Solución IoT para el registro y visualización del estado de la producción en la industria 4.0

Silva Pablo Eduardo
Universidad Nacional de Misiones (UNaM), Facultad de Ingeniería (FI)
Oberá, Misiones, Argentina
pablosilva.ten@gmail.com

Iurinic Gerardo Mathías
Universidad Nacional de Misiones (UNaM), Facultad de Ingeniería (FI)
Oberá, Misiones, Argentina
gerardo_m_iurinic@hotmail.com

Krzyzanowski, Lucas Nicolás
Universidad Nacional de Misiones (UNaM), Facultad de Ingeniería (FI)
Oberá, Misiones, Argentina
lucasclark997@gmail.com

***Abstract*— This work presents an IoT-based solution for real-time monitoring of machine operation in industrial environments. The proposed system integrates distributed sensing modules that detect machine states and transmit them to a local server for storage and processing. A web platform provides interactive dashboards that combine percentage-based and time-series visualizations, enabling users to identify idle periods, filter results by shifts or custom time ranges, and export historical reports for offline analysis. By offering a modular and low-cost architecture, the solution improves visibility of production processes and supports data-driven decision-making management within the framework of Industry 4.0.**

***Keywords*— IoT; Industry-4.0; Production-Monitoring.**

***Resumen*— Se presenta una solución integral de IoT para el monitoreo en tiempo real del estado de producción en plantas industriales, orientada a identificar periodos improductivos de las máquinas. A través de módulos de sensado distribuidos, se captura información sobre los tiempos de encendido y actividad productiva, la cual es centralizada en un servidor local. Una plataforma web interactiva procesa estos datos, ofreciendo gráficos porcentuales y de evolución temporal, controles para filtrar por turno o rangos personalizados, y la posibilidad de exportar informes en formato Excel. Esta combinación de sensado embebido y visualización dinámica mejora la visibilidad de los procesos productivos y apoya la toma de decisiones basadas en estos datos.**

***Palabras-clave*— IoT;Industria-4.0; Monitoreo-Producción;**

I. INTRODUCCIÓN

La Industria 4.0 representa un cambio de paradigma en los sistemas productivos globales. También conocida como la cuarta revolución industrial [1], esta transformación se caracteriza por la integración de tecnologías digitales, físicas y biológicas que modifican no solo los procesos de manufactura, sino también los modelos de negocio y la interacción entre actores económicos. Según el Foro Económico Mundial, esta revolución no se limita a la automatización, sino que introduce sistemas ciberfísicos, Internet de las Cosas (IoT), inteligencia artificial (IA) y análisis de datos avanzados en todas las etapas de la cadena de valor. La mencionada convergencia tecnológica habilita la creación de entornos de producción más conectados, capaces de recolectar, procesar y compartir información en tiempo real, redefiniendo la manera en que los bienes son diseñados, fabricados, monitoreados y mantenidos.

El IoT es uno de los ejes de la Industria 4.0, consiste en la conexión de sensores y dispositivos a para recopilar en tiempo real datos sobre su estado y rendimiento. Esto permite nuevas formas de gestión de la producción, habilitando el monitoreo remoto y una rápida reacción ante anomalías. En la práctica, la información provista por el IoT está optimizando la eficiencia operativa: al detectar ineficiencias o paros tempranos, se logró reducir el tiempo improductivo y maximizar los recursos utilizados [2].

El IoT en procesos productivos permite la creación de paneles de control y *dashboards* donde se visualiza el desempeño de cada línea en tiempo real como de manera histórica, lo cual mejora significativamente la toma de decisiones sobre la marcha y también a largo plazo [2].

En Latinoamérica la creciente adopción del IoT está impulsando la transformación digital al permitir que empresas manufactureras, agrícolas y logísticas reduzcan costos y mejoren gradualmente la calidad de sus productos o servicios [3]. Por ejemplo, se documenta el caso de Tecpetrol (sector energético), donde 200 sensores envían 3 millones de datos por hora desde yacimientos remotos, permitiendo a la empresa estandarizar operaciones y tomar decisiones desde centros urbanos [4].

La digitalización de la planta productiva conlleva una capacidad inédita del control de la producción. Con el IoT surgen el Sistema de Ejecución de la Manufactura (MES) o uno más completo el cual se trata de la Gestión de Operaciones de Manufactura (MOM), con estos integrados las empresas pueden medir en tiempo real indicadores clave de fabricación como la Eficacia Global de los Equipos (OEE), tiempos de parada de máquina, unidades defectuosas, entre otros. También, al disponer de terminales interfaz hombre-máquina (HMI) conectadas a los equipos, el personal puede reportar automáticamente cada incidencia o finalización de orden, alimentando el sistema con datos precisos. De esta forma, en un futuro el equipo de mantenimiento puede anticipar paradas antes de que ocurran; además, los sistemas inteligentes de análisis predictivo pueden alertar sobre tendencias anómalas y programar intervenciones preventivas [5]. Todo esto contribuye a reducir los periodos improductivos — como lo pueden ser máquinas encendidas sin producir— y a incrementar la productividad general. Estas soluciones IoT para el piso de fábrica facilitan reportes personalizados y alarmas, mejorando notablemente la supervisión y gestión de la producción.

