



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería Electrónica y Eléctrica

Escuela Profesional de Ingeniería Electrónica

**Diseño e implementación de un sistema local y remoto
para el monitoreo y almacenamiento de variables
relacionadas a servomotores y KPIs en el proceso de
llenado de botellas**

TESIS

Para optar el Título Profesional de Ingeniera Electrónica

AUTOR

Brigitte Sarita SALAZAR GUZMÁN

ASESOR

Mg. Jean Carlos MALCA FERNÁNDEZ

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Salazar, B. (2026). *Diseño e implementación de un sistema local y remoto para el monitoreo y almacenamiento de variables relacionadas a servomotores y KPIs en el proceso de llenado de botellas*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Brigitte Sarita Salazar Guzmán
Tipo de documento de identidad	DNI
Número de documento de identidad	75602180
URL de ORCID	https://orcid.org/0009-0006-2532-5129
Datos de asesor	
Nombres y apellidos	Jean Carlos Malca Fernández
Tipo de documento de identidad	DNI
Número de documento de identidad	45535181
URL de ORCID	https://orcid.org/0000-0001-5633-5182
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Nicanor Raúl Benites Saravia
Tipo de documento	DNI
Número de documento de identidad	10189914
Miembro del jurado 1	
Nombres y apellidos	Jorge Elías Moscoso Sánchez
Tipo de documento	DNI
Número de documento de identidad	07206008
Miembro del jurado 2	
Nombres y apellidos	Roberto Valeriano Mandujano Mieses
Tipo de documento	DNI
Número de documento de identidad	08336832

Datos de investigación	
Línea de investigación	C.0.4.8 Automatización Industrial
Grupo de investigación	No aplica
Agencia de financiamiento	No aplica
Ubicación geográfica de la investigación	Edificio: Facultad de Ingeniería Electrónica y Eléctrica País: Perú Departamento: Lima Provincia: Lima Distrito: Lima Urbanización: Ciudad Universitaria - Universidad Nacional Mayor de San Marcos Calle: Av. Germán Amézaga s/n. Latitud: -12.05503 Longitud: -77.08684
Año o rango de años en que se realizó la investigación	Noviembre 2024 – Mayo 2025 2024-2025
URL de disciplinas OCDE	Sistemas de automatización, Sistemas de control https://purl.org/pe-repo/ocde/ford#2.02.03



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú. Decana de América
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

ACTA DE SUSTENTACIÓN DE TESIS N°001-EPIE-FIEE/2026

Los suscritos miembros del jurado, docentes permanentes de las Escuelas Profesionales de la Facultad de Ingeniería Electrónica y Eléctrica, reunidos en la fecha 9 de enero del 2026, como presidente de jurado el **Dr. NICANOR RAÚL BENITES SARA VIA**, integrado por el miembro de jurado el **Dr. JORGE ELÍAS MOSCOSO SÁNCHEZ**, el miembro de jurado el **Mg. ROBERTO VALERIANO MANDUJANO MIESES** y miembro asesor el **Mg. JEAN CARLOS MALCA FERNÁNDEZ**.

Después de escuchar la Sustentación Presencial de Tesis de la

Bachiller **BRIGGITTE SARITA SALAZAR GUZMÁN** código **N°18190028**, para optar el Título Profesional de Ingeniera Electrónica sustentó la tesis titulada: **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA LOCAL Y REMOTO PARA EL MONITOREO Y ALMACENAMIENTO DE VARIABLES RELACIONADAS A SERVOMOTORES Y KPIS EN EL PROCESO DE LLENADO DE BOTELLAS"**.

El jurado examinador procedió a formular las preguntas reglamentarias y luego de una deliberación en privado, decidió otorgarle el calificativo de 18 (DIECIOCHO)

Ciudad Universitaria, 9 de enero del 2026

Dr. NICANOR RAÚL BENITES SARA VIA
Presidente Jurado

Dr. JORGE ELÍAS MOSCOSO SÁNCHEZ
Miembro Jurado

Mg. ROBERTO VALERIANO MANDUJANO
MIESES
Miembro Jurado

Mg. JEAN CARLOS MALCA FERNÁNDEZ
Miembro Asesor



Anexo 1

CERTIFICADO DE SIMILITUD

Yo, **Jean Carlos Malca Fernández** en mi condición de asesor acreditado con el Dictamen de la EP N° 001-EPIE-2026-FIEE/UNMSM de la tesis, cuyo título es **‘DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA LOCAL Y REMOTO PARA EL MONITOREO Y ALMACENAMIENTO DE VARIABLES RELACIONADAS A SERVOMOTORES Y KPIS EN EL PROCESO DE LLENADO DE BOTELLAS’**, presentado por la bachiller **Brigitte Sarita Salazar Guzmán** para optar el **Título profesional de Ingeniera Electrónica**, CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el **7%** de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del **Título profesional de Ingeniera Electrónica**.

Lima, 16 de Enero de 2026

Firma del Asesor: _____

Nombres y Apellidos del asesor: **Jean Carlos Malca Fernández**
DNI: **45535181**

* Según corresponda: bachiller, magíster, doctor; título profesional o título de segunda especialidad

La vigencia del presente certificado corresponde a un período de noventa (90) días desde la fecha de su emisión; siempre que no haya observaciones en el texto durante el proceso de revisión por el jurado informante o surja alguna contingencia.

DEDICATORIA

A mi madre por su constante esmero y apoyo en mi formación profesional y personal, a mi padre por enseñarme con el ejemplo, la determinación y el esfuerzo, y a mi hermano por guiarme durante toda mi formación.

Brigitte Salazar.

AGRADECIMIENTO

Agradezco a la universidad que financio el Módulo de Motion Control a través del proyecto de investigación del grupo TAS con RR-015156 2020-R/UNMSM y código C20190500a.

Al asesor Jean Carlos Malca por el apoyo y guía durante este trabajo de investigación.

A Félix Ospina y Jesús Aquino por su apoyo y orientación durante mi proceso de aprendizaje en automatización industrial.

RESUMEN

El presente estudio tiene por finalidad desarrollar un sistema local y remoto para monitorear y almacenar las variables de velocidad, posición y torque de los servomotores, así como los indicadores clave de rendimiento (KPIs) relevantes en los escenarios de llenado de botellas, tales como la tasa de utilización de capacidad, tasa de calidad, tasa de disponibilidad y productividad.

El sistema permitirá observar y almacenar localmente las variables de los servomotores mediante una interfaz desarrollada en WinCC Professional que, a su vez, mediante diversos scripts, estará enlazada con el programa SQL Server. Esta conexión permitirá crear o eliminar bases de datos y tablas, definir las variables a almacenar y establecer un rango de tiempo para ejecutar el monitoreo.

De manera remota, el sistema almacenará en InfluxDB Cloud las variables necesarias para calcular los KPIs, las cuales serán posteriormente enviadas a Grafana Cloud, con el fin de aplicar fórmulas automatizadas para obtener dichos indicadores y visualizar los resultados mediante un dashboard interactiva y de alta disponibilidad, que brindará un gran aporte para la toma de decisiones.

Palabras claves: Monitoreo y almacenamiento de datos, sistema local, sistema remoto, automatización, proceso de llenado de botellas.

ABSTRACT

The purpose of this study is to develop a local and remote system to monitor and store the variables of speed, position, and torque of the servomotors, as well as the relevant Key Performance Indicators (KPIs) in bottle-filling scenarios, such as capacity utilization rate, quality rate, availability rate, and productivity.

The system will allow for local observation and storage of the servomotor variables through an interface developed in WinCC Professional, which, in turn, will be linked to the SQL Server program via various scripts. This connection will enable the creation or deletion of databases and tables, the definition of variables to be stored, and the establishment of a time range for monitoring execution.

Remotely, the system will store in InfluxDB Cloud the variables required to calculate the KPIs, which will later be sent to Grafana Cloud. There, automated formulas will be applied to obtain these indicators and the results will be visualized through an interactive, high-availability dashboard, providing significant support for decision-making.

Keywords: Data monitoring and storage, local system, remote system, automation, bottle-filling process.

ÍNDICE GENERAL

I. INTRODUCCIÓN.....	1
I.1. Introducción.....	1
I.2. Planteamiento del problema	3
I.2.1. Determinación del problema	3
I.2.2. Formulación del problema	5
I.2.2.1. Problema general.....	5
I.2.2.2. Problemas específicos.	5
I.3. Objetivos	6
I.3.1. Objetivo general	6
I.3.2. Objetivos específicos.....	6
I.4. Importancia y alcance de la investigación.....	7
I.5. Limitaciones de la investigación	7
II. REVISIÓN DE LITERATURA	9
II.1. Marco teórico	9
II.2. Antecedentes del estudio	10
II.3. Bases teóricas	11
II.3.1. Control de movimiento.....	11
II.3.1.1. Controlador Lógico Programable (PLC).....	13
II.3.1.2. Servodrive.	20
II.3.1.3. Servomotor.	23
II.3.2. Nivel de supervisión en la automatización industrial.....	30
II.3.2.1. Sistemas supervisión, control y adquisición de datos (SCADA).....	30
II.3.3. Indicadores Clave de Rendimiento (KPIs).....	34
II.3.3.1. ISO 9001:2015 – Sistemas de gestión de la calidad.	34
II.3.3.2. KPIs en la industria..	35
II.3.3.3. Cálculo y visualización de KPIs.	36
II.3.4. Almacenamiento y gestión de datos.....	37
II.3.4.1. SQL server.	37

II.3.4.2. InfluxDB Cloud.....	38
II.3.5. Proceso de elaboración de bebidas.....	39
II.3.5.1. Llenado de botellas.	40
II.4. Definición de términos.....	43
II.4.1. Plataformas remotas o en la nube.....	43
II.4.2. Node-RED.....	43
II.4.3. Dashboard.....	44
III. HIPÓTESIS Y VARIABLES.....	45
III.1. Hipótesis.....	45
III.1.1. Hipótesis general.....	45
III.1.2. Hipótesis específicas.....	45
III.2. Variables.....	46
III.2.1. Variable independiente.....	46
III.2.2. Variable dependiente.....	46
III.3. Operacionalización de las variables.....	47
IV. MATERIALES Y MÉTODOS.....	48
IV.1. Área de estudio.....	48
IV.2. Diseño de la investigación.....	48
IV.3. Población y muestra.....	49
IV.3.1. Población.....	49
IV.3.2. Muestra.....	49
IV.4. Procedimiento, técnicas e instrumentos de recolección de información.....	49
IV.4.1. Descripción del sistema propuesto.....	49
IV.4.1.1. Propuesta de sistema local.....	53
IV.4.1.2. Propuesta de sistema remoto.....	57
IV.4.2. Implementación del sistema propuesto.....	59
IV.4.2.1. Implementación del sistema local.....	59
IV.4.2.2. Implementación del sistema remoto.....	86
V. RESULTADOS.....	99
V.1. Presentación y análisis de los resultados.....	99
V.1.1. Sistema local.....	99
V.1.2. Sistema remoto.....	123

VI. DISCUSIÓN	137
VII. CONCLUSIONES Y RECOMENDACIONES	138
VII.1. Conclusiones	138
VII.2. Recomendaciones	140
Referencias.....	141
Anexo A Matriz de consistencia.....	149
Anexo B Líneas de código del Script <i>CrearDB</i>	150
Anexo C Líneas de código del Script <i>CrearTabla</i>	151
Anexo D Líneas de código del Script <i>BorrarDB</i>	152
Anexo E Líneas de código del Script <i>BorrarTabla</i>	153
Anexo F Líneas de código del Script <i>InsertarDataMAN</i>	154
Anexo G Líneas de código del Script <i>InsertarDataAUTO</i>	155
Anexo H Líneas de código del Script <i>MostrarTabla</i>	156
Anexo I Líneas de código del Script <i>DBrefres</i>	157
Anexo J Líneas de código del Script <i>TablaporDBrefresh</i>	158
Anexo K Líneas de código del Script <i>ColumnaRefreshDBdinamico</i>	159

