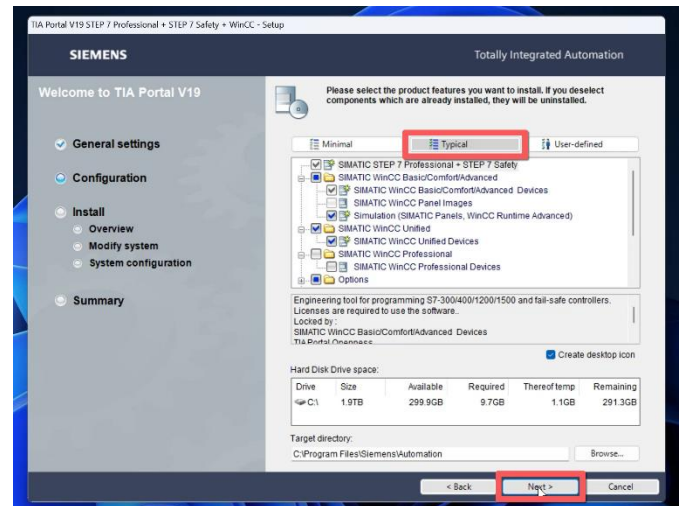


Ilustración 12.- Opción “typical” en TIA Portal V19

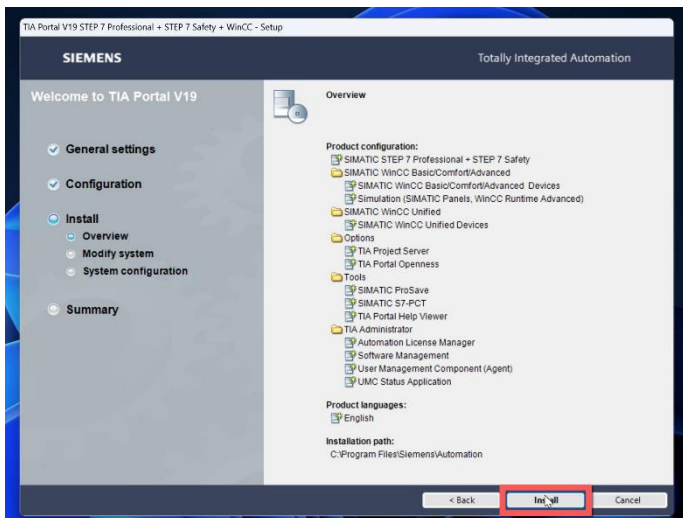


Ilustración 13.- Instalación final de TIA Portal v19

Se verifica que todo este correcto y finalmente se da clic en instalar, normalmente tarda de 35 a 40 min. Una vez finalizada la instalación se tiene que reiniciar el sistema y listo.

3. Carga del proyecto existente de la línea 1

Se cargó una copia del proyecto ya existente de la línea 1, que contiene los DB (Data Blocks) previamente configurados por la empresa, gracias a esta integración, fue posible visualizar en tiempo real cuándo un paro comenzaba y cuándo finalizaba, así como identificar qué estaciones entraban en paro, esta información resultó fundamental para conectar el PLC con el web server permitiendo que el sistema registrara automáticamente el inicio, fin y duración de los paros.

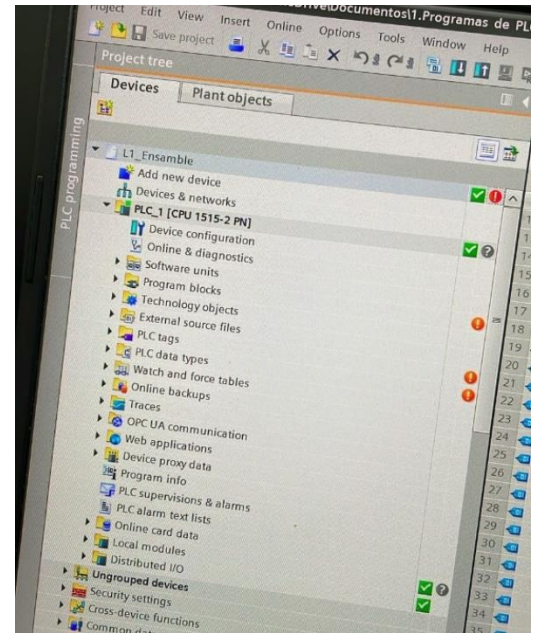
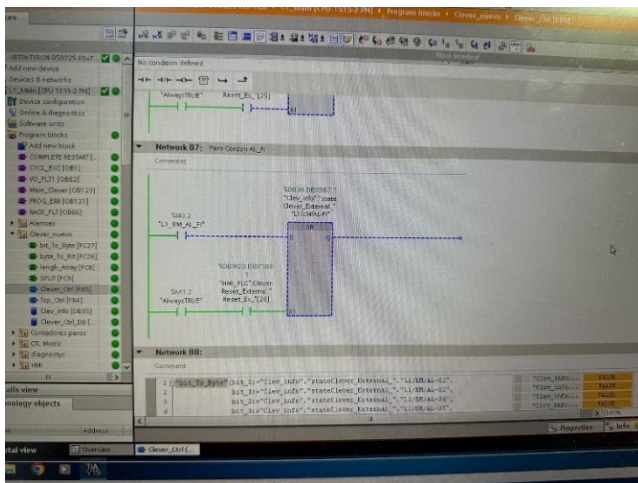


Ilustración 14.- Carga del proyecto de línea 1

4. Pruebas de funcionamiento



Se estableció la comunicación entre la computadora y el PLC Siemens S7-1500, se verificó la correcta lectura de los DB desde TIA Portal y se confirmó la sincronización con las señales de paro internas y externas.

Ilustración 15.- Conexión exitosa del PLC

Instalación y configuración de Python



Ilustración 16.- Logo de python

Python es un lenguaje de programación de propósito general, interpretado, de alto nivel y con una semántica dinámica. Es ampliamente utilizado en desarrollo web, análisis de datos, inteligencia artificial y automatización, entre otros.

1. Primero se descarga el instalador desde la página oficial de Python.

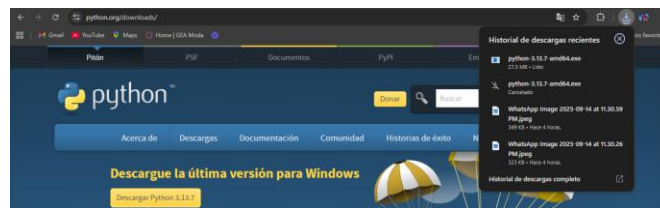


Ilustración 17.- Descarga de python

2. Se marca la casilla "Add Python to PATH" en la primera ventana del instalador y se da clic en "Install Now" para iniciar la instalación.

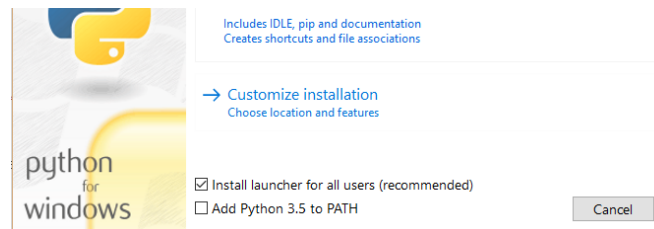


Ilustración 18.- Instalar PATH

3. Se abre el Símbolo del Sistema (CMD) y escribimos python --version para confirmar la instalación.

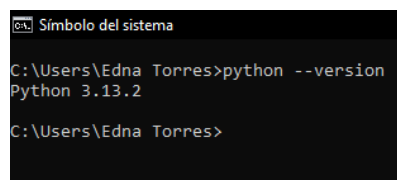


Ilustración 19.- Instalación exitosa de Python

Instalación y configuración de Visual Studio Code



Ilustración 20.- Logo Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente ligero pero potente, desarrollado por Microsoft. Ofrece soporte integrado para depuración, control de versiones Git, resaltado de sintaxis, autocompletado de código inteligente y una vasta colección de extensiones.

1. Descargar el instalador desde la página oficial de Visual Studio Code.

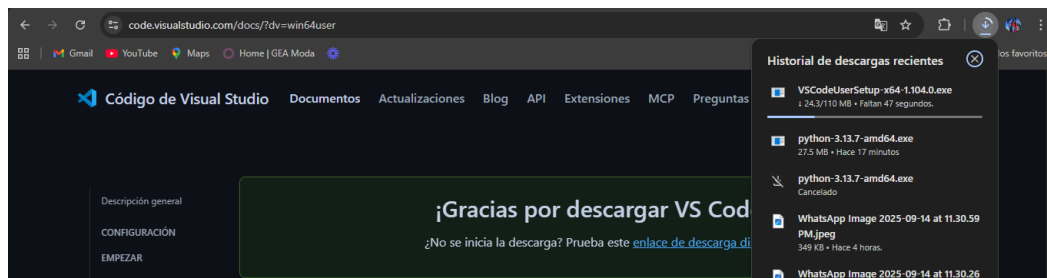


Ilustración 21.- Descarga de Visual Studio Code

2. Aceptar el acuerdo de licencia.

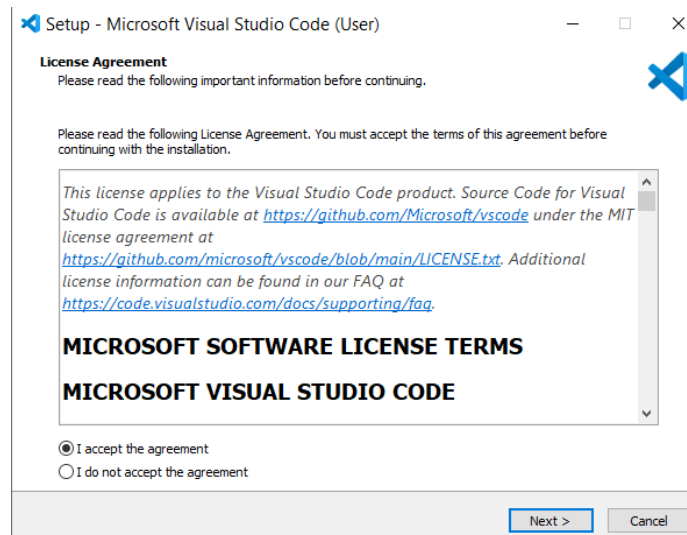


Ilustración 22.- Acuerdo de licencia de Visual Studio Code

3. Marcar las opciones recomendadas en la ventana de Tareas Adicionales, especialmente las que añaden la acción "Abrir con Code" al menú contextual, hacer clic en "Instalar" y esperar a que el proceso finalice.