En Argentina, el interés por Industria 4.0 e IoT crece rápidamente y se encuentra en pleno desarrollo. El informe del Banco Interamericano de Desarrollo (BID) destaca que tecnologías como IoT, *big data* e IA “refuerzan la importancia de la industria manufacturera a partir de la fabricación de

productos personalizados e inteligentes" [6]. En la práctica, muchas empresas argentinas ya incorporan componentes 4.0 en sus procesos. Un estudio con Pequeñas y Medianas Empresas (PyMEs) argentinas señala que diversas compañías están aplicando Tecnología de la Información y las Comunicaciones (TIC), servicios *cloud* e IA para automatizar su producción de principio a fin [7]. Este fenómeno no se limita a sectores avanzados sino hoy en día la manufactura básica, como los alimentos y otros rubros también integran IoT y automatización para mejorar su operación.

Sin embargo, persisten los desafíos locales. En la región falta infraestructura de conectividad robusta y presupuestos amplios, lo que retrasa la adopción masiva del IoT [3] [4]. En Argentina en particular, sólo un tercio de las empresas afirma planear la incorporación total de tecnologías 4.0 en los próximos años y el 70% identifica la escasez de personal calificado como la principal limitante [4]. A pesar de ello, organismos públicos e iniciativas de colaboración regional (Argentina, Paraguay y Brasil) están impulsando la transformación digital mediante programas de formación, incentivos e investigación conjunta [6].

A pesar de todos los indicadores y sistemas mencionados anteriormente, muchas plantas industriales en la región aún carecen de herramientas que midan con precisión los períodos improductivos de sus equipos. Con frecuencia, las máquinas permanecen encendidas sin estar efectivamente produciendo, lo cual genera consumos de energía y costos de mantenimiento innecesarios que pasan desapercibidos. La falta de visibilidad y de análisis de estos eventos limita la capacidad de los responsables de planta para cuantificar pérdidas y tomar decisiones informadas, dado que los sistemas tradicionales de registro suelen ser manuales, esporádicos y de baja granularidad temporal.

En este contexto, surge la necesidad de una solución que combine la simplicidad de un sistema de sensado local con la potencia de una plataforma web para el registro y la visualización continua del estado de producción. El presente trabajo describe una propuesta basada en nodos de adquisición distribuidos, capaces de detectar periódicamente el estado de cada máquina dentro de una o más líneas de producción. Estos datos se transmiten a un concentrador central, que los almacena en una base de datos para su posterior consulta. A partir de allí, se implementa un *dashboard* interactivo que permite explorar, filtrar y comparar el comportamiento de todas las máquinas, tanto por turnos en días específicos como en rangos de tiempo arbitrarios. Se establecen dos tipos principales de visualizaciones:

- **Gráficos de torta** que muestran de forma porcentual el tiempo durante el cual la máquina estuvo encendida y, dentro de ese conjunto, el tiempo que realmente estuvo produciendo.
- **Gráficas temporales** donde el eje horizontal representa la secuencia de muestras a lo largo del tiempo y permite observar con exactitud los instantes en que cambian los estados de la máquina.

La interfaz facilita la comparación simultánea de varias máquinas, para detectar si un evento corresponde a una falla general en la planta o a un problema puntual de un solo equipo.

El documento se estructura de la siguiente manera: La Sección II detalla la arquitectura general del sistema IoT completo. La Sección III describe el módulo de adquisición de datos embebido y su comunicación. La Sección IV explica la plataforma web de visualización y consulta de datos. La Sección V presenta los resultados de hardware ensamblado y de las pruebas de funcionamiento. Finalmente, la Sección VI expone las conclusiones y líneas de trabajo futuro.

II. Arquitectura General del Sistema

La solución propuesta en este trabajo tiene como fin el sensado, registro y visualización de los tiempos en los cuales las maquinas dentro de una industria, se encuentran encendidas/apagadas y estan produciendo o no. En la Fig. 1 se muestra como se interconectan los principales componentes del sistema, a la izquierda, cada máquina dispone de un módulo de adquisición embebido que mediante el sensado de señales eléctricas, comunica el estado de la máquina. Estos módulos forman una topología en cadena *daisy chain* y emplean el protocolo de comunicación Modbus RTU, el cual utiliza el estándar RS-485 en la capa física, para transmitir datos hasta la unidad central.