LISTA DE CUADROS

<i>Cuadro 1:</i> Operacionalización de las variables.	47
<i>Cuadro 2:</i> Pseudocódigo del script CrearDB.....	70
<i>Cuadro 3:</i> Pseudocódigo del script CrearTabla.	71
<i>Cuadro 4:</i> Pseudocódigo del script BorrarDB.	73
<i>Cuadro 5:</i> Pseudocódigo del script BorrarTabla.	74
<i>Cuadro 6:</i> Pseudocódigo del script InsertarDataMAN.	76
<i>Cuadro 7:</i> Pseudocódigo del script InsertarDataAUTO.	78
<i>Cuadro 8:</i> Pseudocódigo del script MostrarTabla.	80
<i>Cuadro 9:</i> Pseudocódigo del script DBrefres.	82
<i>Cuadro 10:</i> Pseudocódigo del script TablaporDBRefresh.....	83
<i>Cuadro 11:</i> Pseudocódigo del script ColumnaRefreshDBdinamico.	85

LISTA DE FIGURAS

<i>Figura 1.</i> Vista de la CPU 1511C-1 PN con tapas frontales cerradas.....	14
<i>Figura 2.</i> Vista de la CPU 1511C-1 PN sin tapa frontal... ..	14
<i>Figura 3.</i> Comunicación PROFINET entre el PLC y servodrive.....	17
<i>Figura 4.</i> La integración de control de movimiento en la CPU S7-1500.....	18
<i>Figura 5.</i> Vistas de los objetos tecnológicos.	19
<i>Figura 6.</i> Vistas de parámetros en los objetos tecnológicos.....	19
<i>Figura 7.</i> Partes del servodrive.. ..	21
<i>Figura 8.</i> Funcionamiento del sistema.....	22
<i>Figura 9.</i> Conectores de servomotor.....	22
<i>Figura 10.</i> Conexión entre el servodrive y servomotor.	23
<i>Figura 11.</i> Servomotor del fabricante Siemens.	23
<i>Figura 12.</i> Principio de encoder.....	24
<i>Figura 13.</i> Señales de encoder.	25
<i>Figura 14.</i> Partes del servomotor.	26
<i>Figura 15.</i> Eje con husillo.	27
<i>Figura 16.</i> Sistema de engranajes.. ..	28
<i>Figura 17.</i> Casos de aplicación de torque.....	29
<i>Figura 18.</i> Sistema SCADA.	31
<i>Figura 19.</i> Componentes.....	33
<i>Figura 20.</i> Panel de Grafana.	37
<i>Figura 21.</i> Explorador de objetos en SSMS.	38
<i>Figura 22.</i> Datos en formato de tabla en InfluxDB Cloud.	39
<i>Figura 23.</i> Proceso de elaboración de bebidas.....	40

<i>Figura 24.</i> Sistema de llenado a nivel.....	41
<i>Figura 25.</i> Boquillas con sensores en la llenadora volumétrica.	42
<i>Figura 26.</i> Sistema de llenadora ponderal.	42
<i>Figura 27.</i> Interfaz de Node-RED.	44
<i>Figura 28.</i> Sistema propuesto.	50
<i>Figura 29.</i> Prototipo de llenado de botellas.	51
<i>Figura 30.</i> CPU 1511C-1 PN (6ES7511-1CK01-0AB0).....	52
<i>Figura 31.</i> SINAMICS V90 (6SL3210-5FB10-2UF2).....	52
<i>Figura 32.</i> SIMOTICS S-1FL6 (1FL6032-2AF2x-xAA\Gx).....	53
<i>Figura 34.</i> TIA Portal V17.	54
<i>Figura 35.</i> Logo de SQL Server.	55
<i>Figura 36.</i> SQL Server Management Studio (SSMS).	55
<i>Figura 37.</i> Sistema local.	56
<i>Figura 38.</i> Enlace de Node-RED con TIA Portal y InfluxDB Cloud.	57
<i>Figura 39.</i> Centro de recursos de InfluxDB Cloud.....	57
<i>Figura 40.</i> Recursos en Grafana Cloud.	58
<i>Figura 41.</i> Sistema remoto.....	58
<i>Figura 42.</i> Cable ethernet.	59
<i>Figura 43.</i> Esquema de red PROFINET.	60
<i>Figura 44.</i> Objetos tecnológicos de cada servomotor.....	61
<i>Figura 45.</i> Inicio de sesión de SQL Server Management Studio 20 (SSMS20).....	62
<i>Figura 46.</i> Bases de datos y tablas visualizadas desde SSMS20.....	63
<i>Figura 47.</i> Añadir una PC station.	64
<i>Figura 48.</i> Configuración de la PC station.	64
<i>Figura 49.</i> Diagrama de imágenes a usar.....	65

<i>Figura 50.</i> Pantalla de inicio.....	65
<i>Figura 52.</i> DB para almacenar variables de la faja y llenadora.....	66
<i>Figura 53.</i> Almacenamiento de los valores actuales de la velocidad, torque y posición de la faja.....	66
<i>Figura 54.</i> Almacenamiento de los valores actuales de la velocidad, torque y posición del llenador o dispensador.....	67
<i>Figura 55.</i> Pantalla de control automático.....	68
<i>Figura 56.</i> Pantalla de creación de DBs y tablas.....	69
<i>Figura 57.</i> Pantalla de borrado de DBs y tablas.....	72
<i>Figura 58.</i> Pantalla de ingreso de datos.....	75
<i>Figura 59.</i> Pantalla de vista de tabla.....	79
<i>Figura 60.</i> Pantalla de DBs y tablas disponibles.....	81
<i>Figura 61.</i> Bloque de datos (DB) creado en TIA Portal.....	86
<i>Figura 62.</i> Contador de total de botellas.....	87
<i>Figura 63.</i> Total de botellas en buen estado.....	87
<i>Figura 64.</i> FB Minutos.....	88
<i>Figura 65.</i> Segmento 1 del FB <i>Minutos</i>	88
<i>Figura 67.</i> Segmento 3 del FB <i>Minutos</i>	89
<i>Figura 68.</i> Configuración del nodo S7 – Parte 1.....	89
<i>Figura 69.</i> Configuración del nodo S7 – Parte 2.....	90
<i>Figura 70.</i> Configuración del nodo S7 – Parte 3.....	90
<i>Figura 71.</i> Creación del bucket en InfluxDB Cloud.....	91
<i>Figura 72.</i> Bucket disponible en InfluxDB Cloud.....	91
<i>Figura 73.</i> Generación del API Token en InfluxDB Cloud.....	91
<i>Figura 74.</i> API Token en InfluxDB Cloud.....	92
<i>Figura 75.</i> Configuración del nodo Influxdb out– Parte 1.....	92

<i>Figura 76.</i> Configuración del nodo Influxdb out– Parte 2.....	92
<i>Figura 77.</i> Generación de un segundo API Token.	93
<i>Figura 78.</i> Configuración de fuente de datos en Grafana Cloud – Parte 1.	93
<i>Figura 79.</i> Configuración de fuente de datos en Grafana Cloud – Parte 2.	94
<i>Figura 80.</i> Configuración de fuente de datos en Grafana Cloud – Parte 3.	94
<i>Figura 81.</i> Script para el cálculo de la tasa de utilización de capacidad.	95
<i>Figura 82.</i> Script para el cálculo de la tasa de calidad – Parte 1.	95
<i>Figura 83.</i> Script para el cálculo de la tasa de calidad – Parte 2.	96
<i>Figura 84.</i> Script para el cálculo de la tasa de disponibilidad.	96
<i>Figura 85.</i> Script para el cálculo de la productividad – Parte 1.....	97
<i>Figura 86.</i> Script para el cálculo de la productividad – Parte 2.....	97
<i>Figura 87.</i> Creación de la base de datos <i>FAJA</i>	100
<i>Figura 88.</i> Selección de columnas para la tabla <i>SERVO1</i> perteneciente a la base de datos <i>FAJA</i>	100
<i>Figura 89.</i> Creación de la tabla <i>SERVO1</i> perteneciente a la base de datos <i>FAJA</i>	101
<i>Figura 90.</i> Creación de la base de datos <i>LLENADOR</i>	102
<i>Figura 91.</i> Selección de columnas para la tabla <i>SERVO2</i> perteneciente a la base de datos <i>LLENADOR</i>	102
<i>Figura 92.</i> Creación de la tabla <i>SERVO2</i> perteneciente a la base de datos <i>LLENADOR</i>	103
<i>Figura 93.</i> Demo en funcionamiento.	103
<i>Figura 94.</i> Almacenamiento de datos en modo manual.	104
<i>Figura 96.</i> Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 1.....	106
<i>Figura 97.</i> Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 1.	107

<i>Figura 98.</i> Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 2.....	108
<i>Figura 99.</i> Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 2.	109
<i>Figura 100.</i> Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 3.....	110
<i>Figura 101.</i> Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 3.	111
<i>Figura 102.</i> Visualización de variables dinámicas del llenador en la pantalla desarrollada en WinCC Professional – Caso 1.....	112
<i>Figura 103.</i> Variables dinámicas del llenador mostradas en SQL Server Management Studio 20 (Caso 1).....	113
<i>Figura 104.</i> Visualización de variables dinámicas del llenador en la interfaz desarrollada en WinCC Professional (Caso 2).	114
<i>Figura 105.</i> Visualización de variables dinámicas del llenador en SQL Server Management Studio 20 (SSMS 20) (Caso 2).....	115
<i>Figura 106.</i> Visualización de variables dinámicas del llenador en la interfaz desarrollada en WinCC Professional (Caso 3).	116
<i>Figura 107.</i> Visualización de variables dinámicas del llenador en SQL Server Management Studio 20 (SSMS 20) (Caso 3).....	117
<i>Figura 108.</i> Bases de datos disponibles.....	118
<i>Figura 109.</i> Tablas disponibles según base de dato seleccionada.	118
<i>Figura 110.</i> Caso en que no se brinde datos de búsqueda.	119
<i>Figura 111.</i> Columnas disponibles según tabla seleccionada.....	119
<i>Figura 112.</i> Borrado de tabla.	120
<i>Figura 113.</i> Falta de datos para iniciar el borrado de una base de datos o tablas - Caso 1.....	121

<i>Figura 114.</i> Falta de datos para iniciar el borrado de una base de datos o tablas - Caso 2.....	121
<i>Figura 115.</i> Borrado de una tabla inexistente.....	122
<i>Figura 116.</i> Tabla de datos mostrados en una pantalla desarrollada en WinCC Professional.....	122
<i>Figura 117.</i> Bloque de datos <i>Var_KPI</i> [DB30] en TIA Portal.....	123
<i>Figura 118.</i> Envío de datos a Influx DB Cloud desde Node - RED.....	124
<i>Figura 119.</i> Datos recepcionados por InfluxDB Cloud.....	125
<i>Figura 120.</i> Períodos de toma de datos en Node-RED a analizar.....	126
<i>Figura 121.</i> Recepción de datos de Node-RED – Parte 1.....	127
<i>Figura 122.</i> Recepción de datos de Node-RED – Parte 2.....	127
<i>Figura 123.</i> Tasa de utilización de capacidad – Caso 1.....	128
<i>Figura 124.</i> Tasa de disponibilidad – Caso 1.....	129
<i>Figura 125.</i> Tasa de calidad – Caso 1.....	129
<i>Figura 126.</i> Productividad – Caso 1.....	130
<i>Figura 128.</i> Tasa de utilización de capacidad – Caso 2.....	132
<i>Figura 129.</i> Tasa de disponibilidad – Caso 2.....	132
<i>Figura 130.</i> Tasa de calidad – Caso 2.....	133
<i>Figura 131.</i> Productividad – Caso 2.....	134
<i>Figura 132.</i> Dashboard realizado – Parte 1.....	134
<i>Figura 133.</i> Dashboard realizado – Parte 2.....	135
<i>Figura 134.</i> Dashboard realizado – Parte 3.....	135
<i>Figura 135.</i> Dashboard realizado – Parte 4.....	136

I. INTRODUCCIÓN

I.1. Introducción

La industria de bebidas se distingue por su dinamismo y exigencia dentro del sector manufacturero. Maneja altos volúmenes de producción, cumple estrictos estándares de calidad y mantiene una eficiencia operativa constante. Para lograrlo, este sector ha incorporado procesos altamente automatizados que aseguran uniformidad, inocuidad alimentaria y rapidez. Ante una demanda cambiante y regulaciones cada vez más rigurosas, resulta imprescindible implementar sistemas avanzados de monitoreo y almacenamiento que permitan sostener la competitividad.