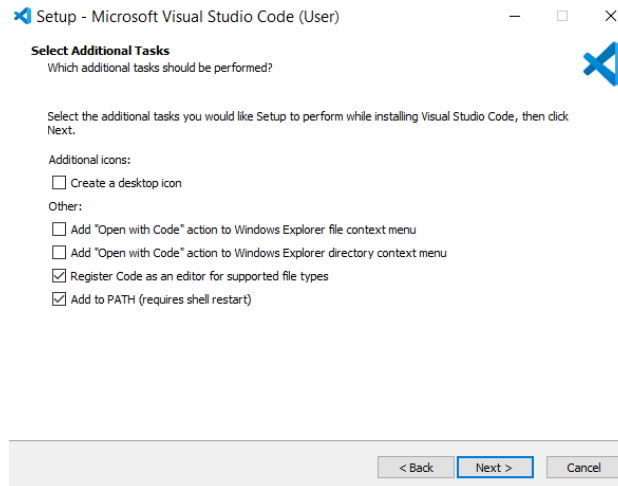


Ilustración 23.- Tareas adicionales de Visual Studio Code

4. Abrir Visual Studio Code, ir a la sección de Extensiones (el icono de cuatro cuadrados) y buscar e instalar la extensión "Python" de Microsoft.



Ilustración 24.- Sección de Extensiones (Python)

5. Abrir la Paleta de Comandos (Ctrl + Shift + P), escribir "Python: Select Interpreter" and seleccionar la versión de Python que se instaló previamente.

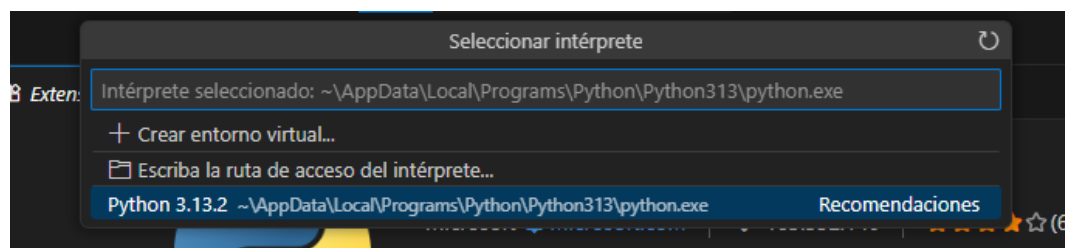


Ilustración 25.- Intérprete Python

Organización del proyecto en Visual Studio Code

Tras la instalación de las herramientas necesarias para el desarrollo, se ha configurado la estructura del proyecto en Visual Studio Code. Esta organización busca una gestión clara y eficiente del código, clasificando los archivos por su función dentro de carpetas específicas.

La estructura se compone de las siguientes carpetas y archivos principales:

- **templates:** Contiene los archivos **HTML** del proyecto, como `data_analysis.html`, `index.html` y `resumen.html`. Esta carpeta se usa para separar la lógica de programación del contenido de la interfaz de usuario.
- **static:** Almacena todos los archivos estáticos necesarios para la web, incluyendo hojas de estilo **CSS** (`styles_data_analysis.css`, etc.) e imágenes (`logo_motherson.png`).
- Los archivos principales del proyecto, escritos en **Python**, se encuentran en la raíz del directorio (`app.py`, `plc.py`, `stations_config.py`, `utils.py`), lo que facilita el acceso a la lógica central de la aplicación.
- Además, se encuentran carpetas como **instance**, que contiene la base de datos `paros.db`, y **__pycache__**, utilizada por Python para guardar archivos de código compilado.

Esta organización asegura que cada tipo de archivo se encuentre en su lugar correspondiente, facilitando la colaboración, el mantenimiento y la escalabilidad del proyecto.

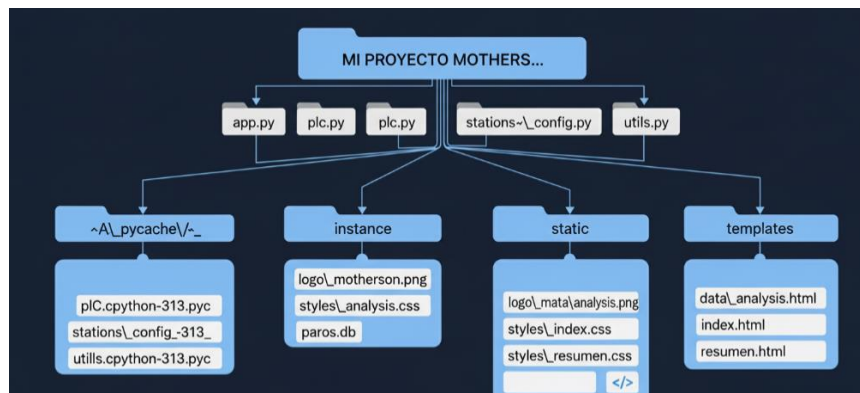


Ilustración 26.- Estructura de archivos y carpetas

Conexión del Web Server con el PLC mediante Snap7

```
1 import snap7
2
3 # Dirección IP del PLC
4 PLC_IP = '192.168.0.1' # Cambia esta IP por la del PLC
5 RACK = 0 # RACK del PLC, normalmente 0
6 SLOT = 1 # SLOT del PLC, normalmente 1 para los PLCs S7-
7
8 # Crear la conexión con el PLC
9 client = snap7.client.Client()
10
11 # Conectar al PLC
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** PUERTOS

PS C:\Users\Edna Torres\Desktop\Paros Externos L1\Paros Externos - L1> a Torres/Desktop/import datetime.py"
Conexión exitosa al PLC.

Para establecer la comunicación entre el PLC Siemens S7-1500 y el Web Server, se implementó un módulo en Python denominado `plc.py`. Este archivo utiliza la librería Snap7, la cual permite leer y escribir datos directamente en los bloques de memoria del PLC a través del protocolo S7Comm sobre TCP/IP.

Ilustración 27.- Conexión mediante Snap7

El objetivo principal de esta conexión es lograr que el servidor web desarrolle una comunicación en tiempo real con el PLC, garantizando la correcta captura de los estados de los paros de línea y permitiendo la escritura de valores cuando es necesario restablecer la operación.

Explicación del archivo `plc.py`

El archivo está diseñado como una clase llamada PLC, que centraliza toda la lógica de comunicación. Sus funciones principales son:

1. Establecer la conexión con el PLC

La clase recibe como parámetros la IP, rack y slot del PLC y, mediante el método `_conectar()`, se encarga de iniciar la comunicación.

```

29 def _conectar(self):
30     """Intenta conectar al PLC. Debe ser llamado dentro del lock."""
31
32     if self._conectado:
33         return True
34
35     try:
36         logger.info(f"Intentando conectar al PLC en {self.ip}:{self.rack}:{self.slot}...")
37         self.cliente.connect(self.ip, self.rack, self.slot)
38         self._conectado = self.cliente.get_connected()
39         if self._conectado:
40             logger.info(f"Conectado exitosamente al PLC en {self.ip}")
41             return True
42         else:
43             logger.warning(f"No se pudo conectar al PLC en {self.ip}. Estado de conexión: {self.cliente.get_connected()}")
44             return False
45     except RuntimeError as e:
46         logger.error(f"Error de ejecución Snap7 al conectar al PLC en {self.ip}:{self.rack}:{self.slot}: {e}")
47         self._conectado = False
48         return False
49     except Exception as e:
50         logger.error(f"Error inesperado durante la conexión al PLC en {self.ip}:{self.rack}:{self.slot}: {e}")
51         self._conectado = False
52         return False

```

Ilustración 28.-Establecer conexión con el PLC

- Intenta conectarse al PLC.
- Si falla, registra el error en log y marca la conexión como desconectada.

2. Mantener la conexión activa automáticamente

El sistema utiliza un **hilo en segundo plano** que ejecuta el método `_loop_reconexion()`. Este bucle comprueba si el PLC sigue conectado y, si no lo está, intenta reconectar cada 5 segundos (configurable).

```

22 self._hilo_reconexion = threading.Thread(target=self._loop_reconexion, daemon=True)
23 self._hilo_reconexion.start()

```

Ilustración 29.- Mantener la conexión activa automáticamente con el PLC

```

67 def _loop_reconexion(self):
68     """Bucle para intentar reconectar al PLC periódicamente."""
69
70     with self._lock_conexion:
71         self._conectar()
72
73     while not self._evento_parar.is_set():
74         with self._lock_conexion:
75             if not self._conectado:
76                 self._conectar()
77         self._evento_parar.wait(self.intervalo_reconexion)

```

Ilustración 30.- Mantener la conexión activa automáticamente con el PLC

- `_hilo_reconexion` se lanza automáticamente al crear la clase.
- `_loop_reconexion` revisa constantemente si el PLC está conectado.
- Si detecta una desconexión, intenta reconectar de manera automática.

3. Lectura de valores booleanos

El método **leer_bool()** permite obtener el estado de una variable booleana desde el PLC y así poder ver si una estación generó un paro.

```
79 def leer_bool(self, db_numero, byte, bit):
80     """Lee un valor booleano del PLC."""
81     with self._lock_conexion:
82         if not self._conectado:
83             logger.warning("El PLC no está conectado. No se puede leer.")
84             raise ConnectionError("PLC no conectado o conexión perdida.")
85         try:
86             datos = self.cliente.db_read(db_numero, byte, 1)
87             return get_bool(datos, 0, bit)
88         except RuntimeError as e:
89             logger.error(f"Error de ejecución Snap7 al leer DB{db_numero}.DBB{byte}.X{bit}: {e}")
90             self._desconectar()
91             raise ConnectionError(f"Error de lectura PLC (Snap7): {e}")
92         except Exception as e:
93             logger.error(f"Error inesperado al leer DB{db_numero}.DBB{byte}.X{bit}: {e}")
94             self._desconectar()
95             raise ConnectionError(f"Error de lectura PLC (inesperado): {e}")
96
```

Ilustración 31.- Lectura de valores booleanos

- Accede a un **DB del PLC** para leer un bit específico.
- Devuelve **True** o **False** según el valor en el PLC.

4. Escritura de valores booleanos

El método **escribir_bool()** permite modificar el valor de una variable booleana dentro del PLC. Esto es útil para **resetear un paro externo desde la interfaz web**.

```
97 def escribir_bool(self, db_numero, byte, bit, valor):
98     """Escribe un valor booleano en el PLC."""
99     with self._lock_conexion:
100         if not self._conectado:
101             logger.warning("El PLC no está conectado. No se puede escribir.")
102             raise ConnectionError("PLC no conectado o conexión perdida.")
103         try:
104             datos = self.cliente.db_read(db_numero, byte, 1)
105             set_bool(datos, 0, bit, valor)
106             self.cliente.db_write(db_numero, byte, datos)
107             logger.info(f"Escrito {valor} en DB{db_numero}.DBB{byte}.X{bit}")
108         except RuntimeError as e:
109             logger.error(f"Error de ejecución Snap7 al escribir en DB{db_numero}.DBB{byte}.X{bit}: {e}")
110             self._desconectar()
111             raise ConnectionError(f"Error de escritura PLC (Snap7): {e}")
112         except Exception as e:
113             logger.error(f"Error inesperado al escribir en DB{db_numero}.DBB{byte}.X{bit}: {e}")
114             self._desconectar()
115             raise ConnectionError(f"Error de escritura PLC (inesperado): {e}")
116
```

Ilustración 32.- Escritura de valores booleanos

- Lee el byte donde está el bit, lo modifica y lo escribe de nuevo.
- Permite **activar/desactivar paros** desde el web server.

5. Control seguro de la conexión

Se implementan métodos auxiliares para verificar y cerrar la conexión de forma ordenada.