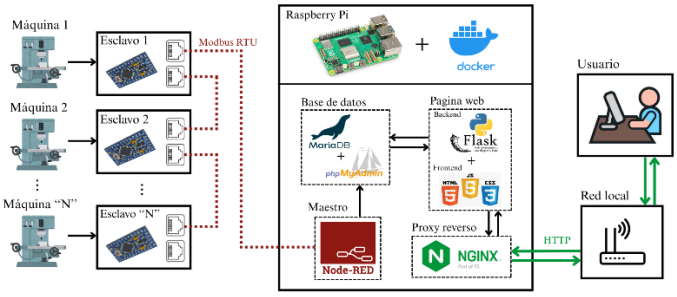


Fig. 1. Diagrama en bloques de la solución IoT completa.

En el centro del diagrama se encuentra la Raspberry Pi que orquesta todo el sistema dentro de contenedores Docker, donde:

- **Node-RED** recibe las tramas Modbus RTU de cada servidor, las decodifica y las inserta en la base de datos.
- **MariaDB** almacena los registros de estado y mantiene vistas optimizadas para su consulta. Se ofrece administración de esta mediante el servicio **phpMyAdmin**.
- **Nginx** se configura como *proxy* reverso, donde según la ruta solicitada, encamina el tráfico hacia el *backend* **Flask** o las páginas estáticas del *frontend*.
- **Flask** implementa la *API REST* que atiende los diferentes *endpoints* procesando consultas a la base de datos y devolviendo un *JSON* con los datos necesarios para los *dashboards*.

A la derecha, desde cualquier navegador en la red local, el usuario accede al frontend mediante peticiones *HTTP* al *proxy* Nginx, el cual las redirecciona hacia las API de Flask que devuelven las pantallas del sistema donde el usuario puede alternar entre gráficos de tortas y series de tiempo, así como descargar reportes Excel sobre los periodos de interés. Esta arquitectura modular aprovecha el uso de contenedores Docker para asegurar que cada componente sea fácil de desplegar, escalar y actualizar sin afectar al resto del sistema.

III. Módulo de Adquisición de Datos

A. *Sensado en las Máquinas*

El módulo de sensado instalado en cada máquina integra el muestreo directo de canales de detección independientes los cuales, mediante un optoacoplador que brinda una aislación optica, convierte la presencia de tensión en cierto circuito de la máquina en una señal digitalizada apta para un microcontrolador. Su principio de funcionamiento consiste en detectar alimentaciones especificas de la máquina que indican el estado de funcionamiento que se encuentra la misma, por ejemplo una bobina que solo se energiza cuando un operario esta activamente trabajando sobre la maquina. Esta señal de entrada es adecuada, para luego excitar el diodo *LED* interno del optoacoplador cuando existe voltaje, esta configuracion se puede observar con mas detalle en la *Fig. 2*. La salida del fototransistor del optoacoplador se conecta a un pin de un Arduino Pro Mini, el cual es el corazon de este circuito. El firmware del microcontrolador interpreta cada canal como un parametro clave de la maquina a registrar. El diseño se realizó pensando en máxima flexibilidad, aunque hoy sólo se usan dos entradas (alimentación de la máquina y producción), el tercer canal queda libre para futuras expansiones o para algun caso donde se requiera la medición de 3 parametros. Pudiendo de igual manera incorporar la cantidad necesaria segun el caso de uso, quedando limitado este valor a la cantidad de entradas que posea el microcontrolador.

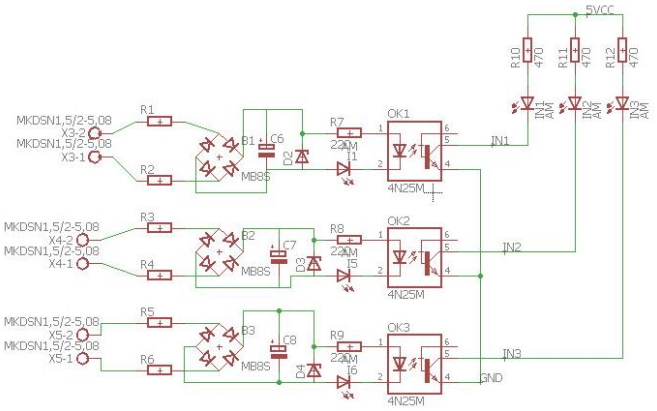


Fig. 2. Esquema electrico del circuito que realiza el sensado.