Dentro de este contexto, el proceso de llenado de botellas representa una etapa crítica en la línea de producción. Las botellas son trasladadas a través de cintas automatizadas hasta las estaciones de llenado, donde se dosifica con precisión el contenido para evitar desperdicios o sobrellenado. Este proceso es altamente sensible a errores mecánicos, pequeñas variaciones en el ritmo de trabajo y retrasos mínimos, los cuales pueden afectar la eficiencia, la calidad del producto y la continuidad de la operación.

En este escenario, el monitoreo de las variables de los servomotores resulta esencial, ya que garantiza un movimiento preciso y sincronizado, permitiendo que cada botella sea ubicada y llenada correctamente. Este seguimiento constante también facilita la detección de desviaciones y el ajuste eficiente de parámetros operativos evitando paradas no programadas y reduciendo el desgaste de los equipos. Atakok et al.

(2020) señalan que el mantenimiento predictivo en servomotores, basado en el análisis de datos, permite anticipar fallos y optimizar la eficiencia operativa.

Es común que, para revisar las variables relacionadas con los servomotores, sea necesario establecer una conexión con el PLC para luego identificar los actuadores de interés y sus telegramas. Sin embargo, para ello, el usuario debe contar con el software y la versión compatible con el PLC, así como conocimientos sobre las palabras de estados, consignas, conversiones u otros datos relevantes para obtener las variables de los servomotores en las unidades deseadas.

No obstante, existen fabricantes que cuentan con software exclusivo para conectarse directamente con sus servomotores y que permitan su monitoreo. Sin embargo, esto puede resultar complicado, ya que, en las zonas de llenado de botellas, los servomotores provienen de diferentes proveedores. En muchos casos, sus programas ya no están vigentes, requieren pagos anticipados elevados, están en idiomas diferentes a la lengua nativa del usuario o presentan una interfaz confusa. Es en este punto donde pueden ocurrir graves errores, ya que estos programas también permiten modificar parámetros. Entonces, el usuario, en su intento de monitorear, podría alterar alguna variable imprescindible para el correcto funcionamiento sincronizado de los servomotores. Otro inconveniente se presenta durante el intento de conexión con el servomotor, debido a que los equipos suelen estar inaccesibles o protegidos con cubiertas.

Ante ello, esta tesis busca diseñar e implementar un sistema local de monitoreo y almacenamiento de las variables dinámicas de servomotores como la velocidad, posición y torque. En este sistema, el usuario podrá visualizar las variables de interés de manera ordenada, directa y sin conexiones previas, desde una pantalla desarrollada en WinCC Professional. Además, mediante esta pantalla podrá seleccionar que variables se almacenarán en las respectivas bases de datos (DBs) y tablas de SQL Server.

Por otro lado, evaluar el rendimiento del proceso de llenado es fundamental, puesto que se trata de una producción continua y rápida, que genera grandes volúmenes de datos. Para ello, se requiere recopilar las variables necesarias para el cálculo de los indicadores clave de desempeño (KPIs), tales como la disponibilidad, productividad, la calidad del producto y el aprovechamiento de la capacidad. Según Ultatek (2020), resulta esencial automatizar y agilizar este cálculo contar con información precisa, actualizada y confiable que facilite la toma de decisiones operativas y estratégicas.

Por ello, se propone el diseño e implementación de un sistema remoto de monitoreo y almacenamiento de variables relacionadas con los KPIs. Las variables necesarias para el cálculo se almacenarán en una base de datos en InfluxDB Cloud y, posteriormente, se enviarán a Grafana Cloud, donde se le aplicarán las fórmulas automatizadas para calcular los KPIs. Los resultados podrán ser monitoreados a través de un dashboard que incluye diversas herramientas visuales, las cuales ofrecen flexibilidad para adaptarse a las necesidades del usuario.

I.2. Planteamiento del problema

I.2.1. Determinación del problema

Las industrias de bebidas operan con procesos que se caracterizan por el alto control, rapidez, precisión, salubridad y seguridad. Por ello, las líneas de producción utilizan sistemas basados en servomotores, cilindros neumáticos, sensores fotoeléctricos, sensores de nivel, bombas centrífugas, interfaz hombre-máquina (HMI), sistemas safety, entre otros componentes.

En este contexto, los servomotores cumplen la función de sincronización para lograr un llenado homogéneo y un transporte seguro, con el fin de evitar caídas, roturas y derrames de botellas, ya que este tipo de incidentes podría generar pérdidas

económicas, considerando que la producción es elevada en un corto periodo de tiempo y que, al tratarse de un producto de consumo humano, se debe retirar tanto la botella afectada como las unidades adyacentes que pudieran contaminarse con líquido o fragmentos de vidrio.

Frente a esta problemática, resulta esencial establecer correctamente las variables dinámicas de los servomotores, realizar pruebas y registrar los valores de los parámetros durante los ciclos de ejecución. Esto permite lograr una parametrización adecuada y diagnosticar rápidamente la causa de una falla en determinado punto. En respuesta a esta necesidad, se plantea diseñar e implementar un sistema de monitoreo y almacenamiento local, basado en una pantalla desarrollada en WinCC Professional y SQL Server.

En este tipo de industrias, monitorear los KPIs es fundamental para mejorar la eficiencia, calidad y productividad, con el objetivo de optimizar la toma de decisiones estratégicas sobre inversiones de equipos o reajustes en las líneas de producción, promoviendo así una mejora continua frente a la alta competencia del mercado. Sin embargo, aún existen plantas que cuentan con procesos semiautomáticos para el cálculo de KPIs, o utilizan plataformas con altos costos iniciales, complejas de implementar y con opciones de personalización limitadas.

Por ello, esta investigación también se enfocará en el diseño e implementación de un sistema remoto de monitoreo y almacenamiento de KPIs, utilizando InfluxDB Cloud y Grafana Cloud, plataformas que se complementan entre sí y que reducen la intervención de herramientas externas para el envío y procesamiento de datos entre ellas. De este modo, se logra un cálculo de KPIs más ágil y preciso.

I.2.2. Formulación del problema

I.2.2.1. Problema general.

¿Cómo diseñar e implementar un sistema local y remoto que permita el monitoreo y almacenamiento de variables asociadas a servomotores y KPIs en el proceso de llenado de botellas?

I.2.2.2. Problemas específicos.

- a. ¿Cómo implementar un sistema local de monitoreo de las variables dinámicas de los servomotores, a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas?
- b. ¿Cómo implementar un sistema local de almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas?
- c. ¿Cómo implementar un sistema remoto de monitoreo de KPIs, mediante un panel visual desarrollado en Grafana Cloud, en el proceso de llenado de botellas?
- d. ¿Cómo implementar un sistema remoto de almacenamiento de las variables necesarias para el cálculo de KPIs en InfluxDB Cloud, en el proceso de llenado de botellas?

I.3. Objetivos

I.3.1. Objetivo general

Diseñar e implementar un sistema local y remoto para el monitoreo y almacenamiento de variables asociadas a servomotores y KPIs en el proceso de llenado de botellas.

I.3.2. Objetivos específicos

- a. Implementar un sistema local de monitoreo de las variables dinámicas de los servomotores a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas.
- b. Implementar un sistema local para el almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas.
- c. Implementar un sistema remoto de monitoreo de KPIs, mediante un panel visual desarrollado en Grafana Cloud, en el proceso de llenado de botellas.
- d. Implementar un sistema remoto para el almacenamiento de las variables necesarias para el cálculo de KPIs, utilizando InfluxDB Cloud, en el proceso de llenado de botellas.

I.4. Importancia y alcance de la investigación

Esta investigación es relevante porque responde a la necesidad de contar con sistemas de monitoreo y almacenamiento de datos que optimicen los procesos de automatización industrial. El estudio aporta beneficios al reducir los tiempos de respuesta mediante el uso de plataformas locales y aplicaciones en la nube, las cuales ofrecen alta disponibilidad, interactividad y flexibilidad.

Asimismo, brinda aportes tecnológicos a través de la construcción de códigos para el manejo de datos, generando automatización y orden en los procesos. De esta manera, se contribuye al fortalecimiento de las herramientas de diagnóstico y a la mejora en la toma de decisiones estratégicas dentro del ámbito industrial.

El presente estudio desarrollará una aplicación interactiva de monitoreo y almacenamiento local, que permitirá al usuario decidir cuándo iniciar la toma de datos, eliminarlos, seleccionar las variables a observar y definir las bases de datos y tablas desde una interfaz intuitiva. En el ámbito remoto, se implementará un sistema de almacenamiento en la nube con alta integración, complementado con un panel visual personalizado que se actualizará de forma continua. Este panel mostrará indicadores relevantes para el proceso de llenado de botellas y facilitará un análisis ágil y eficiente. Cabe precisar que el estudio se centra en el desarrollo y la integración de software y plataformas de monitoreo y almacén.

I.5. Limitaciones de la investigación

Una de las limitaciones encontradas fue la escasa literatura o antecedentes sobre configuraciones complejas de bases de datos y tablas mediante scripts desarrollados en WinCC Professional, así como la estructuración de códigos en Grafana Cloud para la

visualización de resultados. Sin embargo, se logró desarrollar códigos específicos que permitieron un funcionamiento flexible y eficiente de los sistemas propuestos. En ese sentido, el sistema no será implementado en una línea de producción real, sino en un prototipo o demo de llenado de botellas perteneciente al grupo de investigación TAS de la Facultad de Ingeniería Electrónica y Eléctrica.

II. REVISIÓN DE LITERATURA

II.1. Marco teórico

En este capítulo se presentan los conceptos y fundamentos teóricos que sirven de base para el diseño e implementación del sistema local y remoto de monitoreo y almacenamiento de variables de servomotores y KPIs en el proceso de llenado de botellas. El propósito de este apartado es explicar las ideas y principios que sustentan el proyecto y justificar las decisiones tomadas respecto al uso de ciertas tecnologías y métodos.

El marco teórico cumple un papel importante dentro del trabajo, ya que permite entender el problema de investigación y cómo las variables del sistema, como la velocidad, la posición y el torque de los servomotores, se relacionan con los indicadores clave de rendimiento que ayudan a medir la eficiencia del proceso. De esta manera, se establece una conexión entre los aspectos teóricos y la aplicación práctica dentro del entorno industrial.

En cuanto a su estructura, el capítulo está dividido en diferentes secciones. Primero se desarrollan las bases teóricas del control de movimiento, donde se explican los principales componentes del sistema, como el PLC, el servodrive y el servomotor. Luego se analiza el nivel de supervisión en la automatización industrial, haciendo énfasis en los sistemas SCADA, que permiten la adquisición y el control de datos.