```

117 def esta_conectado(self):
118     """Verifica el estado de conexión del PLC."""
119     with self._lock_conexion:
120         return self._conectado
121
122 def detener(self):
123     """Detiene el hilo de reconexión y desconecta el PLC."""
124     logger.info("Deteniendo la conexión al PLC y el hilo de reconexión.")
125     self._evento_parar.set()
126     if self._hilo_reconexion.is_alive():
127         self._hilo_reconexion.join(timeout=self._intervalo_reconexion + 2)
128     self._desconectar()
129     logger.info("Conexión al PLC detenida.")

```

Ilustración 33.- Control seguro de la conexión

- `esta_conectado()` devuelve el estado actual (conectado o no).
- `detener()` cierra el hilo de reconexión y libera los recursos del cliente.

Con esta estructura el archivo garantiza la conexión automática y robusta con el PLC, la lectura y escritura de las variables booleanas en tiempo real y sobre todo el manejo seguro y recursos al desconectar.

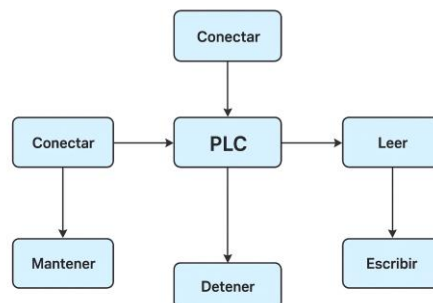


Ilustración 34.- Diagrama de como interactúan los métodos

Configuración de estaciones y definición de paros en el sistema

Para centralizar y organizar de manera eficiente la configuración de los paros en la línea de producción, se separaron los booleanos de las estaciones externas e internas en un archivo independiente llamado `station_config.py`. Esto permite mantener el código más limpio, facilita su mantenimiento y evita duplicaciones al gestionar los estados de paro de cada estación.

En este archivo se definen:

- **PAROS_EXTERNOS:** Lista de todos los paros externos que requieren un control de activación y reset desde el sistema. Estas son las estaciones que pueden ser detenidas y reiniciadas desde el sistema de manera controlada a excepción del "ECHECK":

```
1  PAROS_EXTERNOS = [  
2      "AL_01", "AL_02", "AL_04", "AL_05", "AL_06", "AL_07", "AL_08",  
3      "AL_10", "AL_12", "AL_13", "AL_15", "AL_16", "AL_17", "AL_18",  
4      "AL_19", "AL_20", "AL_21", "AL_22", "AL_23", "AL_24", "AL_25",  
5      "AL_26", "AL_27", "AL_28", "AL_29", "AL_30", "AL_31", "AL_32",  
6      "AL_33", "AL_36", "AL_FI", "ECHECK"  
7  ]
```

Ilustración 35.- Paros Internos

- **PAROS_INTERNOS:** Lista de paros internos, que generalmente no cuentan con un reset automático. Estas son las estaciones cuya detención se produce de manera interna en la línea y requiere identificación de los bits activos:

```
9  PAROS_INTERNOS = [  
10     "P1_AL01", "STOP_02", "T1_AP03", "HV_AL03", "DR_AL05", "T3_AL06", "DF_AL07", "N6_AL10",  
11     "P3_AL09", "STOP_10", "TD_AP12", "KA_AL11", "STOP_13", "STOP_14", "N9_AL14", "STOP_16",  
12     "STOP_17", "STOP_18", "STOP_19", "STOP_20", "STOP_21", "STOP_22", "STOP_23", "STOP_24",  
13     "STOP_25", "STOP_26", "STOP_27", "STOP_28", "M2_AL28", "STOP_30", "STOP_31", "STOP_32",  
14     "STOP_33", "STOP_34", "STOP_35", "STOP_36", "OC_FI", "STOP_38", "STOP_39", "STOP_40",  
15     "STOP_41", "STOP_42", "STOP_43", "CR_FI", "STOP_45",  
16 ]
```

Ilustración 36.- Paros Externos

- **ESTACIONES:** Diccionario que asocia cada estación con sus direcciones de bits en el PLC para el paro (paro) y, cuando aplica, para el reset (reset).

```

18  ESTACIONES = {}
19      "AL_01": {"paro": (35, 364, 0), "reset": (900, 180, 0)},
20      "AL_02": {"paro": (35, 364, 1), "reset": (900, 180, 1)},
21      "AL_04": {"paro": (35, 364, 2), "reset": (900, 180, 2)},
22      "AL_05": {"paro": (35, 364, 3), "reset": (900, 180, 3)},
23      "AL_06": {"paro": (35, 364, 4), "reset": (900, 180, 4)},
24      "AL_07": {"paro": (35, 364, 5), "reset": (900, 180, 5)},
25      "AL_08": {"paro": (35, 364, 6), "reset": (900, 180, 6)},
26      "AL_10": {"paro": (35, 364, 7), "reset": (900, 180, 7)},

```

Ilustración 37.- Estaciones con reset

```

53      "P1_AL01": {"paro": (35, 354, 0), "reset": None}, #01 P1_AL01
54      "STOP_02": {"paro": (35, 354, 1), "reset": None}, #02
55      "T1_AP03": {"paro": (35, 354, 2), "reset": None}, #03 T1_AP03
56      "HV_AL03": {"paro": (35, 354, 3), "reset": None}, #04 HV_AL03
57      "DR_AL05": {"paro": (35, 354, 4), "reset": None}, #05 DR_AL05
58      "T3_AL06": {"paro": (35, 354, 5), "reset": None}, #06 T3_AL06
59      "DF_AL07": {"paro": (35, 354, 6), "reset": None}, #07 DF_AL07
60      "STOP_08": {"paro": (35, 354, 7), "reset": None}, #08
61      "P3_AL09": {"paro": (35, 355, 0), "reset": None}, #09 P3_AL09
62      "N6_AL10": {"paro": (35, 355, 1), "reset": None}, #10 N6_AL10

```

Ilustración 38.- Estaciones sin reset

Procedimiento para agregar una estación

Para agregar una estación en el diccionario ESTACIONES, se siguen los siguientes pasos:

1. Definir el nombre de la estación:

- Se elige un nombre único y descriptivo, normalmente coincidente con el nombre físico de la estación en la línea de producción.
- Ejemplo: "AL_01".

2. Asignar la dirección del bit de paro (paro):

- Se define como una tupla (byte, word, bit) que indica la ubicación del bit de paro en el PLC.
- Ejemplo: (35, 364, 0) significa byte 35, palabra 364, bit 0.

3. Asignar la dirección del bit de reset (reset):

- Se define de igual manera como una tupla (byte, word, bit) que indica la ubicación del bit de reset en el PLC, si la estación cuenta con reset.
- Si no tiene reset, se asigna None.
- Ejemplo: (900, 180, 0) o None.

4. Agregar la estación al diccionario:

- Se crea una entrada en ESTACIONES con la siguiente estructura:

```
"AL_01": {"paro": (35, 364, 0), "reset": (900, 180, 0)},
```

Ilustración 39.- Ejemplo de paro con reset

```
"P1_AL01": {"paro": (35, 354, 0), "reset": None},
```

Ilustración 40.- Ejemplo de paro sin reset

Este procedimiento permite mantener un control claro y confiable de cada estación, asegurando que los bits de paro y reset correspondan correctamente a la configuración del PLC.

Verificación y corrección de paros internos y externos en TIA Portal

Durante la integración del sistema de monitoreo, se identificó que los nombres de los paros internos y externos en el TIA Portal no coincidían correctamente con la realidad de la línea de producción. Por esta razón, se realizaron ajustes y pruebas detalladas para garantizar la correspondencia entre el sistema y el PLC.

Para los paros externos, se reorganizó la nomenclatura de acuerdo con el orden definido en el sistema de monitoreo, verificando que cada estación tuviera asignado el bit correcto de paro y de reset en el TIA Portal.

En el caso de los paros internos, se provocaron intencionalmente paros uno por uno para identificar correctamente el nombre de cada estación y confirmar qué bits realmente estaban activos. Esto fue necesario porque, aunque en el TIA Portal se definieron 80 paros internos, no todas las estaciones cuentan con este tipo de paro. De esta manera, se pudo establecer cuáles bits correspondían a estaciones activas y cuáles no, asegurando que la información registrada en el sistema fuera confiable y consistente.

Este proceso de verificación permitió alinear completamente la configuración del TIA Portal con la estructura del sistema de monitoreo, garantizando que los paros internos y externos sean correctamente detectados, registrados y reiniciados según corresponda.

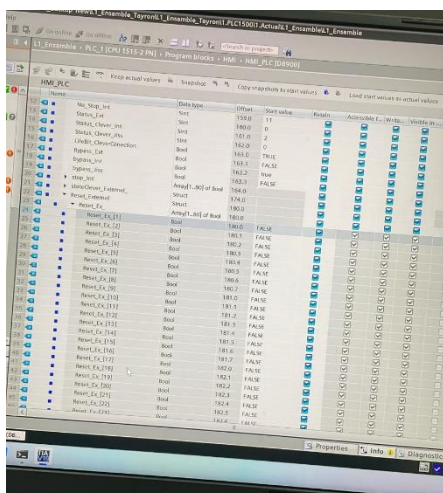


Ilustración 41.- Lista de paros en TIA Portal

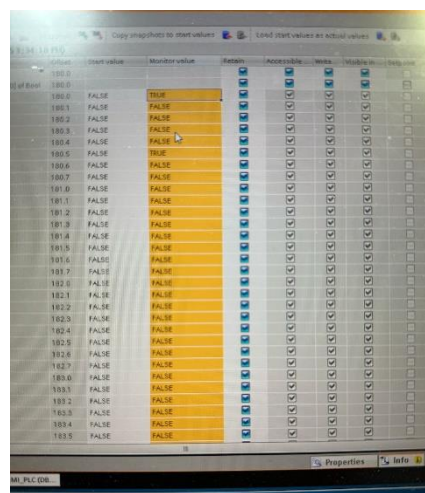


Ilustración 42.- Lista de reset en TIA Portal

Arquitectura y funcionamiento del archivo principal app.py

El archivo **app.py** constituye el núcleo del sistema, ya que integra la comunicación con el PLC Siemens S7-1500, la gestión de la base de datos y la visualización de la información a través de una interfaz web desarrollada con Flask. Su función principal es actuar como puente entre el hardware de control industrial y los usuarios finales, quienes interactúan con el sistema mediante navegadores web o tablets.

Dentro de este archivo se encuentran los elementos esenciales del proyecto:

- **Inicialización del servidor Flask**, que permite crear la aplicación web.
- **Conexión con el PLC**, utilizando la clase PLC del archivo plc.py para leer y escribir variables de proceso.
- **Gestión de la base de datos SQLite**, destinada al registro de incidentes de paro (inicio, fin, duración, causa y estación).
- **Definición de rutas web**, que permiten la navegación y la interacción del usuario (visualización de estados, registro de paros, historial y reseteo de señales).
- **Renderizado de la interfaz gráfica**, empleando plantillas HTML, CSS y JavaScript, para mostrar en tiempo real los estados de cada estación y facilitar la interacción de los operadores.

En conjunto, este archivo asegura que la información capturada en el PLC sea procesada, almacenada y presentada de forma clara y accesible, garantizando así la trazabilidad de los paros y apoyando la toma de decisiones en la línea de producción.

Configuración inicial del archivo app.py

La primera parte del archivo app.py prepara el entorno de la aplicación, importando las librerías necesarias, configurando el servidor Flask, definiendo la conexión con el PLC y estableciendo la base de datos.

1. Importación de librerías