En el esquema mostrado en la *Fig. 2*. puede verse a la entrada de señal a la izquierda el puente de diodos MB8S, un capacitor de suavizado y el resistor que limita la corriente al *LED* del 4N25, como tambien un diodo para proteccion de caidas de voltaje inversas en el opto. Luego a la derecha se tienen las salidas protegida para las entradas del Arduino, donde hay *LEDs* indicadores de presencia de la señal. Gracias al aislamiento óptico que brinda el 4N25, cualquier sobretensión o fallo en la alimentación de la máquina no repercute en el microcontrolador ni en el resto de la placa.

B. *Comunicación y Registro de Datos*

Para garantizar la integridad en entornos industriales con alto ruido electromagnético, la comunicación entre cada modulo y el concentrador de informacion se emplea el protocolo Modbus RTU encapsulado sobre RS-485, donde cada módulo actúa como un esclavo que responde al maestro. Para esto se utiliza un transceptor MAX485 el cual convierte los niveles *TTL* del microcontrolador en líneas diferenciales balanceadas, robustas ante interferencias.

Físicamente, cada módulo dispone de dos conectores RJ-45 conectados con los demas formando una topología "*daisy-chain*" abierta (*open-loop*) lo que permite ampliar la red sin necesidad de cableados extra. Se evito redundancia para una implementacion mas sencilla. Mediante un cable de red utilizando dos pares trenzados recibe tanto la alimentación para la placa, como los pares de datos A/B de RS-485. Gracias a este diseño, los módulos pueden interconectarse en serie, finalizando el bus en el último nodo con una resistencia de terminación para evitar reflexiones. De este modo, se logra una topología sencilla de escalar, con modulos esclavos de sensado que responden a consultas del maestro Node-RED que reside en un contenedor Docker dentro de la Raspberry Pi.

Node-RED a su vez registra la tabla principal del sistema, para cada evento, la marca de tiempo en GMT-3 (hora estándar de Argentina), el identificador de la máquina y dos indicadores booleanos: *mOn*, que refleja si la máquina está encendida (1) o apagada (0); y *mWo*, que señala si está produciendo (1) o no (0). La Tabla I ilustra un ejemplo de registros en la base de datos.

TABLA I
Estructura de los registros almacenados en la base de datos.

id	time	maquina	mOn	mWo
1	2025-07-06 07:00:00	3	1	0
2	2025-07-06 07:03:00	3	1	1

En la solución siguiendo el lineamiento de un diseño modular y ajustable, el intervalo de muestreo es configurable mediante la configuración de los tiempos de consulta por parte del maestro, en esta implementación se estableció en 3 minutos, lo que permite un equilibrio entre granularidad de datos y carga de la red.

IV. Plataforma Web de Visualización y Consulta

A. *Backend y Servicios Web*

La capa de backend se implementó en Python utilizando el *micro-framework* Flask, orquestado junto a Nginx, MariaDB y phpMyAdmin dentro de contenedores Docker. Esta decisión permite un despliegue modular, escalable y desacoplado, donde cada componente puede actualizarse sin afectar al resto.

1) Contenedores y orquestación: **MariaDB** almacena los datos de los sensores que fueron cargados por **Node-RED** en tablas optimizadas para su consulta. Y **phpMyAdmin** ofrece una interfaz administrativa para verificar esquemas, índices y ejecutar consultas *ad-hoc*; Con **Flask** se tiene un *endpoint REST* (/api/vista) que hace uso de una *view* de la base de datos, la cual según los filtros recibidos (fecha/turno o rango personalizado) calcula internamente los instantes de inicio y fin y agrupa en la base de datos los registros de cada máquina para obtener el total de muestras, tiempo encendida, tiempo produciendo y tiempo encendida sin producir. Luego en *endpoint* devuelve estos valores ya agregados en un archivo JSON que el *frontend* consume para construir dinámicamente

los gráficos de torta y permitir al usuario explorar el comportamiento de las máquinas por turnos o periodos libres, sin preocuparse por detalles de zonas horarias, consultas SQL ni procesamiento de datos.

Flask también expone una API REST en el puerto interno 5000, donde atiende varios *endpoints* que procesan y requiero los parámetros de consulta (fecha, turno o rango libre), donde se ejecutan sentencias para la mencionada *view* y obtener series de datos agregados o individuales.

Y **Nginx** funciona como un *proxy* reverso, es decir, recibe todas las peticiones HTTP en el puerto 80, y enruta los paquetes según la ruta solicitada:

- Rutas estáticas (/static, /) hacia los archivos HTML/CSS/JS del *frontend*.
- Prefijo /api/ hacia el contenedor Flask.
- /phpmyadmin hacia el contenedor de administración de la base de datos.