Después se abordan los indicadores clave de rendimiento, su relación con la norma ISO 9001:2015 y la forma en que se aplican dentro del entorno industrial. También se

describe cómo se calculan y visualizan estos indicadores para evaluar el desempeño del sistema. En la siguiente parte se trata el tema del almacenamiento y la gestión de datos, mencionando el uso de SQL Server para la gestión local e InfluxDB Cloud para el almacenamiento remoto. Además, se incluye una descripción del proceso de elaboración de bebidas, con especial atención en la etapa de llenado de botellas, que constituye el punto central del proyecto.

Finalmente, se presenta una sección dedicada a la definición de términos, donde se explican los conceptos y herramientas más relevantes para la comprensión del trabajo, como las plataformas en la nube, Node-RED y el Dashboard, que permiten visualizar y supervisar los datos del sistema en tiempo real.

La elaboración de este marco teórico es fundamental, ya que proporciona el respaldo necesario para el desarrollo del proyecto. Además, permite sustentar las decisiones de diseño a partir de teorías, normas y experiencias previas, garantizando que la propuesta sea viable, confiable y coherente con las tendencias actuales de la automatización industrial.

II.2. Antecedentes del estudio

Existen estudios enfocados en el uso de soluciones gestionadas en la nube, como la investigación realizada por Rímac (2024), la cual explica el diseño de una arquitectura IoT orientada a monitorear en tiempo real la calidad del aire mediante sensores de bajo costo, aplicando los protocolos LoRaWAN y MQTT, así como plataformas como InfluxDB para el almacenamiento de datos y Grafana para la visualización de alarmas relacionadas con la contaminación. Este estudio aporta información sobre sistemas de almacenamiento de datos y paneles remotos, los cuales constituyen uno de los objetivos principales del presente trabajo de investigación.

De igual manera, la investigación de Peña (2022) emplea InfluxDB y Grafana, en conjunto con JMeter y Jenkins, para implementar servicios de pruebas de rendimiento en sistemas basados en la web. El objetivo es reducir los tiempos de respuesta, disminuir los errores y garantizar la calidad esperada por el usuario final. En este estudio, InfluxDB se utiliza para almacenar los datos resultantes de las pruebas, mientras que Grafana permite configurar gráficos que facilitan el análisis del comportamiento del sistema durante el período de evaluación. Este proyecto demuestra la integración entre plataformas remotas y aporta conocimientos valiosos para optimizar el envío de datos.

Por otro lado, esta tesis aborda el monitoreo basado en una pantalla de Siemens complementada con almacenamiento, como es el caso del trabajo de Echevarría (2019), que implementa una base de datos para un proceso de transporte de agua, la cual es monitoreada y controlada mediante un PLC S7-1200, una placa electrónica y un HMI desde donde se envían los registros de las variables hacia la base de datos. Para mostrar una interfaz didáctica y de alta disponibilidad, el investigador desarrolla una aplicación que permite al usuario controlar el sistema desde cualquier punto geográfico.

II.3. Bases teóricas

En este apartado se desarrollan los fundamentos técnicos necesarios para sustentar la propuesta del sistema. Se abordan aspectos vinculados al control de movimiento, la supervisión en entornos industriales, los indicadores clave de desempeño y el manejo de datos, así como el proceso de llenado de botellas, que es el escenario donde se implementará la solución planteada.

II.3.1. Control de movimiento

El control de movimiento es el conjunto de sistemas que permiten controlar la velocidad, posición, par de motores y ejes de máquinas, con el fin de obtener

movimientos sincronizados y precisos. Suele estar compuesto por servomotores, sensores para la retroalimentación y controladores que procesan la información y ajustan el movimiento. El motor y el componente que mueve se denomina eje, y es común que una máquina tenga varios ejes que deban operar de forma sincronizada, como ocurre en el caso de una máquina CNC.

Este tipo de control incrementa la vida útil de los sistemas mecánicos debido a sus perfiles suaves de aceleración y desaceleración, que evitan un gran desgaste en componentes como guías lineales, engranajes, entre otros. Además, reduce los gastos energéticos y de consumo, puesto que devuelve la energía a la red durante el proceso de frenado regenerativo; es decir, durante el frenado el motor actúa como generador, convirtiendo la energía cinética en energía eléctrica. Sin embargo, solo algunos servodrives permiten esta reinyección de energía al sistema eléctrico en lugar de perderla en forma de calor.

En la actualidad, el control de movimiento tiene un crecimiento exponencial en el trabajo de distintas maquinarias porque el mercado demanda que la forma de producción sea más flexible, rápida y con menos tiempos muertos. Un sistema de control de movimiento consta básicamente de una interfaz hombre – máquina, también conocido como HMI; un controlador; accionamientos o drives; actuadores; mecanismos de transmisión; y retroalimentación o feedback.

En estos sistemas de control, normalmente el PLC envía las órdenes de movimiento al servodrive mediante protocolos industriales como PROFINET, que utiliza telegramas PROFIdrive. El servodrive alimenta y controla el servomotor, ejecutando las instrucciones recibidas del PLC. Para garantizar la precisión, un encoder acoplado al servomotor proporciona información sobre la posición y la velocidad del eje. Esta información es procesada por el servodrive, el cual reporta al PLC, el estado del sistema y otros datos. De este modo, el PLC cumple el rol de coordinador y supervisor de alto nivel, mientras que el servodrive asume la función de control directo y en tiempo real del servomotor.

En los siguientes subapartados se abordarán de manera detallada los elementos que conforman este tipo de control.

II.3.1.1. Controlador Lógico Programable (PLC).

Los controladores lógicos programables son dispositivos electrónicos similares a una computadora, diseñados para automatizar procesos industriales. Son equipos robustos, capaces de soportar las condiciones adversas de ambientes industriales, como exceso de polvo, humedad y temperatura. Además, su instalación suele ser sencilla, lo que los hace muy demandados en diferentes sectores industriales.

Existen diversos fabricantes que ofrecen diferentes modelos y configuraciones orientados a sectores específicos de producción. Los PLCs son altamente eficientes para el manejo de tareas, desde las más simples hasta las más complejas, ejecutadas mediante programas escritos en lenguajes de programación como Ladder, SCL y AWL. Un PLC puede contar con entradas y salidas analógicas y digitales; sin embargo, estas varían según el modelo y fabricante. En la Figura 1 se observa la periferia, señalada con los números 2 y 3. Entonces, un PLC puede recibir señales de entrada provenientes de sensores, botoneras e interruptores y, según su programación, envía señales de salidas a actuadores, pilotos y otros elementos. Al PLC se le puede adicionar módulos de expansión I/O analógicos o digitales, módulos tecnológicos y módulos de comunicación.

Para que el usuario pueda verificar si el proceso controlado por el PLC se encuentra en condiciones óptimas y está en ejecución, los PLCs cuentan con LEDs de estado y, en algunos casos, con una pantalla (display) que permite al usuario ingresar manualmente la dirección IP, así como visualizar de manera detallada los mensajes de diagnóstico y error. Estos indicadores y pantallas se muestran en las Figuras 1 y 2.

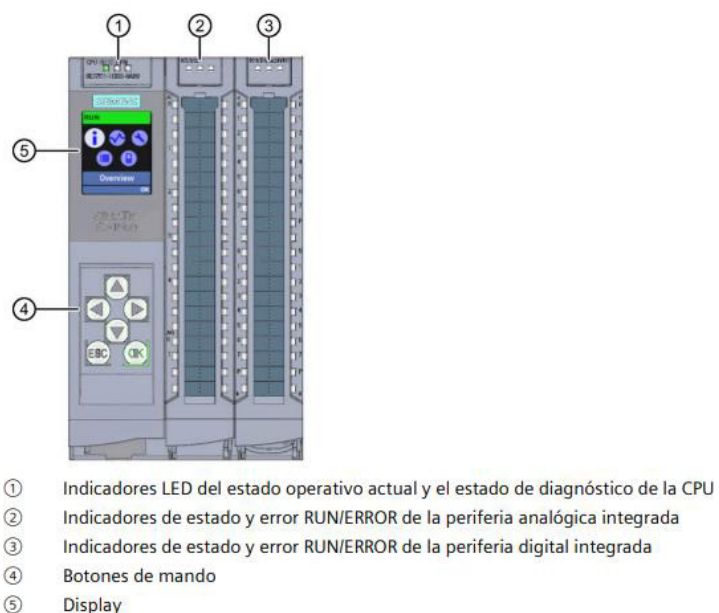


Figura 1. Vista de la CPU 1511C-1 PN con tapas frontales cerradas. Fuente: Siemens AG (2021).

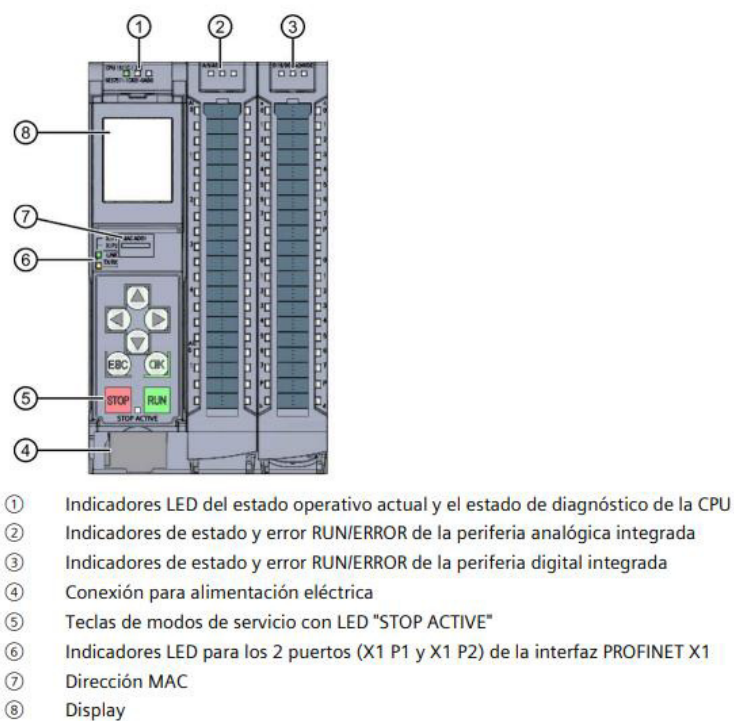


Figura 2. Vista de la CPU 1511C-1 PN sin tapa frontal. Fuente: Siemens AG (2021).

Software de programación

Según Siemens AG (2021), el software Totally Integrated Automation Portal (TIA Portal) es un entorno que integra diferentes herramientas para configurar, programar, visualizar y poner en marcha sistemas de automatización. Incluye STEP 7, que permite programar los controladores SIMATIC mediante distintos lenguajes de programación. Según la complejidad del controlador, existen versiones como STEP 7 Basic y STEP 7 Professional.

En el entorno de STEP 7 se utilizan distintos tipos de bloques para estructurar el funcionamiento del sistema. Los bloques de organización (OB) determinan la secuencia de ejecución del programa, generalmente de forma cíclica o en respuesta a eventos específicos. Las funciones (FC) contienen segmentos de código que se ejecutan sin conservar datos, siendo utilizadas comúnmente una sola vez dentro del programa. Por otro lado, los bloques de función (FB) permiten reutilizar código en varias ocasiones y cuentan con un bloque de datos (DB) asociado que guarda los valores generados durante su ejecución. Finalmente, los bloques de datos se encargan de almacenar variables que pueden ser utilizadas en distintas partes del código.

TIA Portal también incluye el entorno WinCC para el diseño de interfaces HMI en paneles SIMATIC, PC industriales o PC estándar. WinCC cuenta con diferentes ediciones según la complejidad: WinCC Basic, WinCC Comfort, WinCC Advanced, WinCC Runtime Advanced, WinCC Professional, WinCC Runtime Professional y WinCC Unified.

En este entorno se desarrolla el programa de usuario para integrar y configurar el hardware, establecer objetos tecnológicos y controlarlos mediante instrucciones de control de movimiento. Además, TIA Portal facilita la puesta en marcha y la ejecución de diagnósticos del sistema.