```
1  from flask import Flask, render_template, jsonify, request, current_app, send_file
2  from plc import PLC
3  from stations_config import PAROS_EXTERNOS, PAROS_INTERNOS, ESTACIONES
4  from utils import segundos_a_hms
5  import time
6  import threading
7  from collections import defaultdict
8  from datetime import datetime, timedelta
9  import logging
10 from flask_sqlalchemy import SQLAlchemy
11 import pytz
12 import os
13 import pandas as pd
14 import io
```

Ilustración 43.- Importación de librerías

- **Flask:** framework principal para construir el servidor web.
- **PLC:** clase importada de plc.py para la conexión con el PLC Siemens.
- **stations_config:** contiene listas y configuraciones de estaciones y tipos de paros.
- **utils:** funciones auxiliares como conversión de segundos a formato de horas, minutos y segundos.
- **logging:** para registrar eventos y errores del sistema.
- **SQLAlchemy:** ORM para manejar la base de datos SQLite.
- **pytz:** manejo de zona horaria.
- **pandas y io:** permiten generar reportes y manejar datos en formato Excel/CSV.

2. Configuración de logging

```
17 logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
18 logger = logging.getLogger(__name__)
```

Ilustración 44.- Configuración de logging

Permite registrar eventos importantes como conexiones al PLC, errores y operaciones de la base de datos, lo que facilita la depuración y el monitoreo del sistema.

3. Inicialización de la aplicación Flask

```
19
20 app = Flask(__name__) |
21
```

Ilustración 45.- Inicialización de la aplicación Flask

Se crea la instancia principal de la aplicación Flask, que será el servidor web encargado de gestionar las rutas y mostrar las interfaces al usuario.

4. Configuración de la base de datos con SQLAlchemy

```
23 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///paros.db'
24 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
25 db = SQLAlchemy(app)
```

Ilustración 46.- Configuración de la base de datos con SQLAlchemy

- Se usa **SQLAlchemy** para interactuar con la base de datos.
- El motor configurado es **SQLite**, con el archivo paros.db.
- Esto permite definir modelos de datos en Python y usarlos directamente en la base de datos sin necesidad de consultas SQL manuales.

5. Definición de la zona horaria

```
28 TIMEZONE = 'America/Mexico_City'
29 local_timezone = pytz.timezone(TIMEZONE)
30
```

Ilustración 47.- Definición de la zona horaria

Asegura que todos los registros (inicio, fin y duración de paros) se almacenen y gestionen en la zona horaria correcta.

6. Definición de los modelos de datos

```
32 class Paro(db.Model):
33     id = db.Column(db.Integer, primary_key=True)
34     estacion = db.Column(db.String(80), nullable=False)
35     tipo_paro = db.Column(db.String(20), nullable=False)
36     inicio = db.Column(db.DateTime, nullable=False)
37     fin = db.Column(db.DateTime, nullable=True)
38     duracion_segundos = db.Column(db.Integer, default=0)
39     causa = db.Column(db.String(200), nullable=True)
40     area = db.Column(db.String(80), nullable=True)
41     shift_day_id = db.Column(db.String(80), nullable=False)
```

Ilustración 48.- Definición de los modelos de datos (Paro)

El modelo **Paro** representa un registro de paro en la línea: incluye la estación, el tipo (interno o externo), la hora de inicio y fin, duración y la causa.

```
46 class Shift(db.Model):
47     id = db.Column(db.Integer, primary_key=True)
48     shift_name = db.Column(db.String(50), nullable=False, unique=True)
49     start_time = db.Column(db.Time, nullable=False)
50     end_time = db.Column(db.Time, nullable=False)
```

Ilustración 49.- Definición de los modelos de datos (turno)

El modelo **Shift** define los tres turnos de la planta (horarios de inicio y fin), lo que permite organizar los paros registrados según el turno en que ocurrieron.}

7. Creación de tablas y carga inicial

```
56 with app.app_context():
57     db.create_all()
58     if not Shift.query.first():
59         db.session.add(Shift(shift_name="Turno 1", start_time=datetime.strptime("06:00:00", "%H:%M:%S").time(), end_time=datetime.strptime("14:59:59", "%H:%M:%S").time()))
60         db.session.add(Shift(shift_name="Turno 2", start_time=datetime.strptime("15:00:00", "%H:%M:%S").time(), end_time=datetime.strptime("23:29:59", "%H:%M:%S").time()))
61         db.session.add(Shift(shift_name="Turno 3", start_time=datetime.strptime("23:30:00", "%H:%M:%S").time(), end_time=datetime.strptime("05:59:59", "%H:%M:%S").time()))
62     db.session.commit()
63     logger.info("Turnos inicializados en la base de datos.")
```

Ilustración 50.- Creación de tablas y carga inicial

- Se crean automáticamente las tablas paros y shifts en la base de datos si no existen.
- Si no hay turnos registrados, se insertan por defecto los tres turnos definidos para la planta.

8. Conexión con el PLC

```
66 plc = PLC(ip="10.246.150.139", rack=0, slot=1)
67
```

Ilustración 51.- Conexión con el PLC

Crea la instancia de la clase PLC definida en plc.py, lo que permite que el servidor web lea y escriba datos directamente en el PLC Siemens S7-1500.

Con esta configuración inicial, app.py establece:

- El servidor web con Flask.
- La base de datos con SQLAlchemy.
- La definición de modelos para registrar paros y turnos.
- La conexión directa al PLC, asegurando la integración hardware–software.

Definición de rutas principales en app.py

Las rutas son los **endpoints** de la aplicación Flask, encargados de conectar la lógica del servidor con la interfaz web y la base de datos. Cada ruta cumple una función específica, desde mostrar la página principal hasta generar reportes en Excel.

1. Página principal

```
234 @app.route('/')
235 def index():
236     dt_now_local = datetime.now(pytz.utc).astimezone(local_timezone)
237     display_shift_name = current_shift_day_id.split(' - ')[1] if current_shift_day_id else "Cargando..."
238     return render_template('index.html', current_shift=display_shift_name)
239
```

Ilustración 52.- Ruta página principal

- Renderiza **index.html**, mostrando la página principal del sistema.

- Calcula la hora local y determina el turno actual para mostrarlo en pantalla.
- Permite que el operador vea en qué turno está trabajando y acceder al monitoreo en tiempo real.

2. Estado de estaciones

```

240 @app.route("/status")
241 def status():
242     stations_data = {}
243     total_paros_externos = 0
244     tiempo_total_externos = 0
245     total_paros_internos = 0
246     total_tiempo_internos = 0
247     paros_actuales_duracion = {}
248

```

Ilustración 53.- Ruta estado de estaciones

- Devuelve un **JSON** con información detallada de cada estación:
 - Si está en paro (en_paro).
 - Tiempo actual del paro.
 - Número de paros acumulados en el turno.
 - Tiempo total acumulado.
- También identifica:
 - La estación con más paros.
 - La estación con mayor tiempo detenido.
 - Totales de paros **internos** y **externos**.
 - Si hay paros externos sin causa y si se debe bloquear el reset.

3. Reset de estaciones

```

327 @app.route("/reset/<station>", methods=["POST"])
328 def reset(station):
329     global paros_externos_sin_causa_consecutivos
330     if station in PAROS_EXTERNOS and paros_externos_sin_causa_consecutivos >= MAX_PAROS_SIN_CAUSA:
331         return jsonify({"error": "El reset de paros externos está bloqueado por tener demasiados paros sin causa."}), 403
332
333     data = request.get_json()
334     estado = data.get("estado")

```

Ilustración 54.- Ruta reset de estaciones

- Permite enviar una orden de **reset al PLC** desde la web.

- Verifica si la estación tiene configurada la opción de reset.
- Controla si el reset debe bloquearse en caso de paros externos sin causa.
- Responde con un mensaje de éxito o error en JSON

4. Resumen de paros

```

350 @app.route("/resumen")
351 def resumen():
352     with app.app_context():
353         # Obtener todos los paros del turno y día actual
354         paros_para_mostrar = Paro.query.filter(
355             Paro.shift_day_id == current_shift_day_id
356         ).order_by(Paro.inicio.desc()).all()
357
358
359         estadisticas_paros_summary = defaultdict(lambda: {"num_paros": 0, "tiempo_total": 0})
360         total_num_paros_internos = 0
361         total_tiempo_internos_segundos = 0
362         total_num_paros_externos = 0
363         total_tiempo_externos_segundos = 0

```

Ilustración 55.- Ruta resumen de paros

- Recupera todos los paros del turno en curso desde la base de datos.
- Calcula estadísticas: número de paros y tiempo total por estación, diferenciando internos y externos.
- Muestra en **resumen.html**:
 - Tabla de historial con inicio, fin, duración y causa.
 - Totales de paros internos y externos.
 - Tiempo acumulado por categoría.

5. Guardar causa de paro

```

417 @app.route("/guardar_causa_resumen", methods=["POST"])
418 def guardar_causa_resumen():
419     global algún_paro_externo_sin_causa, paros_externos_sin_causa_consecutivos
420     data = request.json
421     estacion = data.get('estacion')
422     inicio_str = data.get('inicio')
423     causa_completa = data.get('causa')
424     tipo_paro = data.get('tipo')

```

Ilustración 56.- Ruta guardar causa de paro

- Permite registrar desde la interfaz la **causa de un paro**.
- Actualiza la base de datos con la causa y área correspondiente.
- Si es un paro externo distinto de ECHECK, también clasifica el área productiva afectada.
- Restablece el contador de paros sin causa para habilitar nuevamente el reset.

6. Página de análisis de datos

```
502 @app.route('/data_analysis')
503 def data_analysis():
504     return render_template('data_analysis.html')
505
```

Ilustración 57.- Ruta página de análisis de datos

- Renderiza la página de análisis (data_analysis.html).
- Desde esta sección se muestran gráficas y estadísticas sobre las principales causas y áreas críticas.

7. Análisis de tiempos

```
580 @app.route('/get_analysis_data')
581 def get_analysis_data():
582     data = calcular_tiempos_de_paro()
583     return jsonify(data)
584
```

Ilustración 58.- Ruta Análisis de tiempo

- Ejecuta la función `calcular_tiempos_de_paro()` que procesa todos los registros del turno.
- Devuelve datos en JSON para graficar en la interfaz:
 - Tiempo por área.
 - Principales causas externas.
 - Causas internas.
 - Paros en ECHECK.

8. Exportar a Excel