2) Enpoints principales del backend:

GET /api/vista?date=<YYYY-MM-DD>&turno=<TM|TT|TN>

Recupera, para cada máquina, el total de muestras (N), cantidad encendido (On), apagado (Off), produciendo (Prod) y sin producir (NoProd) en el intervalo correspondiente al turno seleccionado.

GET /api/vista?start=<ISO>&end=<ISO>

Permite rangos arbitrarios mediante parámetros de inicio y fin en formato ISO-8601 (por ej. 2025-07-06T07:00).

GET /api/maquina/<m>?date=…&turno=…

GET /api/maquina/<m>?start=…&end=…

Devuelven la serie temporal completa de un único módulo: *timestamp*, *mOn*, *mWo* para cada muestra. Para un turno seleccionado o para un rango arbitrario respectivamente.

3) Gestión de parámetros y validaciones: Flask valida que siempre se provean "*date+turno*" o "*start+end*"; en caso contrario, redirige a la página de inicio (/). Las consultas *SQL* utilizan marcadores de posición y vistas materializadas para mejorar el rendimiento bajo alta concurrencia.

4) Seguridad y despliegue local: Aunque la red se limita al ámbito local (no se exponen puertos al exterior), Nginx incluye reglas de firewall y opcionalmente puede integrarse con certificados *Let's Encrypt* para HTTPS.

B. Interfaz de Usuario y Visualización de Datos

La interfaz gráfica se ha desarrollado con una plantilla de Bootstrap 5 (tema Cerulean) y graficos de la librería Chart.js, ofreciendo al usuario controles intuitivos para seleccionar fechas, turnos de trabajo o intervalos arbitrarios mediante campos de tipo *date* y *datetime*. Al enviar estos parámetros, el *frontend* realiza consultas asíncronas (*fetch*) al endpoint "/api/vista", que devuelve para cada máquina los totales de tiempo encendida, apagada, produciendo y sin producir. A partir de esta información, la página genera dinámicamente gráficos de torta superpuestos como se puede observar en la parte izquierda de la *Fig. 3*, aqui la capa interna del gráfico de pastel muestra el porcentaje de encendido frente al tiempo apagada, mientras que la capa externa refleja, dentro del porcentaje correspondiente a cuando la maquina estuvo encendida, que porcentaje de este tiempo fue de producción efectiva frente al tiempo en que la máquina estuvo encendida sin producir. Gracias a leyendas personalizadas y a la capacidad de alternar la visibilidad de cada capa de forma dinamica con el cursor como se puede ver en la parte derecha de la *Fig. 3*, el usuario puede explorar en detalle los datos sin recargar la página.

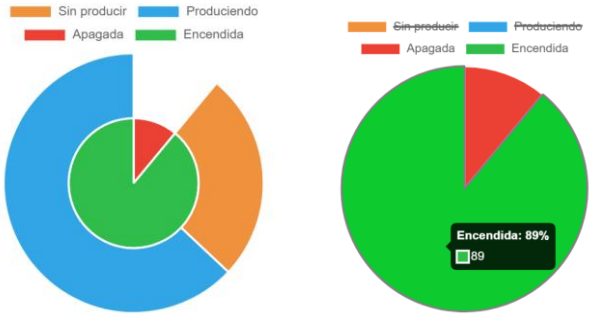


Fig. 3. Grafico dinamico de torta doble representando el tiempo en porcentajes.

Cuando se requiere un análisis más fino sobre la evolución temporal de una sola máquina, la aplicación invoca a "/api/maquina/<m>" con los mismos parámetros de fecha, turno o rango libre, y representa la serie de muestras a partir de un gráfico de líneas escalonado como se observar en la *Fig. 4* donde el eje horizontal recoge la marca de tiempo de cada registro, y el eje vertical distingue tres niveles fijos: "Apagada o sin producir" (valor 0), "Produciendo" (valor 0,8) y "Encendida" (valor 1).

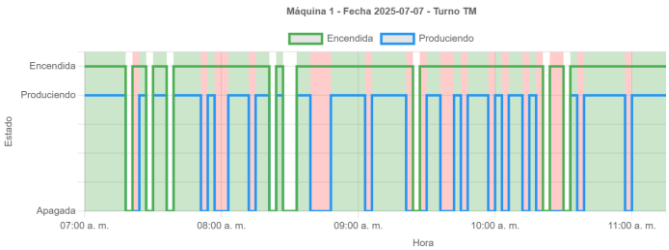


Fig. 4. Grafico de muestras a lo largo del tiempo.