Una vez definidos los objetos tecnológicos y sus respectivas instrucciones, el proyecto se carga a la CPU. Gracias a su función Online, el usuario puede ver paso a paso cómo se va ejecutando el programa. (Siemens AG, 2021)

Protocolos de comunicación industrial

Los controladores lógicos programables (PLC), como el S7-1500, pueden comunicarse con otros equipos, como los servodrive SINAMICS V90, mediante protocolos industriales. Por ejemplo, PROFINET IO es un protocolo basado en Ethernet Industrial, según la norma IEEE 802.3, y se caracteriza por brindar una comunicación ágil, flexible y robusta. PROFINET IO permite la transferencia de datos en tiempo real a través de dos modos: PROFINET IO RT, aplicado en procesos estándar, y PROFINET IO IRT, enfocado en aplicaciones de control de movimiento. (Siemens AG, 2022)

Básicamente, este protocolo toma los aspectos más eficaces de PROFIBUS DP y los adapta a un entorno Ethernet, lo que facilita la integración y comunicación entre diferentes equipos. Cuando el control de movimiento se realice mediante PROFINET (ver Figura 3), la comunicación entre el PLC y el servodrive se establece a través de telegramas PROFIdrive, los cuales gestionan el envío de consignas, estados y diagnósticos del eje. Estos telegramas se configuran al establecer los objetos tecnológicos dentro del entorno TIA Portal.

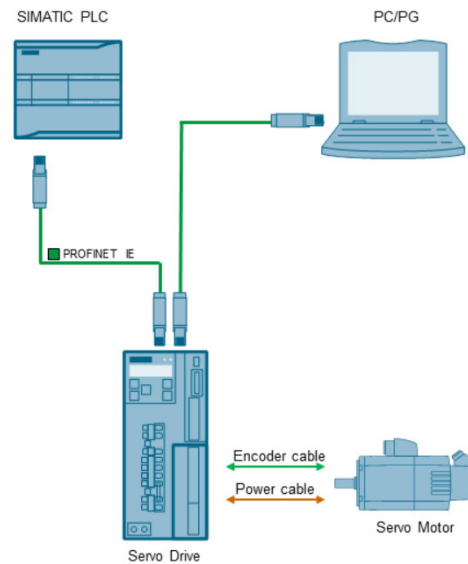


Figura 3. Comunicación PROFINET entre el PLC y servodrive. Fuente: Siemens AG (2020).

Objetos tecnológicos

Los objetos tecnológicos son representaciones virtuales de los ejes físicos. Para la configuración de los objetos tecnológicos, se requiere conocer las propiedades y parámetros del componente físico, los cuales serán posteriormente almacenados en un bloque de datos tecnológicos (véase la Figura 4; Siemens AG, 2021).

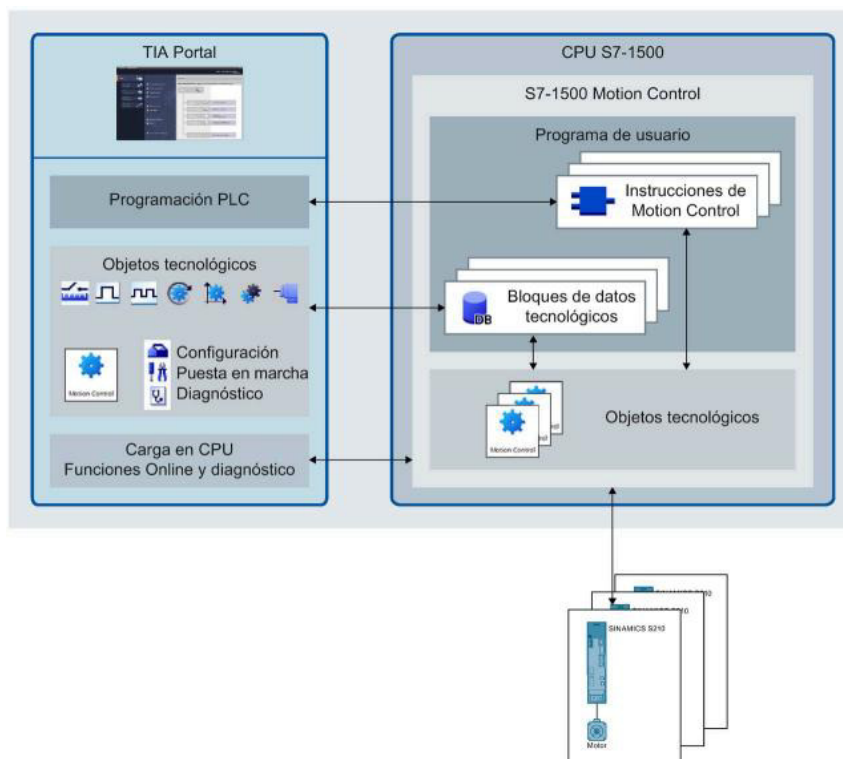


Figura 4. La integración de control de movimiento en la CPU S7-1500. Fuente: Siemens AG (2021).

En el programa de usuario en TIA Portal, mediante instrucciones de control de movimiento que se relacionan a estos objetos tecnológicos, se pueden ejecutar sus funciones asociadas. Dichas instrucciones van dentro de los bloques de organización (OBs). Estas funciones sirven para supervisar y determinar el movimiento del eje real, además de proporcionar información sobre sus estados. Siemens presenta una variedad de objetos tecnológicos, entre los que destacan TO_SpeedAxis y TO_PositioningAxis. El TO_SpeedAxis representa virtualmente el componente mecánico encargado de la velocidad de giro del eje. Las consignas de velocidad se indican y ejecutan mediante instrucciones de control de movimiento en el programa de usuario.

Por su parte, el TO_PositioningAxis se enfoca en tareas de posicionamiento. Mediante instrucciones se establece la posición a la que debe desplazarse el eje, y se permite observar el estado actual de la posición. (Siemens AG, 2021)

Los objetos tecnológicos en TIA Portal ofrecen 3 vistas principales: Configuración, Puesta en marcha y Diagnostico. En la figura 5, se observa que debajo del objeto tecnológico “TO_PositioningAxis” llamado en este caso Faja_posicion [DB1], existe varias vistas.

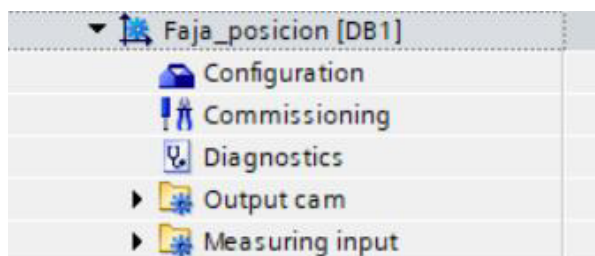


Figura 5. Vistas de los objetos tecnológicos.

En la pestaña “Configuración” se realiza el ajuste del hardware del servodrive y del encoder, incluyendo la selección del telegrama PROFIdrive, la definición de la relación de transmisión, la especificación del paso del husillo y la configuración de los finales de carrera, entre otras opciones. Además, esta sección permite el monitoreo de los parámetros de los ejes (véase la Figura 6).

	Name in function view	Name in DB	Configuration status	Start value project	Minimum value	Maximum value	Co...
Configuration	Axis name	..Mode	✓	Faja_posicion			
Basic parameters	Virtual axis	..Mode	✓	false			*Virt...
Hardware interface	Axis type	..Mode	✓	Linear			
Drive	Motor type	..MotorType	✓	Standard motor			Mot...
Encoder	Use position values w...	..Mode	✓	false			
Data exchange with th...	Unit of measure for p...	..Mode	✓	mm			
Data exchange with en...	Unit of measure for v...	..Mode	✓	mm/s			
Leading value settings	Unit of measure for t...	..Mode	✓	Nm			
Extended parameters	Unit of measure for f...	..Mode	✓	N			
Mechanics	Enable modulo	..Enable	✓	true			Acti...
Dynamic default values	Modulo start value	..StartValue	✓	0.0	mm -1.0e+12	mm 1.0e+12	mm Mo...
Emergency stop	Modulo length	..Length	✓	1000.0	mm 0.0	mm 1.0e+12	mm Mo...
Limits	Activate simulation	..Mode	✓	false			Sim...
Position limits							
Dynamic limits							
Torque limits							
Fixed stop detection							

Figura 6. Vistas de parámetros en los objetos tecnológicos.

II.3.1.2. Servodrive.

El servodrive es un componente electrónico que controla la velocidad, el par y la posición de los servomotores, con el fin de obtener un movimiento preciso. Los fabricantes de equipos de automatización ofrecen distintos modelos de servodrives, disponibles en versiones de 200 V, 400 V y otras tensiones, tanto para redes de alimentación monofásicas como trifásicas. Este dispositivo recibe la alimentación de red, la rectifica y la convierte en una salida trifásica para alimentar al servomotor, utilizando los pines inferiores U, V y W.

Entre las partes generales de un servodrive se encuentra el panel de operador básico (BOP), que permite realizar ajustes de los parámetros, designar valores de referencia, establecer unidades y visualizar los estados y alarmas del servodrive. Por medio de esta interfaz, el usuario puede iniciar una puesta en marcha sencilla.

También suelen contar con una interfaz mini USB que permite la conexión de una PC al servodrive. Mediante un software del mismo fabricante, es posible visualizar y configurar los parámetros de manera más ágil y ordenada que desde el BOP, ya que estos se presentan en esquemas que facilitan la realización de múltiples configuraciones desde una sola pestaña.

Concentrándonos en el tipo de servodrive de bucle cerrado, se observa en la Figura 7 que presenta un conector de alimentación de 24 V/STO, cuya función predeterminada es la desactivación de par seguro (STO). Esta función impide el reinicio intempestivo del drive. Este conector está conformado por los pines: STO 1, STO +, STO 2, +24 y M. (Siemens AG, 2015)

Las tarjetas micro SD almacenan archivos de firmware y parámetros; estos últimos pueden ser transferidos a la memoria ROM. La interfaz de control y estado cuentan con entradas y salidas de tren de pulsos, digitales y analógicas.

La interfaz del encoder se conecta mediante un cable apantallado a encoders de tipo incremental o absoluto, contando con 14 pines que se enlazan a los 8 pines del encoder. Asimismo, la interfaz PROFINET permite realizar integraciones industriales con otros equipos como servodrives, PLCs, paneles HMIs y PCs, destacando por su alta velocidad de comunicación y la posibilidad de supervisar los parámetros desde un punto remoto al equipo.

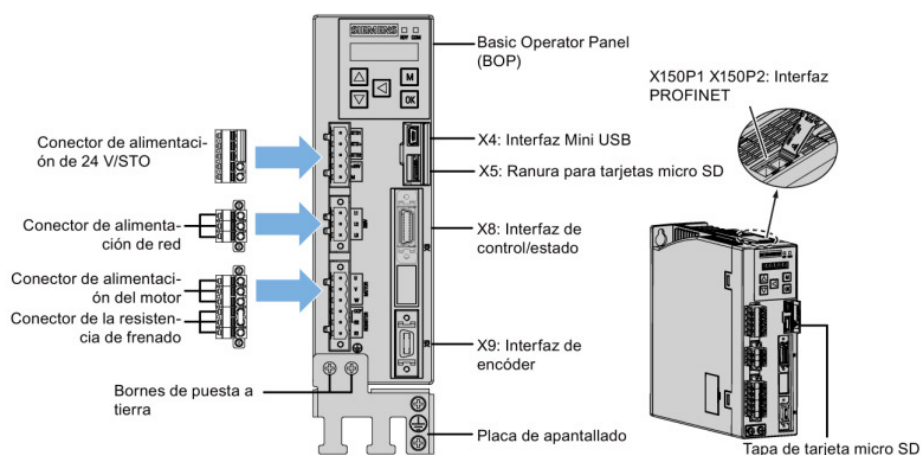


Figura 7. Partes del servodrive. Fuente: Siemens AG (2016).