```
585 @app.route('/download_shift_data_excel')
586 def download_shift_data_excel():
587     with app.app_context():
588         # Obtener el shift_day_id actual
589         shift_id = current_shift_day_id
590         if not shift_id:
591             return jsonify({"error": "No hay un turno activo para descargar datos."}), 400
592
593         # --- 1. Datos de Paros Totales y Tiempo General (Resumen de Paros) ---
594         paros_del_turno = Paro.query.filter_by(shift_day_id=shift_id).all()
```

Ilustración 59.- Ruta exportar Excel

- Genera un archivo **Excel dinámico** con toda la información del turno:
 - Historial detallado.
 - Resumen general.
 - Estadísticas por estación.
 - Tiempos y causas por área.
- Permite descargar el reporte completo para análisis externo.

Con estas rutas, app.py cumple tres funciones clave:

1. **Interacción en tiempo real** con el PLC y la base de datos.
2. **Gestión de paros**: registro, consulta, reseteo y clasificación de causas.
3. **Reportes y análisis**: visualización en la web y exportación en Excel.

Funcionamiento de la Base de Datos en el Sistema de Monitoreo

La base de datos está estructurada con un modelo principal llamado **Paro**, que representa cada evento de paro registrado en la línea de producción. Este modelo incluye los siguientes campos esenciales:

- **estacion** → identifica la estación donde ocurrió el paro.
- **tipo_paro** → indica si el paro es **Interno (Clever)** o **Externo (Andon)**.
- **inicio y fin** → registran la hora exacta de inicio y fin del paro.

- **duracion_segundos** → almacena automáticamente la duración total calculada.
- **causa** y **area** → permiten clasificar los paros según el motivo y el área responsable.
- **shift_day_id** → relaciona el paro con el turno y el día en que ocurrió.

De esta forma, cada fila en la base de datos equivale a un evento único de paro, trazable desde su inicio hasta su resolución.

1. Creación de registros

Cuando el sistema detecta que una estación entra en estado de paro (a través del PLC), se genera un **nuevo registro en la base de datos**:

```
paro_nuevo = Paro(
    estacion=station,
    tipo_paro="Externo" if station in PAROS_EXTERNOS else "Interno",
    inicio=datetime.now(local_timezone),
    shift_day_id=current_shift_day_id
)
db.session.add(paro_nuevo)
db.session.commit()
```

Ilustración 60.- Creación de registros

Esto asegura que cada paro queda registrado con su hora de inicio y la identificación de la estación.

2. Actualización de registros

```
paro_existente.fin = datetime.now(local_timezone)
paro_existente.duracion_segundos = (paro_existente.fin - paro_existente.inicio).total_seconds()
db.session.commit()
```

Ilustración 61.- Actualización de registros

De esta manera, el sistema calcula y guarda automáticamente la **duración del paro**.

3. Asignación de causa y área

Los operadores, desde las tablets, asignan la causa del paro a través de la interfaz web. El sistema busca el paro en la base de datos y actualiza los campos correspondientes:

```
paro_to_update.causa = causa_completa
paro_to_update.area = area_extraida
db.session.commit()
```

Ilustración 62.- Asignación de causa y área

Esto enriquece la información con datos de análisis que permiten identificar las **principales causas de ineficiencia**.

4. Consultas y análisis

El sistema consulta los datos registrados para generar:

- **Tablas históricas** de todos los paros del turno actual (resumen.html).
- **Estadísticas** como número total de paros por estación, tiempo acumulado y áreas más afectadas.
- **Gráficas de análisis** (data_analysis.html) que muestran causas más frecuentes y distribución de tiempos.
- **Exportación a Excel** con reportes completos para cada turno.

Definición y establecimiento de causas en resumen.html

Durante el desarrollo del sistema se llevó a cabo una junta con personal de diferentes áreas de la planta, con el objetivo de definir un catálogo oficial de causas de paro. Esta actividad fue fundamental para estandarizar la clasificación de

incidencias y garantizar que la información registrada en el sistema fuera clara, precisa y útil para el análisis posterior.

Una vez consensuadas las causas, estas se integraron directamente en el archivo resumen.html, mediante menús desplegables que permiten al operador seleccionar el área, la causa y, en algunos casos, especificaciones adicionales. Asimismo, se incorporaron cuadros de texto para introducir datos libres en situaciones no contempladas, asegurando así la flexibilidad del sistema.

Causas para paros externos (Andon)

```
"Logistica": {
  "Material equivocado": [],
  "Material faltante": ["Boornerezt", "Columna de dirección ", "Molduras", "Preseries"],
  "Material dañado ": ["Caja relex rota", "Caja clima "],
  "Otro": []
},
"Ingenieria": {
  "Falla de herramienta": [],
  "Falla de equipo": ["echeck", "Picktollight"],
  "Falla de clever (configuración)": [],
  "Falta de dispositivo ": [],
  "Autospreserie": [],
  "Cambio de modelo erroneo ": [],
  "Otro": []
},
"Calidad": {
  "Provedor": [],
  "Herramientas y dispositivos (calibración)": [],
  "Otro": []
},
},
```

Ilustración 63.- Causas para paros externos

```
"Produccion": {
  "Atraso de operación": [],
  "Capacitación": [],
  "Ensamble equivocado": [],
  "Omisión de proceso": [],
  "Falta de personal": [],
  "Otro": []
},
"Mantenimiento": {
  "Falla de cadena": [],
  "Falla de equipo ": [],
  "Falla de motriz": [],
  "Falla de jis": [],
  "Otro": []
},
"Sistemas": {
  "Falla de hardware (pantallas, escaner, impresoras, cables)": [],
  "Falla de clever": [],
  "Falla de red": [],
  "Otro": []
},
"Cliente": {
  "Mesa llena": [],
  "Falta de bastidores": [],
  "Sin señal": [],
  "Otro": []
},
},
```

Ilustración 64.- Causas para paros externos 2

Causas paros internos (clever)

```
<option value="tiempo expirado">tiempo expirado</option>
<option value="cockpit incompleto">cockpit incompleto</option>
<option value="error de secuencia">error de secuencia</option>
<option value="ventana nok">ventana nok</option>
<option value="Otro">Otro</option>
```

Ilustración 65.- Causas paros internos

Causas paros de ECHECK

```
<option value="prueba nok">prueba nok</option>
<option value="retrabajo">retrabajo</option>
<option value="falla de equipo">falla de equipo</option>
<option value="Otro">Otro</option>
```

Ilustración 66.- Causas paros de ECHECK

Para garantizar un mejor control y trazabilidad, se agregaron **cuadros de texto en casos específicos**, como:

- Ingreso de detalles cuando se selecciona Otro.
- Registro de nombre en ciertas causas relacionadas con Producción.
- Especificaciones adicionales (ej. tipo de material faltante en Logística).

Con esta implementación, el sistema permite que **cada paro quede debidamente clasificado** por área y causa y especificaciones.

Creación del análisis de paros por área

Como parte del sistema de monitoreo, se desarrolló una interfaz dedicada al análisis visual y estadístico de los paros, implementada en el archivo data_analysis.html. Esta sección tiene como finalidad ofrecer al personal de ingeniería, calidad y mantenimiento una herramienta gráfica para identificar rápidamente las áreas críticas y sus principales causas de paro.

1. Obtención de los datos desde el backend

El análisis se alimenta de la ruta `/get_analysis_data`, que consulta la base de datos y devuelve la información en formato JSON:

```
async function fetchAnalysisData() {
  try {
    const response = await fetch('/get_analysis_data');
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }
    const data = await response.json();
    console.log("Datos de análisis recibidos:", data);
    return data;
  } catch (error) {
    console.error("Error al obtener los datos de análisis:", error);
    return null;
  }
}
```

Ilustración 67.- Obtención de los datos desde el backend

Este bloque se encarga de **solicitar la información de paros** al servidor Flask y almacenarla en el objeto `data`, que después alimentará a los gráficos.

2. Creación de los gráficos con Chart.js

Una función genérica que construye los gráficos dinámicos:

```
function createChart(chartId, type, labels, data, title, onClickHandler = null) {
  const ctx = document.getElementById(chartId).getContext('2d');
  const existingChart = Chart.getChart(chartId);
  if (existingChart) {
    existingChart.destroy();
  }
}
```

Ilustración 68.- Creación de los gráficos con Chart.js

Aquí se usa **Chart.js** para dibujar gráficos de barras. Los parámetros permiten personalizar:

- `labels` → nombre de áreas, causas o estaciones.
- `data` → tiempos acumulados de paro en segundos.
- `title` → título dinámico del gráfico.

- onClickHandler → evento que permite mostrar más detalles al hacer clic.

3. Gráfico de paros externos por área

Este bloque construye el gráfico principal que muestra el tiempo de paros en cada área:

```
function createExternosByAreaChart(data) {
  const labels = data.map(item => item.area);
  const tiempos = data.map(item => item.tiempo);

  externosByAreaChartInstance = createChart('externosByAreaChart', 'bar', labels, tiempos, 'Tiempo de Paro por Área');

  const noDataMessage = document.getElementById('noDataMessageExternos');
  if (labels.length > 0) {
    noDataMessage.classList.add('hidden');
  } else {
    noDataMessage.classList.remove('hidden');
  }
  updateTopCausasExternosChart(null);
}
```

Ilustración 69.- Gráfico de paros externos por área

Se recorren los datos y se asignan al gráfico:

- **labels** → áreas como Logística, Ingeniería, Producción, etc.
- **tiempos** → tiempo acumulado de paros por cada área.

4. Top 3 causas por área seleccionada

Cuando el usuario hace clic en una barra del gráfico anterior, se muestran las tres causas más frecuentes:

```
function updateTopCausasExternosChart(selectedArea) {
  const topCausasTitle = document.getElementById('topCausasExternosTitle');
  const noDataMessage = document.getElementById('noDataMessageExternos');
  let labels = [];
  let tiempos = [];
}
```

Ilustración 70.- Top 3 causas por área seleccionada

- Permite **profundizar en cada área**.
- Despliega las **3 principales causas** que provocan paros, con su tiempo asociado.

5. Inicialización de los gráficos

Finalmente, todos los gráficos se cargan al abrir la página:

```
async function initCharts() {
  allAnalysisData = await fetchAnalysisData();

  if (allAnalysisData && allAnalysisData.tiempo_por_area_externos) {
    createExternosByAreaChart(allAnalysisData.tiempo_por_area_externos);
  }

  if (allAnalysisData && allAnalysisData.tiempo_por_causa_internos) {
    createInternosByCausaChart(allAnalysisData.tiempo_por_causa_internos);
  }

  if (allAnalysisData && allAnalysisData.tiempo_por_causa_echeck) {
    createEcheckByCausaChart(allAnalysisData.tiempo_por_causa_echeck);
  }
}

document.addEventListener('DOMContentLoaded', initCharts);
```

Ilustración 71.- Inicialización de los gráficos

Este bloque asegura que, al cargar la página, se generen **tres análisis principales**:

- Paros externos por área.
- Paros internos por causa.
- Paros en ECHECK.

Descarga de reportes en Excel

La opción de descarga en Excel se implementa para que los supervisores o ingenieros puedan analizar los paros fuera del sistema web, generar reportes oficiales y compartir la información con otras áreas.

Esta función concentra toda la información de un turno activo en diferentes hojas dentro de un mismo archivo Excel.

1. Ruta definida en el servidor Flask