Nótese tambien que se pintan de verde los intervalos donde la maquina se encuentra efectivamente encendida produciendo y de rojo los tiempos que son de especial importancia reconocer donde la maquina se encuentra encendida sin producir. Cada vez que cambia la selección de máquina o de intervalo, la instancia de Chart.js correspondiente se destruye y vuelve a crearse para reflejar con exactitud los nuevos datos. Cabe mencionar que si no se proporcionan parámetros al cargar la página, el sistema automáticamente consulta el último turno con datos disponibles y ajusta los controles para mostrar siempre la información más reciente.

V. RESULTADOS

A. Hardware final ensamblado

El diseño del módulo de sensado y comunicación de estado se envió a producción en cantidad a un fabricante de PCBs,

obteniéndose placas profesionales que incorporan todos los componentes críticos y algunas mejoras para futuras aplicaciones. En la *Fig. 5* se puede observar la versión final de cada módulo donde incluye, además de los puentes rectificadores y optoacopladores 4N25, el lugar para colocar un circuito de potencia con un relé electromagnetico que abre la puerta a integraciones de automatización (por ejemplo, activar señalizaciones o alarmas), manteniendo la maxima flexbilidad en la aplicacion como se menciono anteriormente. El transceptor RS-485 se seleccionó con un encapsulado DIP de 8 pines para un facil reemplazo ante una eventual falla en la linea de comunicación, podemos ver tambien a la izquierda los dos conectores RJ-45 que suministran alimentación y datos en un único cable, manteniendo la topología "daisy-chain". Tambien en la parte inferior se dejo un pequeño recuadro blanco para poder escribir el numero de esclavo que le es asignado a cada modulo por el maestro para la comunicación, pudiendo tener asi una rapida visualizacion de que que placa fisica se corresponde a cada esclavo .

Fig. 5. Circuito confeccionado para sensado y envio del estado de las maquinas.

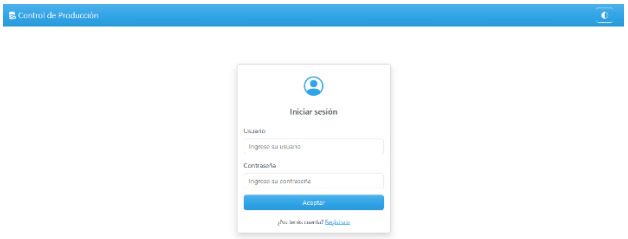
Como parte de la implementación en la planta, la Fig. 6 muestra el tablero industrial donde se aloja el servidor central que orquesta todo el sistema. Este servidor utiliza una Raspberry Pi que se conecta al bus RS-485 a través de un adaptador de protocolo USB/RS-485 (*dongle* azul). El cableado de la línea RS-485 utiliza conectores RJ45, de los cuales solo se emplea el par trenzado que transmite la señal diferencial A/B, garantizando la robustez de la comunicación en un entorno industrial ruidoso. El tablero también incluye las protecciones eléctricas y las fuentes de alimentación necesarias para el funcionamiento continuo del concentrador de datos y el router local.

Fig. 6. Tablero que contiene al servidor central (Raspberry Pi).

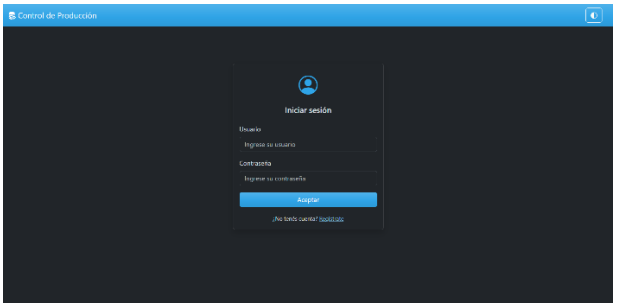
B. Pantallas del Sistema IoT

Además de las funcionalidades específicas de cada vista, la plataforma incorpora elementos transversales que enriquecen la experiencia del usuario y refuerzan su usabilidad. En la barra de navegación superior se encuentra el botón de cambio de tema (claro/oscuro), el cual es persistente mediante la sesión del usuario. Esto permite que, al volver a iniciar sesión, la interfaz mantenga automáticamente la preferencia seleccionada anteriormente, garantizando coherencia visual. Junto a él se sitúa el botón de descarga de registros, que invoca al endpoint /descargar, generando un archivo Excel con los datos del período consultado. Esta función resulta especialmente útil para análisis *offline*, informes periódicos o auditorías internas, y está implementada usando el paquete *pandas* de Python en el *backend*. Estas funcionalidades complementarias convierten al sistema no solo en una herramienta de monitoreo en tiempo real, sino también en una plataforma de consulta histórica adaptable a distintos entornos operativos.