Funcionamiento

Los servodrives de bucle cerrado representan una propuesta avanzada que realiza la lectura constante de las señales del encoder para conocer con exactitud la posición del eje y ajustar el control. Este bucle cerrado consta de una señal de comando que es enviada desde un PLC y recibida por el servodrive (véase la Figura 8), en donde se le indican las consignas con las que trabaja el servomotor en un determinado momento. Luego, el servodrive toma esta señal y la compara con la señal recibida del encoder para obtener el error, con el cual modifica el voltaje y la corriente suministrados al servomotor, con el fin de corregir la desviación y alcanzar las consignas requeridas según las líneas de producción.

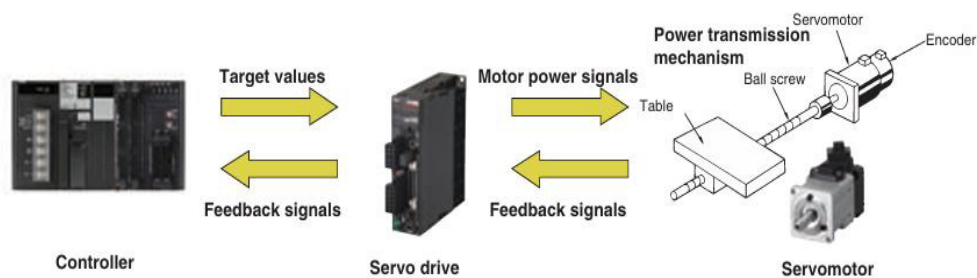
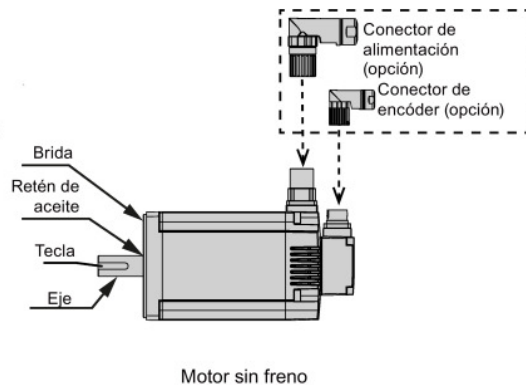


Figura 8. Funcionamiento del sistema. Fuente: Omron Corporation (s.f.).

Conexión con el servomotor

Los servodrive y servomotores pueden conectarse por medio de dos cables: uno designado para la alimentación y otro para la señal del encoder, ya sea incremental o absoluto, según sea el caso (Figuras 9 y 10). El cable de potencia trifásica proporciona la energía que permite el funcionamiento del servomotor, mientras que el cable de retroalimentación brinda una señal constante al servodrive sobre la posición y velocidad del motor. El servodrive procesa esta información para lograr un control dinámico y continuo.



Figuro 9. Conectores de servomotor. Fuente: Siemens AG (2016).

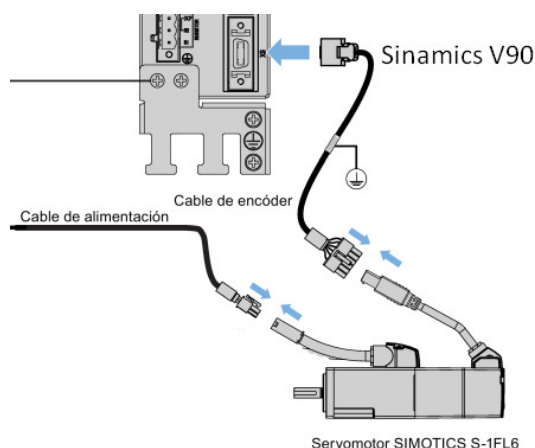


Figura 10. Conexión entre el servodrive y servomotor. Fuente: Adaptado de Siemens AG (2016).

II.3.1.3. Servomotor.

Un servomotor (véase la Figura 11) es un actuador rotativo o lineal que convierte la energía eléctrica en energía mecánica. Generalmente, está acompañado de un encoder que mide el movimiento del eje, generando señales que son interpretadas por el servodrive. Este último regula y envía las señales de control para gobernar la posición, la velocidad, el par u otras consignas. Existen fabricantes que ofrecen desde modelos pequeños de servomotores, los cuales conservan la misma fuerza, par y potencia que otros motores de mayor tamaño.



Figura 11. Servomotor del fabricante Siemens. Fuente: Siemens AG (s.f.).

Encoder

Un encoder es un dispositivo electromecánico que convierte el movimiento físico, ya sea rotacional o lineal, en señales eléctricas codificadas. Estas señales son interpretadas por dispositivos como el PLC o el servodrive, para determinar parámetros como la posición, la dirección y la velocidad de un eje. Existen dos tipos principales de encoders: absolutos e incrementales. Los encoders absolutos entregan una señal única para cada posición del eje y, aun cuando el sistema se apaga, mantienen la última posición registrada, lo que los hace ideales para aplicaciones críticas donde se requiere alta confiabilidad.

Los encoders incrementales son dispositivos que generan pulsos eléctricos proporcionales al movimiento detectado, destacando por su simplicidad y capacidad para detectar pequeños desplazamientos. Sin embargo, no conservan información de posición luego de una pérdida de energía. Su principio de funcionamiento se basa en un sistema óptico compuesto por un emisor de luz, un fotorreceptor y un disco con ranuras montado en el eje. A medida que el eje gira, las ranuras interrumpen el haz de luz, produciendo una secuencia de pulsos eléctricos; cada paso de luz genera un nivel lógico 1 y cada interrupción genera un 0, lo que resulta en una señal digital que representa el movimiento. Además, algunos encoders incrementales incluyen la señal z, la cual genera un único pulso por cada vuelta completa del eje y sirve como punto de referencia para sincronización o para establecer una posición inicial específica.

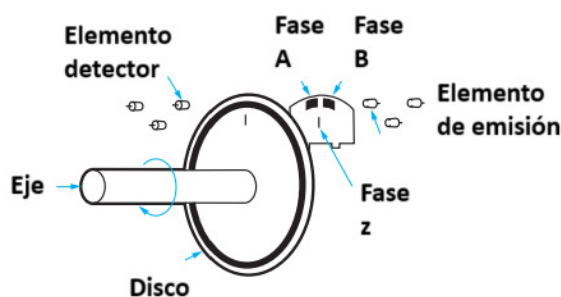


Figura 12. Principio de encoder. Fuente: Adaptado de Omron Corporation (s.f.).

Para determinar la dirección del movimiento, se utilizan sensores ópticos desfasados entre sí en un cuarto del espacio entre ranuras, lo que da como resultado señales A y B desfasadas 90 grados entre sí (Figura 13). (Omron Corporation, s.f.)

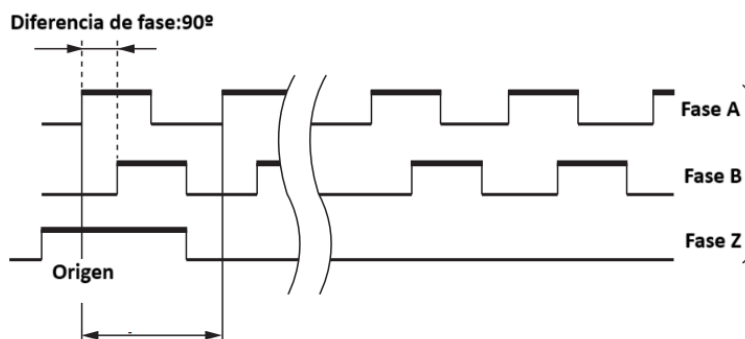


Figura 13. Señales de encoder. Fuente: Adaptado de Omron Corporation (s.f.).

En modelos más avanzados, los encoders pueden contar con una salida de alarma que indica fallos internos, como el desgaste de rodamientos o problemas en los sensores ópticos. Además, la resolución de un encoder incremental se mide en pulsos por revolución (PPR), y mientras mayor sea la cantidad de pulsos, mayor será la precisión del dispositivo, ya que cada pulso representa un avance muy pequeño del eje.

Funcionamiento

Esta clase de servomotores AC se divide en dos grupos: síncronos y asíncronos. Los servomotores síncronos tienen una velocidad de rotación en el rotor que coincide con la del campo magnético del estator, es decir, no hay deslizamiento. El rotor se sincroniza con la frecuencia de la corriente trifásica de alimentación, resultando en un control de velocidad más exacto.

El rotor puede estar compuesto por un imán permanente (ver Figura 14) o bobinas de corriente continua, lo que le permite generar un campo magnético constante o variable. El torque del servomotor se genera por la relación entre los campos magnéticos del rotor y estator, y está determinado por la fuerza de dichos campos y su sincronización.

Este tipo de servomotores suelen presentar alta eficiencia de energética, aunque tiende a sobrecalentarse si no cuenta con una adecuada ventilación y requiere un sistema de control más elaborado.

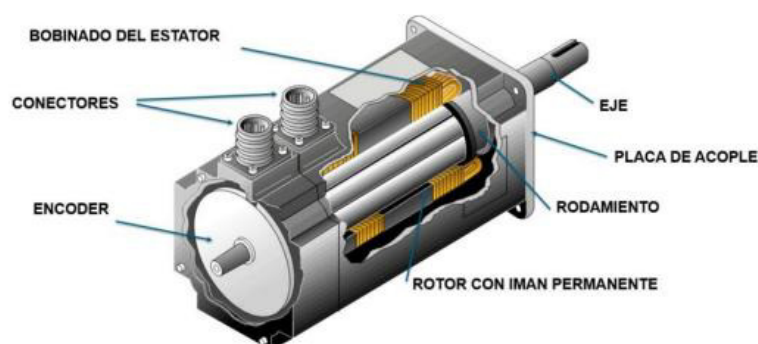


Figura 14. Partes del servomotor. Fuente: Calvo (2022).

Los servomotores AC trabajan con corriente alterna y se utilizan en aplicaciones donde se requiere máxima precisión en la posición, velocidad y par. Operan en sistemas de bucle cerrado, donde el servodrive desempeña el rol de monitorear y ajustar el voltaje, la frecuencia y la corriente que se entrega al servomotor, en base a la desviación generada entre la señal de control y la retroalimentación del encoder. Estos servomotores destacan por su capacidad de manejar altas cargas de potencia de manera eficiente, así como por brindar una aceleración y frenado de alta precisión. Gracias a su robustez, son requeridos con mayor frecuencia en escenarios de alto esfuerzo.

Velocidad

La velocidad del servomotor se refiere a la rapidez con la que se mueve el eje. Según las unidades requeridas, se distinguen dos tipos: la velocidad angular, que mide las revoluciones por minuto (RPM) o radianes por segundo (rad/s), y se emplea cuando se trabaja directamente con el servomotor; y la velocidad lineal, que se aplica cuando el movimiento del servomotor genera un desplazamiento rectilíneo mediante un husillo (véase la Figura 15), cremallera o correa, y se mide en milímetros por segundos (mm/s).

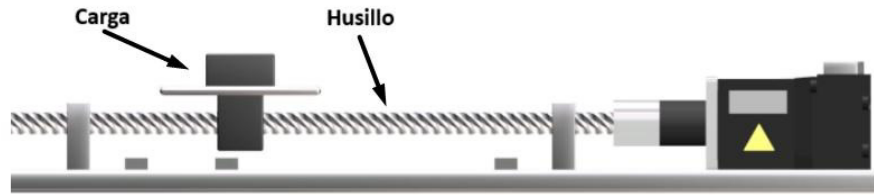


Figura 15. Eje con husillo. Fuente: Adaptado de Mitsubishi Electric (s.f.).