```
@app.route('/download_shift_data_excel')
def download_shift_data_excel():
    with app.app_context():
        # Obtener el shift_day_id actual
        shift_id = current_shift_day_id
        if not shift_id:
            return jsonify({"error": "No hay un turno activo para descargar datos."}), 400
```

Ilustración 72.- Ruta definida en el servidor Flask

Aquí se valida que exista un turno activo (shift_day_id) antes de generar el archivo. Si no hay turno, se devuelve un error.

2. Historial de paros

Primero se obtiene el historial completo de los paros del turno y se organiza en una tabla:

```
historial_data.append([
    'Estacion': paro.estacion,
    'Tipo Paro': paro.tipo_paro,
    'Inicio': inicio_local.strftime('%Y-%m-%d %H:%M:%S'),
    'Fin': fin_local.strftime('%Y-%m-%d %H:%M:%S') if fin_local else "Activo",
    'Duracion (segundos)': paro.duracion_segundos,
    'Duracion (HH:MM:SS)': segundos_a_hms(paro.duracion_segundos) if paro.duracion_segundos is not None else "0",
    'Causa': paro.causa,
    'Area': paro.area
])
df_historial = pd.DataFrame(historial_data)
```

Ilustración 73.- Historial de paros

Cada fila representa un paro con su inicio, fin, duración, causa y área responsable.

3. Resumen total de paros

Se agregan estadísticas generales para separar paros internos y externos:

```
summary_data = {
    "Tipo de Paro": ["Externos", "Internos"],
    "Total de Paros": [total_num_paros_externos, total_num_paros_internos],
    "Tiempo Total (segundos)": [total_tiempo_externos_segundos, total_tiempo_internos_segundos],
    "Tiempo Total (HH:MM:SS)": [segundos_a_hms(total_tiempo_externos_segundos), segundos_a_hms(total_tiempo_internos_segundos)]
}
df_summary_totals = pd.DataFrame(summary_data)
```

Ilustración 74.- Resumen total de paros

Aquí se genera una hoja con la comparación global de todos los paros externos vs internos.

4. Estadísticas por estación

Se calculan indicadores específicos de cada estación:

```
# Estadísticas por estación
estadisticas_por_estacion_list = []
for station, data in estadisticas_paros_summary.items():
    estadisticas_por_estacion_list.append({
        "Estacion": station,
        "Numero de Paros": data["num_paros"],
        "Tiempo Total (segundos)": data["tiempo_total"],
        "Tiempo Total (HH:MM:SS)": segundos_a_hms(data["tiempo_total"])
    })
df_estadisticas_estacion = pd.DataFrame(estadisticas_por_estacion_list)
```

Esta parte permite identificar cuáles estaciones son las que más contribuyen a los tiempos muertos.

5. Exportación final a Excel

Todo se combina y se guarda en distintas hojas del mismo archivo Excel:

```
output = io.BytesIO()
with pd.ExcelWriter(output, engine='openpyxl') as writer:
    df_historial.to_excel(writer, sheet_name='Historial de Paros', index=False)
    df_summary_totals.to_excel(writer, sheet_name='Resumen Total Paros', index=False)
    df_estadisticas_estacion.to_excel(writer, sheet_name='Estadísticas por Estación', index=False)
```

Ilustración 75.- Crear el archivo Excel en memoria

Finalmente, se envía el archivo al navegador para su descarga:

```
filename = f"Reporte_Paros_{shift_id.replace(' ', '_').replace('-', '')}.xlsx"
return send_file(output, download_name=filename, as_attachment=True, mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')
```

Ilustración 76.- Nombre del archivo Excel

El usuario recibe un archivo .xlsx con múltiples pestañas que organizan la información en:

- **Historial de Paros**
- **Resumen Total**
- **Estadísticas por Estación**
- **Análisis por área o causa**

De esta forma, el sistema no solo muestra los datos en tiempo real en la web, sino que también genera reportes listos para auditorías, juntas de producción y análisis de eficiencia.

Resultados

Se logró diseñar e implementar un **web server de paros** conectado a un **PLC Siemens S7-1500** mediante la librería **Snap7**, integrando una base de datos relacional gestionada con **SQLAlchemy**. El sistema permite la **visualización en tiempo real** de los paros por estación, la **clasificación automática** en externos (Andon) e internos (Clever), y la **asignación de causas** directamente desde la interfaz web mediante menús desplegables y cuadros de texto.

Previo al desarrollo, los registros de paros dependían de la memoria y de anotaciones manuales, lo que ocasionaba inconsistencias y pérdida de información. Con este sistema, se obtiene el **inicio, fin y duración exacta** de cada paro, logrando un **100% de trazabilidad** y eliminando las omisiones. Toda la información quedó centralizada en la base de datos, disponible para consultas y reportes en cualquier momento.

Se desarrollaron tres interfaces principales:

- **Estado de Paros en Tiempo Real (index.html)** para visualizar paros activos y acumulados.

- **Resumen de Paros (resumen.html)** para asignar causas y revisar históricos.
- **Análisis de Paros por Área (data_analysis.html)** para generar gráficos dinámicos que muestran tiempos acumulados y principales causas.

Además, se implementó la opción de **descarga de reportes en Excel**, que incluye el historial de paros del turno, un resumen global por tipo de paro y estadísticas por estación y área.

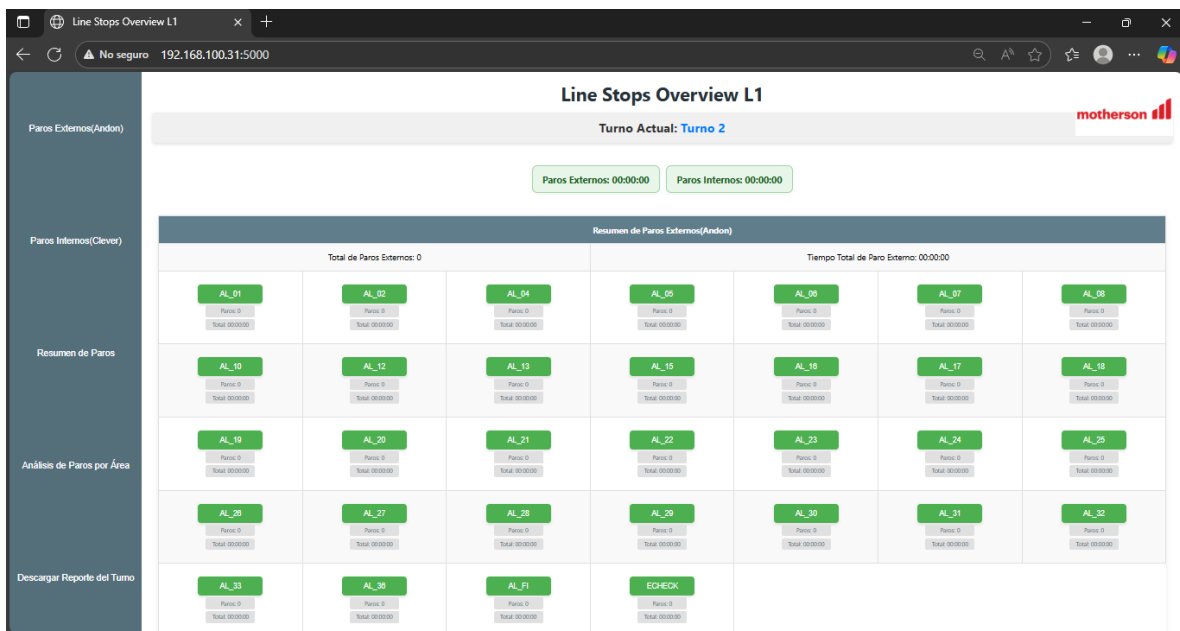


Ilustración 77.- Interfaz principal de los paros externos (Andon)

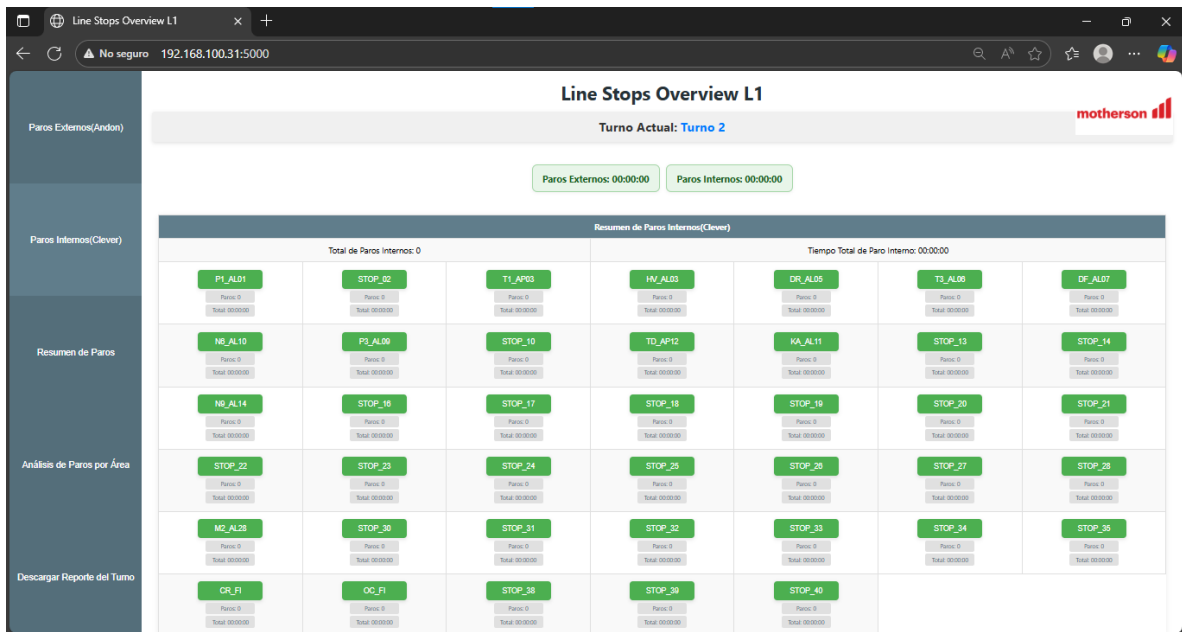


Ilustración 78.- Interfaz principal de los paros internos (Andon)

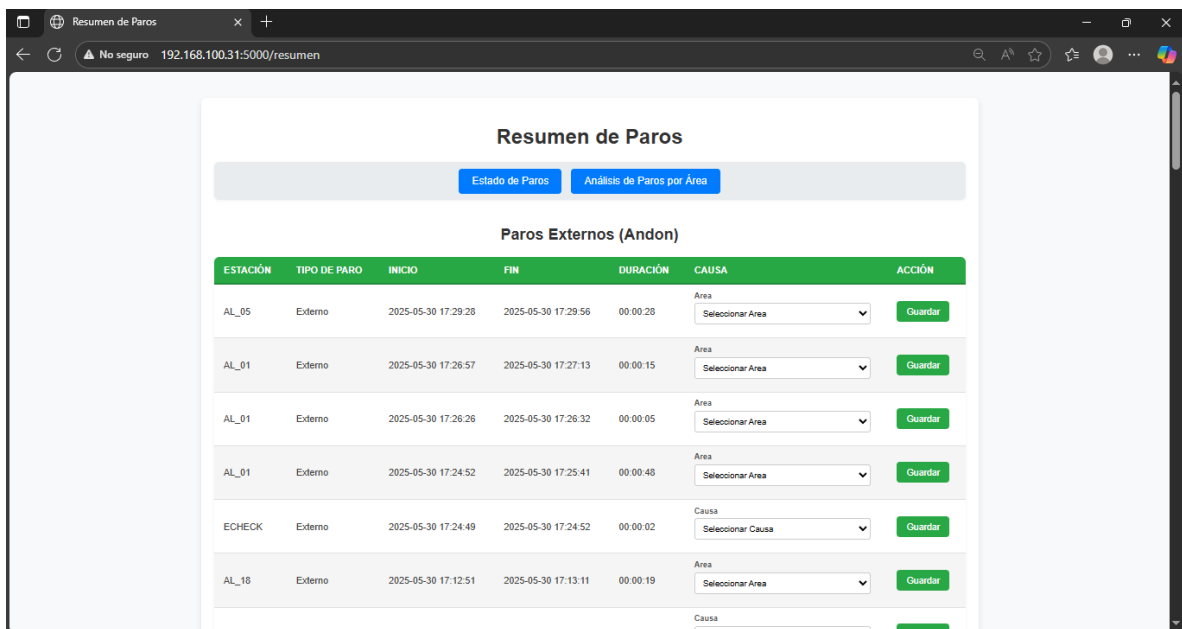


Ilustración 79.- Resumen de paros externos

Resumen de Paros

No seguro 192.168.100.31:5000/resumen

Paros Internos (Clever)

ESTACIÓN	TIPO DE PARO	INICIO	FIN	DURACIÓN	CAUSA	ACCIÓN
P1_AL01	Interno	2025-05-30 17:22:18	2025-05-30 17:22:31	00:00:12	Causa Seleccionar Causa	Guardar
M2_AL28	Interno	2025-05-30 17:22:08	2025-05-30 17:23:13	00:01:05	Causa Seleccionar Causa	Guardar
OC_FI	Interno	2025-05-30 17:06:07	2025-05-30 17:06:11	00:00:04	Causa Seleccionar Causa	Guardar
OC_FI	Interno	2025-05-30 17:05:08	2025-05-30 17:05:12	00:00:04	Causa Seleccionar Causa	Guardar
P3_AL09	Interno	2025-05-30 16:55:29	2025-05-30 16:55:42	00:00:12	Causa Seleccionar Causa	Guardar
P1_AL01	Interno	2025-05-30 16:52:52	2025-05-30 16:53:04	00:00:11	Causa Seleccionar Causa	Guardar
KA_AL11	Interno	2025-05-30 16:48:17	2025-05-30 16:48:20	00:00:03	Causa Seleccionar Causa	Guardar
P1_AL01	Interno	2025-05-30 16:44:51	2025-05-30 16:44:59	00:00:07	Causa Seleccionar Causa	Guardar
P3_AL09	Interno	2025-05-30 16:29:21	2025-05-30 16:29:38	00:00:16	Causa Seleccionar Causa	Guardar

Ilustración 80.- Resumen de paros internos

ESTACIÓN	TIPO DE PARO	INICIO	FIN	DURACIÓN	CAUSA	ACCIÓN
AL_05	Externo	2025-05-30 17:29:28	2025-05-30 17:29:56	00:00:28	Área Logística Seleccionar Área Logística Ingeniería Calidad Producción Mantenimiento Sistemas Cliente	Guardar
AL_01	Externo	2025-05-30 17:26:57	2025-05-30 17:27:13	00:00:15	Área Logística Seleccionar Área Logística Ingeniería Calidad Producción Mantenimiento Sistemas Cliente	Guardar
AL_01	Externo	2025-05-30 17:26:26	2025-05-30 17:26:32	00:00:05	Área Logística Seleccionar Área Logística Ingeniería Calidad Producción Mantenimiento Sistemas Cliente	Guardar

Ilustración 81.- Menú de causas externos empezando por área

ESTACIÓN	TIPO DE PARO	INICIO	FIN	DURACIÓN	CAUSA	ACCIÓN
AL_05	Externo	2025-05-30 17:29:28	2025-05-30 17:29:56	00:00:28	Área Logística Causa Material dañado Especificación Seleccionar Especificación	Guardar

Ilustración 82.- Causa completa en paro externo

ECHECK	Externo	2025-05-30 16:09:56	2025-05-30 16:10:13	00:00:16	Causa Seleccionar Causa ▼	Guardar
ECHECK	Externo	2025-05-30 16:08:34	2025-05-30 16:08:36	00:00:01	Causa Seleccionar Causa ▼	Guardar
AL_FI	Externo	2025-05-30 16:07:46	2025-05-30 16:08:00	00:00:14	Seleccionar Causa prueba nok retrabajo falta de equipo Otro	Guardar
ECHECK	Externo	2025-05-30 16:06:21	2025-05-30 16:06:38	00:00:16		Guardar

Ilustración 83.- Causas ECHECK

ESTACIÓN	TIPO DE PARO	INICIO	FIN	DURACIÓN	CAUSA	ACCIÓN
P1_AL01	Interno	2025-05-30 17:22:18	2025-05-30 17:22:31	00:00:12	Causa Seleccionar Causa ▼	Guardar
M2_AL28	Interno	2025-05-30 17:22:08	2025-05-30 17:23:13	00:01:05	Seleccionar Causa tiempo expirado cockpit incompleto error de secuencia ventana nok Otro	Guardar
OC_FI	Interno	2025-05-30 17:06:07	2025-05-30 17:06:11	00:00:04		Guardar

Ilustración 84.- Causas paros internos

AL_FI	Externo	2025-05-30 15:40:29	2025-05-30 15:41:10	00:00:40	Logística - Material equivocado	-
-------	---------	---------------------	---------------------	----------	---------------------------------	---

Ilustración 85.- Paro externo guardado

T3_AL06	Interno	2025-05-30 15:32:27	2025-05-30 15:33:00	00:00:32	Interno - tiempo expirado	-
---------	---------	---------------------	---------------------	----------	---------------------------	---

Ilustración 86.- Paro interno guardado

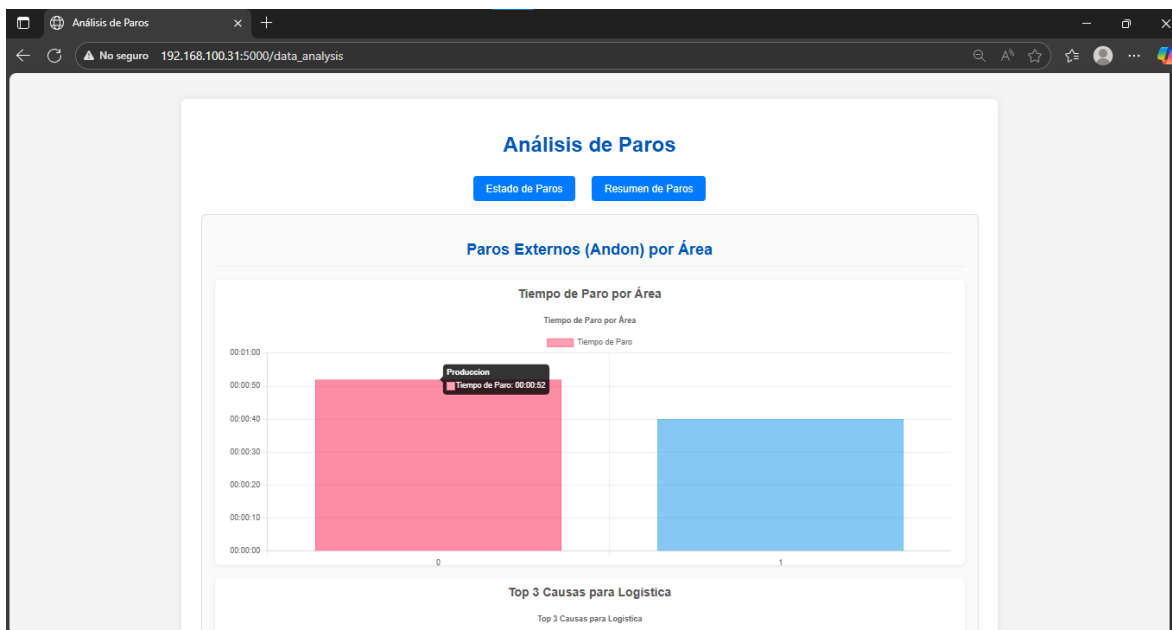


Ilustración 87.- análisis de paros externos

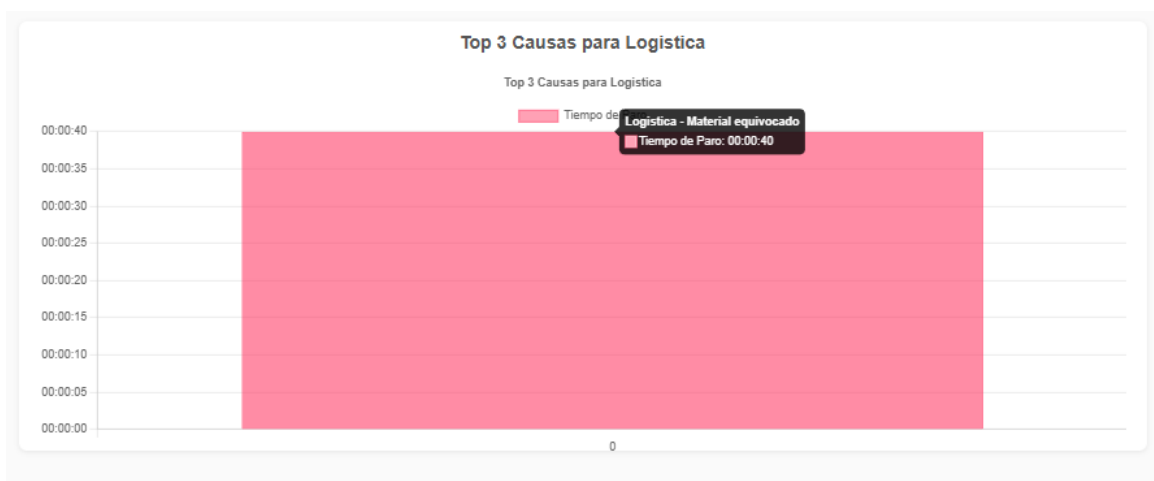


Ilustración 88.- Top 3 causas por Área



Ilustración 89.- análisis de paros internos

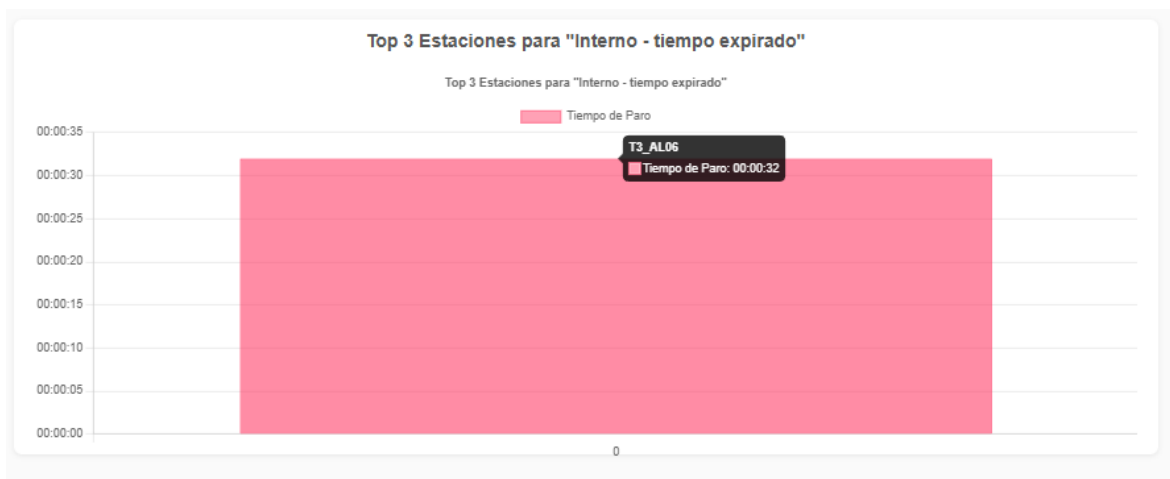


Ilustración 90.- Top 3 estaciones por internos (causas)



Ilustración 91.- análisis paros ECHECK

The screenshot displays the "Line Stops Overview L1" dashboard. At the top, it shows "Turno Actual: Turno 3" and two status indicators: "Paros Externos: 00:00:00" and "Paros Internos: 00:00:00". Below this is a "Resumen de Paros Externos(Andon)" section with a table of stop analysis cards. Each card displays a stop code (e.g., AL_01, AL_02, etc.), a "Paros: 0" status, and a "Total: 00:00:00" time. A red notification bubble in the center of the grid states "Reporte de turno descargado exitosamente." To the right, a "Descargas" (Downloads) menu is open, showing two download links for "Reporte_Paros_20250917_Turno_3 (1).xlsx" and "Reporte_Paros_20250917_Turno_3.xlsx", each with an "Abrir archivo" (Open file) button. The dashboard also includes a sidebar with navigation options like "Paros Externos(Andon)", "Paros Internos(Clevar)", "Resumen de Paros", "Análisis de Paros por Área", and "Descargar Reporte del Turno". The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 05:02 a. m. on 17/09/2025.

Ilustración 92.- Descarga de reporte por turno en Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1																					
2		Externos	0	0	00:00:00																
3		Internos	0	0	00:00:00																
4																					
5																					
6																					
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					
16																					
17																					
18																					
19																					
20																					
21																					
22																					

Ilustración 93.- Reporte de Excel

Conclusiones

Conclusiones del proyecto

El desarrollo del Web Server de paros de línea en la empresa Motherson permitió implementar una herramienta digital que centraliza el registro, monitoreo y análisis de los incidentes ocurridos durante la producción. Con esta solución, se superaron las limitaciones de los registros manuales, que en muchos casos eran incompletos, inexactos o poco oportunos, dificultando la toma de decisiones en tiempo real.