1) Pantalla de Login: Al llegar a la *URL* de acceso, el usuario se encuentra con un formulario como se observa en la *Fig. 7*. Adoptando el tono azul característico del tema *Cerulean*. Justo encima del título "Iniciar sesión" aparece un icono de persona de *Bootstrap Icons*, reforzando el propósito de la página. En esta pantalla se dejo la barra de navegación unicamente con el boton para cambiar de tema, de modo que toda la atención se centra en la validación de credenciales. Tras pulsar "Aceptar", si la autenticación es correcta, el usuario es redirigido al *dashboard*; de lo contrario, se muestra un mensaje de error en rojo justo debajo del formulario, indicando "Usuario o contraseña inválidos".



(a)

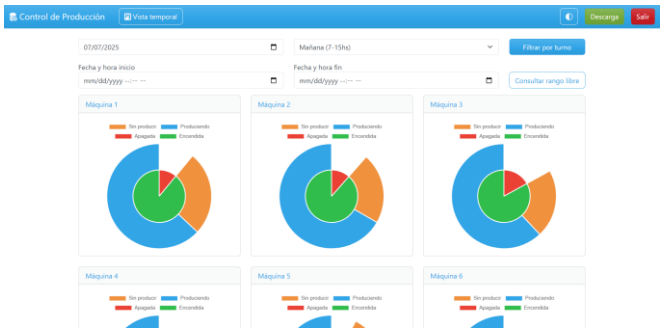


(b)

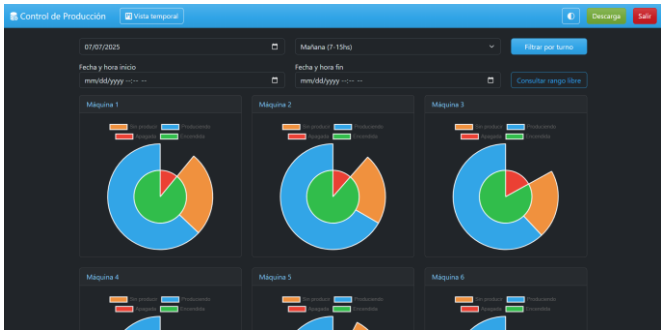
Fig. 7. Pantalla de Incio de sesión: (a) Tema claro; (b) Tema oscuro

2) Dashboard de Gráficos de Torta: Una vez dentro, la barra de navegación reaparece con el logo del proyecto a la izquierda, el selector de vista temporal (icono de *kanban*) — que nos lleva a la página de graficos temporales—, el toggle de tema y los botones de descarga y cierre de sesión a la derecha,

Fig. 8. Bajo la barra, un bloque de controles ocupa toda la anchura: a la izquierda un picker de fecha y junto a él un dropdown de turno, y a la derecha un campo de fecha-hora de inicio y otro de fin para rangos libres. El botón "Filtrar" ejecuta una consulta al endpoint "/api/vista", y "Consultar rango libre" recarga con parámetros personalizados. En cuanto llegan los datos, la sección de tarjetas se llena con una por máquina; cada una incorpora un gráfico de pastel doble —capa interna rojo-verde, capa externa naranja-azul—, una leyenda interactiva y un pequeño texto con totales numéricos. En la parte inferior de cada tarjeta aparece un icono de información que, al pasar el ratón, muestra un tooltip con la cantidad porcentual de minutos encendida y produciendo, para el periodo seleccionado.



(a)



(b)

Fig. 8. Pantalla de graficos de torta: (a) Tema claro; (b) Tema oscuro.

3) Vista Temporal de una Máquinas: Al pinchar el icono de kanban o el botón "Vista temporal" mencionado en la pantalla anterior, accedemos al endpoint "/maquinas" donde podemos ver todas las muestras para un período seleccionado como se muestra en la Fig. 9. Aquí, el navbar y el selector de tema permanecen fijos, pero los controles de fecha/turno desaparecen. Inmediatamente debajo, una hilera de botones checkbox de color secundario listan todas las máquinas; por defecto todas están activas. Cada bloque de máquina ocupa una fila con dos columnas: a la izquierda su gráfico de líneas, en el que el eje vertical discrimina tres estratos ("Apagada", "Encendida", "Produciendo") y el horizontal muestra la hora con marcas cada 1 hora; a la derecha un recuadro con un resumen numérico y porcentual del período visualizado. Cuando el usuario desmarca una máquina, su gráfico desaparece al instante, sin recargar la página. Si cambia nuevamente fecha o rango libre, los gráficos se destruyen y recrean usando Chart.js para evitar acumulación de memoria.

(a)

(b)

Fig. 9. Gráficos de muestras a lo largo del tiempo con selección de máquinas: (a) Tema claro; (b) Tema oscuro.