El encoder incremental del servomotor genera pulsos eléctricos proporcionales al giro del eje. El servodrive cuenta esos pulsos durante un intervalo de tiempo conocido y calcula la velocidad angular. (Buczaj & Cisek, 2024)

$$Velocidad\ angular\left(\frac{rev}{s}\right) = \frac{N^{\circ}\ de\ pulsos\ detectados}{Pulsos\ por\ vuelta\ x\ tiempo(s)} \quad (1)$$

Según Bolton (2013), para convertir la velocidad angular en velocidad lineal, el servodrive multiplica el desplazamiento del husillo (mm/rev) por la velocidad angular (rev/s). En caso se utilice la velocidad en RPM, la fórmula sería:

$$Velocidad\ lineal\ \left(\frac{mm}{s}\right) = paso\ de\ husillo\ x\ \frac{RPM}{60} \quad (2)$$

Si la relación de reducción del reductor de carga es diferente de 1, se debe incluir en el denominador.

Posición

Un encoder incremental genera pulsos digitales de los canales A y B, además de un pulso índice del canal Z por cada rotación. Posteriormente, el servodrive cuenta estos pulsos para determinar cuántas revoluciones ha girado el eje desde un punto de referencia. Para obtener la posición absoluta, es necesario contar con una señal índice o realizar un homing antes de iniciar el proceso.

Cuando el servomotor mueve una carga lineal a través de un husillo, como se muestra en la Figura 15, la determinación de la posición es calculada por el servodrive. Estos dispositivos cuentan con la configuración del paso de husillo (mm /rev) y la resolución del encoder (pulsos/rev), por lo que, al recibir y contar los pulsos, calculan la posición.

Según Omron Corporation (s.f.), la fórmula para calcular la precisión de posicionamiento (mm/pulso) es:

$$P. \text{ de posicionamiento} = \frac{\text{Relacion de engranajes} \times \text{Paso de husillo}}{\text{Resolucion} \times \text{Factor de subdivision}} \quad (3)$$

El factor de subdivisión permite fraccionar artificialmente las señales del encoder para aumentar la precisión sin necesidad de cambiar el encoder. Asimismo, es necesario considerar el número de dientes de los engranajes según sea el caso y Omron la denomina como la relación de reducción (G) (véase la Figura 16).

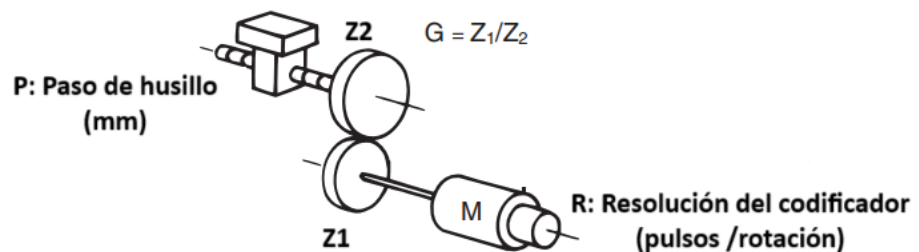


Figura 16. Sistema de engranajes. Fuente: Adaptado de Omron Corporation (s.f.).

La fórmula de precisión de posicionamiento indica cuántos milímetros avanza el sistema por cada pulso de encoder. Por lo tanto, si se multiplica por N pulsos, se obtiene el avance del eje:

$$\text{Posición(mm)} = \text{Numero de pulsos} \times \text{Precisión de posicionamiento} \quad (4)$$

Torque

El torque (T) es la medida de la capacidad de una fuerza para hacer girar un objeto alrededor de un eje. Se calcula como el producto vectorial entre la distancia (r) del punto de aplicación de la fuerza al eje de rotación y la fuerza aplicada (F). La fórmula es la siguiente, donde θ es el ángulo entre los vectores:

$$|T| = |r \times F| = rF \sin \theta \quad (5)$$

En el caso del lado izquierdo de la Figura 17, una fuerza actúa a una distancia r_1 de la bisagra, produciendo un torque. En el segundo caso, el torque será inferior al caso anterior, ya que la misma fuerza se aplica con un ángulo inferior a 90 grados.

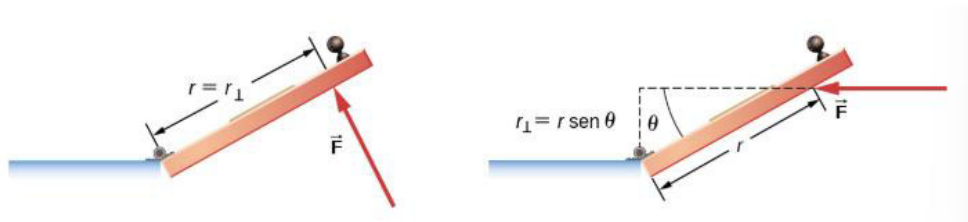


Figura 17. Casos de aplicación de torque. Fuente: Ling, Moebs & Sanny (2021).

En un servomotor síncrono, el torque generado está relacionado con la corriente que circula en sus bobinas. Cada motor tiene una constante de torque (Kt), con unidades N.m/A, que indica la relación entre la corriente aplicada (I) y el torque producido (T):

$$T = I \times Kt \quad (6)$$

Esto significa que cuanto mayor sea esta constante, menor será la corriente necesaria para generar un torque específico. El servodrive utiliza sensores de corriente para medir la corriente en cada fase del motor y calcular el torque. Según Kollmorgen (2022), en aplicaciones donde se requiere controlar el torque, se utiliza exclusivamente un bucle de corriente, ya que el torque está directamente relacionado con esta variable. El servodrive compara la corriente deseada con la medida y ajusta la señal de la salida

para mantener el torque solicitado. En conclusión, el servodrive no calcula directamente el torque, sino que lo estima a partir de la corriente medida y la constante del torque del motor.

II.3.2. Nivel de supervisión en la automatización industrial

La automatización industrial consiste en la integración de diversos componentes de hardware y software para el control automático y el monitoreo de sensores y actuadores. Dicha integración permite optimizar la producción, reducir los tiempos de inoperatividad, garantizar una alta disponibilidad de datos, facilitar los diagnósticos y promover el ahorro en mantenimiento. La estructura del sistema de automatización industrial se divide en tres niveles jerárquicos: supervisor, de control y campo.

En el nivel de supervisión se encuentran los sistemas SCADA en donde se visualizan curvas, visor de alarmas, condiciones del proceso y de los equipos. El monitoreo proporciona una visión integral del proceso productivo, lo que facilita la toma de decisiones efectivas orientadas a mejorar la eficiencia operativa.

II.3.2.1. Sistemas supervisión, control y adquisición de datos (SCADA).

El sistema SCADA integra componentes de software, hardware y comunicaciones para supervisar, adquirir y transferir datos provenientes de sensores y actuadores distribuidos en las líneas de producción, así como para controlar procesos industriales de manera eficaz mediante una interfaz gráfica remota, intuitiva y estructurada. Este sistema permite almacenar y procesar información proveniente de diversos elementos de la planta, incorpora mecanismos de ciberseguridad para proteger la infraestructura digital, y despliega mensajes personalizados relacionados con fallas, alarmas, diagnósticos, eventos, secuencias de procesos y estados operativos. Gracias a estas funcionalidades, refuerza la seguridad operativa, optimiza las tareas de mantenimientos predictivos y reduce los tiempos de inactividad.

Un sistema SCADA está compuesto por una o varias aplicaciones software destinadas a la supervisión y control, cuyo objetivo principal es brindar un acceso a la

información de planta mediante la comunicación digital con los dispositivos de campo, permitiendo así la interacción entre el operario (interfaz gráfica de alto nivel) y el proceso industrial (adquisición de datos).

Este sistema ofrece al operador la posibilidad de supervisar el funcionamiento de la planta y visualizar en tiempo real las variables que influyen en el proceso, otorgándole la capacidad de modificar dichos parámetros de manera inmediata. Gracias a su interfaz intuitiva, esta capacidad de ajuste representa una ventaja significativa frente a otras soluciones.

En la actualidad, existen diferentes tipos de SCADA (véase la Figura 18) según las mejoras tecnológicas desarrolladas por los fabricantes, pero todos comparten una estructura básica de conexión con otros equipos del sistema de automatización.

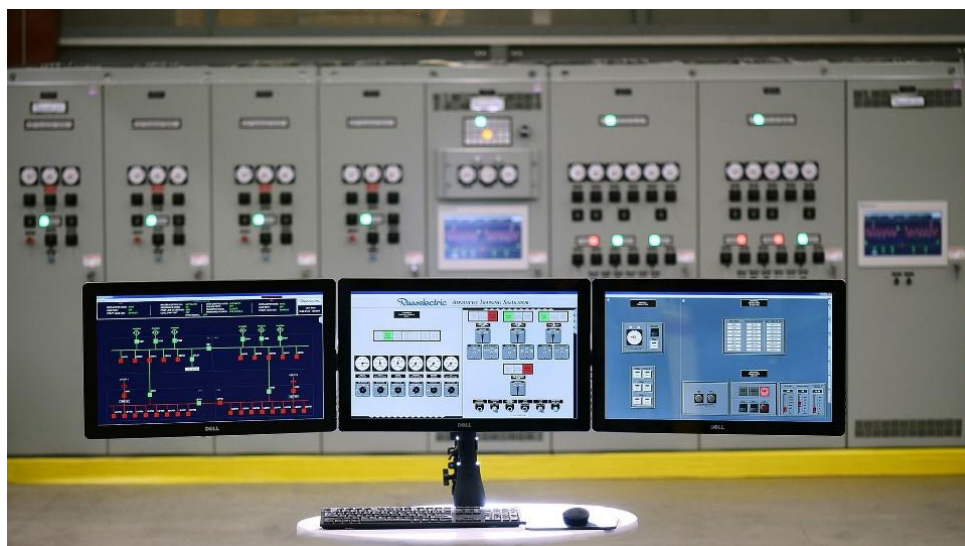


Figura 18. Sistema SCADA. Fuente: Siemens AG (s.f.).

Componentes

Un sistema SCADA (véase la Figura 19) mayormente consta de HMIs, unidades terminales maestras (MTU), unidades terminales remotas (RTU), redes, sensores, actuadores, y el software de supervisión. Los RTU actúan como intermediarios entre el

MTU y los dispositivos de campo, encargándose de recolectar datos y ejecutar comandos de control. Tradicionalmente diferenciadas de los PLCs, en la actualidad muchos PLCs modernos pueden desempeñar funciones de RTU, ofreciendo así una solución más económica, compacta y flexible para diversas aplicaciones. El RTU puede ir dentro de armarios de control junto con elementos auxiliares y de protección eléctrico.

La parametrización se realiza desde un PLC y RTU, los cuales recopilan la información de las señales de campo y las transmiten al MTU. Este último puede ejecutar funciones de análisis y, de ser necesario, se envían de nuevo al proceso y finalmente se presentan en las HMIs para que el operador supervise y controle la planta. El MTU puede ser una PC industrial o servidor, en donde se puede adicionar softwares para incrementar las bondades del sistema, como generación de reportes, almacenamiento histórico, conectividad con plataformas en la nube. Con un único MTU que ejecute supervisión y control, se podría hablar de una arquitectura SCADA simple que, a medida que se demande mayor escalabilidad, este modelo podría convertirse en una arquitectura cliente - servidor con redundancia.