La integración con el PLC Siemens S7-1500 mediante la librería Snap7 garantizó una comunicación confiable y continua entre el sistema y la línea de producción, permitiendo la lectura y escritura de estados de paros. De esta forma, los operadores cuentan con un mecanismo eficiente para registrar tanto los paros internos como los externos, así como sus causas específicas.

La incorporación de módulos de análisis, como el resumen de paros y el análisis por área, facilitó la identificación de los principales factores que provocan interrupciones en la línea, lo que representa un gran apoyo para áreas como Producción,

Mantenimiento, Ingeniería y Calidad. Esto no solo fortalece la trazabilidad de los eventos, sino que también permite establecer planes de acción preventivos basados en información objetiva.

En términos de impacto, el proyecto contribuye directamente a:

- Reducir los tiempos muertos en la producción.
- Mejorar la eficiencia operativa en la Línea 1.
- Favorecer la comunicación entre operadores y supervisores.

Asegurar la disponibilidad de información en tiempo real para la toma de decisiones.

Finalmente, este sistema sienta las bases para futuras mejoras, como la automatización de reportes en Excel por turno, la implementación de notificaciones en tiempo real hacia dispositivos móviles y la integración con otras líneas de producción, lo que permitirá a Motherson seguir avanzando hacia un modelo de industria 4.0.

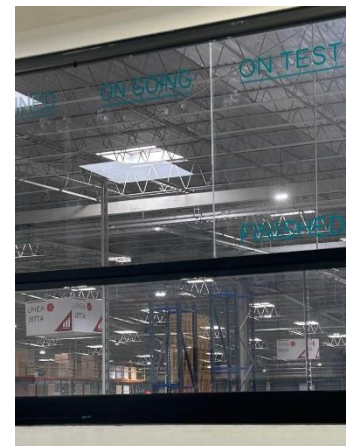
Recomendaciones

1. **Mejorar la capacitación del personal:** Es recomendable continuar con la formación de los operadores en el uso del sistema web de monitoreo, con el fin de garantizar un registro correcto y oportuno de las causas de los paros.
2. **Integrar indicadores de desempeño (KPIs):** Incluir métricas automáticas como OEE (Overall Equipment Effectiveness) y MTTR/MTBF, permitiría a la empresa tener una visión más estratégica del impacto de los paros.
3. **Automatizar reportes periódicos:** Se sugiere programar la generación automática de reportes diarios y semanales en Excel o PDF, que se envíen directamente por correo electrónico a las áreas involucradas.
4. **Escalabilidad del sistema:** Se recomienda ampliar la implementación del sistema a otras líneas de producción, lo que permitiría tener una base de datos centralizada y comparativa entre líneas.
5. **Mantenimiento preventivo del sistema:** Es importante establecer rutinas de respaldo de la base de datos y verificación de la comunicación con el PLC, con el fin de garantizar la confiabilidad de la información en el tiempo.

Experiencia personal y profesional adquirida

La experiencia obtenida durante el desarrollo de este proyecto representó un aprendizaje integral que fortaleció tanto mis competencias técnicas como mis habilidades profesionales:

- **Aplicación práctica del conocimiento:** Pude aplicar de manera directa los conocimientos adquiridos en mi formación académica en ingeniería, integrando áreas como programación, automatización y análisis de datos.
- **Conexión entre la teoría y la industria:** Este proyecto me permitió vivir de primera mano cómo la digitalización y la automatización son aplicadas en una empresa automotriz de gran escala como Motherson.
- **Trabajo colaborativo:** Aprendí la importancia de la comunicación con diferentes áreas (Producción, Calidad, Mantenimiento e Ingeniería) para adaptar el sistema a necesidades reales.
- **Desarrollo de autonomía:** La necesidad de investigar nuevas herramientas (Flask, SQLAlchemy, Chart.js, Pandas, Snap7) me ayudó a fortalecer mi capacidad de autoaprendizaje y de resolución de problemas.
- **Formación profesional sólida:** Considero que la experiencia adquirida me prepara mejor para enfrentar futuros retos laborales, ya que me permitió consolidar una visión integral de cómo las soluciones tecnológicas pueden mejorar la eficiencia y competitividad en un entorno industrial.



Competencias desarrolladas

Durante la realización del proyecto de residencia profesional en la empresa Motherson, se lograron desarrollar y fortalecer diversas competencias, tanto en el ámbito técnico como en el profesional:

1. Competencias técnicas

Programación en Python y Flask: Se adquirió dominio en el uso del framework Flask para el desarrollo de aplicaciones web, integrando rutas, comunicación en tiempo real y manejo de datos desde el servidor.

Bases de datos con SQLAlchemy: Se desarrolló la capacidad de modelar, gestionar y consultar bases de datos relacionales mediante SQLAlchemy, garantizando la persistencia y consulta eficiente de la información de paros.

Integración con PLC Siemens S7-1500: Se fortalecieron habilidades en automatización industrial mediante la comunicación con PLCs utilizando la librería Snap7, permitiendo la interacción directa entre el sistema digital y la línea de producción.

Desarrollo de interfaces web dinámicas: Se implementaron vistas en HTML, CSS y JavaScript para la visualización de estados de paro, registro de causas y análisis de datos.

Gestión y análisis de datos: Se adquirió experiencia en el uso de Pandas y herramientas de visualización (Chart.js) para transformar los registros de paros en reportes y gráficas que apoyan la toma de decisiones.

2. Competencias profesionales

Trabajo en equipo interdisciplinario: Se participó en reuniones con las áreas de Producción, Mantenimiento, Calidad e Ingeniería para definir causas de paro y validar la funcionalidad del sistema.

Resolución de problemas: Se enfrentaron y resolvieron retos relacionados con la integración de hardware y software, la validación de datos y la confiabilidad del sistema en tiempo real.

Gestión de proyectos: Se fortaleció la capacidad de planear, documentar y dar seguimiento a las actividades dentro de un cronograma establecido, cumpliendo con los objetivos del proyecto.

Comunicación efectiva: Se desarrollaron habilidades de comunicación para documentar avances, presentar resultados y explicar el funcionamiento del sistema a personal técnico y no técnico.

3. Competencias personales

Autonomía y aprendizaje autodidacta: Se adquirió la capacidad de investigar nuevas tecnologías y adaptarlas al contexto del proyecto.

Adaptabilidad: Se aprendió a responder rápidamente a los cambios en los requerimientos del sistema y a las necesidades de la empresa.

Pensamiento crítico y analítico: Se logró analizar datos históricos de paros para identificar patrones y proponer mejoras orientadas a la reducción de tiempos muertos.

Fuentes de información

Anderson, B., & Fagerhaug, T. (2000). *Root Cause Analysis: Simplified Tools and Techniques*. ASQ Quality Press.

Bolton, W. (2015). *Programmable Logic Controllers*. Elsevier.

Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.

Gilchrist, A. (2016). *Industry 4.0: The Industrial Internet of Things*. Apress.

Holbrook, R., & Al-Akaidi, M. (2014). *Introduction to Industrial Automation and Control Systems*. De Gruyter.

Hughes, T. A. (2011). *Programmable Controllers*. ISA.

Kagermann, H., Wahlster, W., & Helbig, J. (2013). *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0*. Acatech.

Mahdavi, A. (2013). OPC UA: The Basics. In *OPC UA: The Foundation for a Service-Oriented Manufacturing*. Springer.

Nakajima, S. (1988). *Introduction to TPM: Total Productive Maintenance*. Productivity Press.

Nielsen, J., & Budiu, R. (2013). *Mobile Usability*. New Riders.

Norman, D. A. (2013). *The Design of Everyday Things*. Basic Books.

Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Productivity Press.

Sharda, R., Delen, D., & Turban, E. (2020). *Business Intelligence, Analytics, and Data Science: A Managerial Perspective*. Pearson.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database System Concepts*. McGraw-Hill Education.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The Machine That Changed the World: The Story of Lean Production*. Harper Perennial.