Es importante destacar que los valores mostrados en la Fig. 8 (Gráficos de Torta) y la Fig. 9 (Vista Temporal) se obtuvieron a partir de datos reales adquiridos durante la primera prueba piloto del sistema. Si bien la arquitectura de *hardware* y *software* se encuentra desplegada y funcional en la planta como se mostró, el sistema de visualización sigue actualmente en una fase de refinamiento funcional activo basado en requerimientos específicos del cliente. Estos ajustes esenciales incluyen la implementación de lógica para agrupar máquinas por líneas de producción y la calibración de los periodos de inactividad para excluir descansos del cómputo de tiempo improductivo, ya que no deben penalizar la eficiencia. Por esta razón, las métricas cuantitativas de desempeño del sistema (como tasas de error del bus RS-485 o tiempos de latencia del *backend*), requeridas para una validación exhaustiva, aún no se han estabilizado ni documentado, quedando pendiente su publicación para futuras iteraciones del proyecto.

VI. Conclusiones

Este trabajo presentó el diseño e implementación de una solución integral de monitoreo para plantas industriales basada en tecnologías de bajo costo, código abierto e interfaces web modernas. La arquitectura propuesta permite registrar, almacenar y visualizar el estado de producción de múltiples máquinas en tiempo real, proporcionando a los responsables de planta una herramienta confiable para la supervisión operativa.

Mediante el uso de módulos de adquisición distribuidos conectados mediante un bus de paquetes Modbus y la posterior centralización de datos en una plataforma web desarrollada con Flask, MariaDB y Node-Red, se logra una trazabilidad completa de los eventos productivos con alta granularidad temporal. El sistema contempla dos modalidades principales de visualización: gráficos porcentuales que resumen la eficiencia operativa por turno, y gráficos temporales que

permiten estudiar el comportamiento histórico de cada equipo.

Si bien el sistema no interviene directamente en los procesos productivos, constituye un paso inicial fundamental para la digitalización de las plantas. Al estructurar de forma ordenada y estandarizada la información sobre los estados de producción, se abren nuevas oportunidades para aplicar modelos de análisis de datos, detección de anomalías y predicción de ineficiencias operativas. En este sentido, la solución desarrollada puede entenderse como una base flexible y escalable hacia esquemas más avanzados de mantenimiento predictivo y control basado en datos.

Finalmente, se concluye que la implementación de este sistema en entornos industriales de la región puede contribuir significativamente a reducir pérdidas ocultas de eficiencia y acercar a las Empresas manufactureras al paradigma de la Industria 4.0. No obstante, es crucial reconocer las limitaciones actuales. El sistema se enfoca únicamente en el monitoreo pasivo, sin ejecutar acciones de control directo, y la topología *daisy-chain* del bus RS-485 sin redundancia podría presentar un único punto de fallo en entornos de alta criticidad. Estos puntos representan las líneas de trabajo a futuro.

VII. Referencias

[1] K. Schwab, "The Fourth Industrial Revolution: what it means and how to respond," World Economic Forum, Jan. 16, 2016. [Online]. Available: https://www.weforum.org/stories/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/. [Accessed: Jul. 21, 2025].

[2] 9altitudes, "Cómo el IoT está transformando la industria," 2023. [Online]. Available: https://9altitudes.com/es/aprende-y-conecta/articulos/como-el-iot-esta-transformando-la-industria. [Accessed: Jul. 21, 2025].

[3] M. O. Orquera, "Industria 4.0 en Latinoamérica: un camino hacia la transformación digital," La Comunidad Logística, Jun. 14, 2024. [Online]. Available: https://logistica.enfasis.com/ultimas-noticias/industria-4-0-en-latinoamerica-un-camino-hacia-la-transformacion-digital/. [Accessed: Jul. 21, 2025].

[4] Innovación Digital360, "IoT en la industria argentina: casos, avances y desafíos," May 27, 2025. [Online]. Available: https://www.innovaciondigital360.com/industria-4-0/iot-en-la-industria-argentina-entre-la-transformacion-y-el-desafio/. [Accessed: Jul. 21, 2025].

[5] DOEET, "Industria 4.0, la fábrica digital, flexible y optimizada," Nov. 3, 2022. [Online]. Available: https://doeet.com/industria-4-0/. [Accessed: Jul. 21, 2025].

[6] A. L. Basco, G. Beliz, D. Coatz, and P. Garnero, *Industria 4.0: fabricando el futuro*, Banco Interamericano de Desarrollo, 2018, doi: 10.18235/0001229.

[7] F. Aleman, D. G. Cortés, E. Hurtado, J. M. Paz Portilla, and E. S. Tokashiki Etcheverry, "Industria 4.0 en Pequeñas y Medianas Empresas argentinas," in Proc. 9th Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNaIISI), Buenos Aires, Argentina: Universidad Tecnológica Nacional, 2022.