Todo el hardware ya mencionado es necesario complementarlo con software especializado. Por ejemplo, Siemens AG presenta los sistemas SIMATIC WinCC y SIMATIC PCS7, que brindan una supervisión completa de procesos industrial, gestión de usuarios, alarmas, configuración de estaciones y servidores, almacenamiento histórico e integración con sistemas MES y ERP. Según Siemens AG (2021), WinCC ofrece diferentes versiones, siendo WinCC Professional una de las más completas porque permite configurar tanto paneles como PCs con WinCC Runtime Advanced o sistema SCADA WinCC Runtime Professional. Este último está diseñado para sistemas de tipo monopuesto o multipuesto, admitiendo tantos clientes estándar como web. La licencia de WinCC Runtime Professional se basa en el número de variables de proceso

conectadas, conocidas como PowerTags, con opciones que van desde 128 hasta 256k tags.

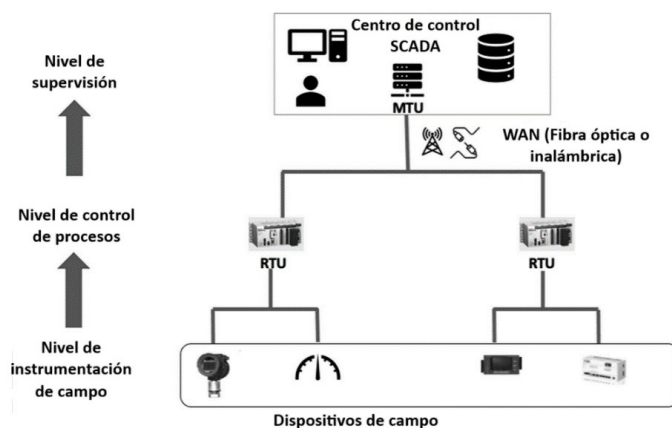


Figura 19. Componentes. Fuente: Adaptado de Ghosh, Joshi, Sampalli & Zaman (2024).

Tipos

El sistema SCADA monolítico o autónomo es una de las primeras arquitecturas desarrolladas para cubrir las necesidades de monitoreo y control en plantas industriales. Se caracteriza por ser independiente, sin conexión con otros sistemas, y resulta ideal para plantas pequeñas con procesos simples, donde se busca reducir costos en la implementación, ya que todas las funciones de supervisión y control se realizan desde una sola estación central.

Para grandes líneas de producción se aconseja usar la robusta arquitectura de un SCADA distribuido, la cual ofrece la posibilidad de conectarse con otras estaciones de trabajo para monitorear desde una sola estación varios procesos. Aquí aparece el modelo servidor – cliente, en donde los clientes obtienen información que gestiona el servidor. Esta arquitectura se encuentra protegida ante fallos, porque presenta mecanismos de redundancia.

En la arquitectura de SCADA de cuarta generación, basada en internet de las cosas, se integra la computación en la nube, redes inalámbricas, entre otros elementos tecnológicos. Al estar alojada en la nube, este tipo de SCADA ofrece alta escalabilidad,

facilidad de mantenimiento y una reducción de costos relacionados a infraestructuras, ya que no necesita servidores locales ni almacenamiento físico.

II.3.3. Indicadores Clave de Rendimiento (KPIs)

Los indicadores clave de rendimiento son métricas específicas y cuantificables que miden el desempeño de equipos, áreas o procesos dentro de una organización. Los indicadores permiten monitorear y analizar datos que describen el comportamiento operativo, lo cual favorece una toma de decisiones eficaz y ágil, que este orientada a mejorar el desempeño actual. Mayormente, los KPIs se calculan a partir de datos del proceso productivo, ya sea provenientes de las máquinas o de las áreas responsables como mantenimiento, comercial, operaciones, entre otras. Por medio de fórmulas predeterminadas aplicadas durante un período específico, se pueden obtener valores que luego son comparados con valores objetivos. En este punto, al existir una diferencia amplia o al buscar mejorar la eficiencia o productividad, se generan acciones correctivas y se establecen nuevas estrategias. Según el tipo de proyecto o empresa, se tienen los KPI de ventas, financieros, marketing digital, redes sociales, logística y producción.

II.3.3.1. ISO 9001:2015 – Sistemas de gestión de la calidad.

Si bien esta norma no proporciona una lista de KPI's a utilizar, sí exige que las compañías definan objetivos mediables relacionados a la calidad, analicen el desempeño de los procesos, evalúen si las acciones correctivas realmente mejoran dichos procesos y tomen decisiones en base a datos cuantitativos. Las organizaciones en función de sus procesos, metas y riesgos, determinan qué KPIs son más pertinentes. En la mayoría de sistemas, es común el uso de indicadores como el tiempo de inactividad, tasa de rechazos, la satisfacción del usuario, entre otros.

Además, la norma recalca que, para una toma de decisiones eficaz, se debe tener datos documentados y organizados. Con ello, se refuerza la idea que es de vital importancia contar con un almacenamiento ordenado y una monitorización continua de los KPIs seleccionados.

II.3.3.2. KPIs en la industria. Se hará hincapié en los siguientes KPIs, acorde a la necesidad de la presente tesis.

➤ tasa de disponibilidad

La disponibilidad mide el tiempo en que la máquina ha estado produciendo en comparación con el tiempo programado para trabajar. Suele considerar los mantenimientos o periodo de fallas.

$$\text{Disponibilidad} = \frac{\text{Tiempo de ejecución}}{\text{Tiempo de producción planificado}} \times 100 \% \quad (7)$$

➤ tasa de calidad

La calidad considera los productos que están en óptimas condiciones y listos para el público consumidor, respecto al total de producción, en donde se contabilizan las unidades que no lograron pasar los estándares de calidad.

$$\text{Calidad} = \frac{\text{Unidades en buena calidad}}{\text{Unidades totales producidas}} \times 100\% \quad (8)$$

➤ tasa de utilización de la capacidad

Este KPI mide, en porcentaje, el grado en que un proceso utiliza su capacidad de trabajo. Para ello, se divide el total de unidades producidas entre la capacidad teórica máxima, entendida como el número de unidades que podrían haberse producido en un período determinado, multiplicado por el número total de períodos transcurridos.

$$\text{Tasa de utilización de capacidad} = \frac{\text{Producción total}}{\text{Total de producción teorica}} \times 100 \% \quad (9)$$

➤ productividad

Mide la capacidad de una línea de producción para generar unidades de calidad en un determinado periodo, que puede ser minutos, horas, según la duración de sus ciclos de trabajo.

$$Productividad = \frac{Producción\ total - Merma}{Tiempo\ trabajado} \quad (10)$$

II.3.3.3. Cálculo y visualización de KPIs.

Para el cálculo y visualización de KPIs en industrias manufactureras, se utilizan diferentes tipos de software, como sistemas ERP, MES, herramientas de Business Inteligencia (BI), plataformas de analítica avanzada y también hojas de cálculo como Microsoft Excel o Google Sheets. Las hojas de cálculo son una propuesta básica que permite visualizar y organizar datos; sin embargo, implican una recolección manual de datos o la importación de archivos, aumentar la probabilidad de errores y tiempo de procesamiento. La automatización de reportes necesita conocimientos de fórmulas y macros avanzados.

Además, presentan una escalabilidad limitada, ya que tienen problemas para manejar grandes volúmenes de datos o múltiples KPIs de forma simultánea. Aunque permiten la creación de gráficos, sus opciones visuales y de análisis son limitadas en comparación con plataformas BI. También puede existir limitaciones en cuanto a la colaboración en tiempo real, gestión de usuarios y monitoreo de cambios.

Grafana Cloud

Como una propuesta más eficiente y moderna se tiene a Grafana Cloud (véase la Figura 20), que es una plataforma especializada en el análisis de datos en series de tiempo. La plataforma recopila y ordena los flujos de datos, y los muestra por medio de graficas personalizables. De esta manera, el usuario puede analizar las tendencias durante periodos definidos con el fin de entender el comportamiento de las aplicaciones y detectar posibles errores. No necesita algún código adicional para conectarse a bases de datos como InfluxDB, MySQL, PostgreSQL, Prometheus, entre otros.

En su interfaz, el usuario puede arrastrar y soltar componentes de diversos tipos de paneles, como gráficos de líneas, áreas apiladas, mapas geográficos y de calor,

histogramas, gráficos de barras, tablas, indicadores, paneles de texto o diagramas de flujo, y luego ajustar colores, ejes, leyendas, rangos y transformaciones según sus necesidades. Por último, integra un motor de alertas configurable que, cuando se superan los umbrales establecidos, procede a notificar vía correo u otro medio.



Figura 20. Panel de Grafana. Fuente: Grafana Labs (2025).

II.3.4. Almacenamiento y gestión de datos.

Es el proceso de almacenar, resguardar, proteger y administrar los datos de una organización.

II.3.4.1. SQL server.

Es un sistema de administración de bases de datos relacionales desarrollado por Microsoft. Se entendiendo como relacional al tipo de bases de datos que organiza la información en tablas con filas y columnas, similares a una hoja de Excel.

Según Microsoft (2025), SQL server puede instalarse en máquinas virtuales, sistemas operativos Windows o Linux, o dentro de contenedores como los que usa Docker. Además de guardar datos, SQL server tiene otras funciones muy útiles. Por ejemplo, puede ayudar a analizar datos, crear reportes, limpiar datos repetidos o

incorrectos, compartir los datos con otras bases e incluso usar inteligencia artificial para hacer predicciones. Todo esto lo convierte en una herramienta muy completa para trabajar con datos en empresas.

Para acceder a los componentes de SQL server, administrarlos y desarrollarlos, es necesario usar SQL Server Managment Studio (SSMS), que combina interfaces gráficas y editores de código. En la Figura 21, podemos ver que SSMS tiene un explorador de objetos que muestra y gestiona todos los elementos que existen en una base de datos, como las tablas y otros elementos. También incluye un explorador de plantillas para almacenar fragmentos de código reutilizable con el fin agilizar la creación de scripts. En caso se esté trabajando con varios archivos, se pueden agrupar con el explorador de soluciones. Además, incluye herramientas de diseño visual para compilar consultas, bases de datos y tablas, así como editores de script con resaltado de sintaxis y herramientas de depuración. (Microsoft, 2025)

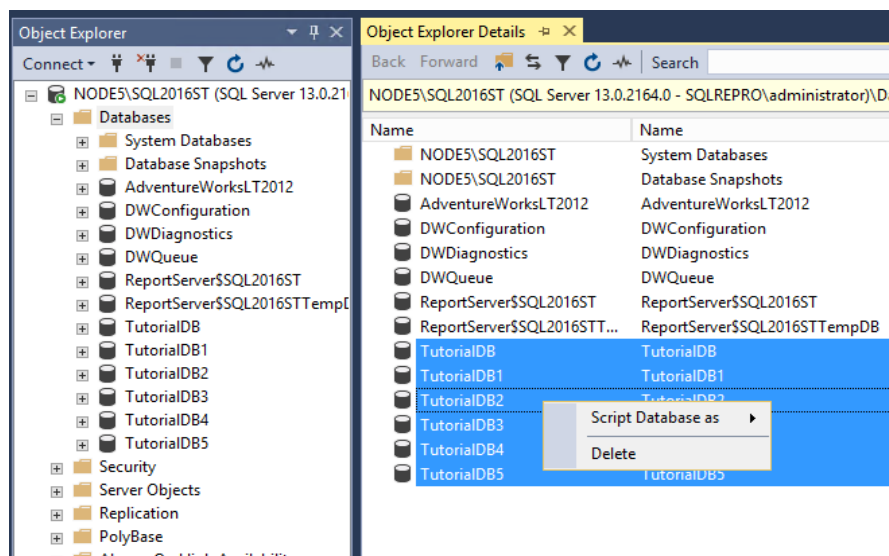


Figura 21. Explorador de objetos en SSMS. Fuente: Microsoft (2025).

II.3.4.2. InfluxDB Cloud.

Es una plataforma que almacena y consulta datos de series temporales (ver Figura 22), es decir, datos que se recopilan en momentos específicos y siguen un orden

cronológico. Además, puede procesar un gran flujo de datos en tiempo real, siendo capaz de escribir hasta 45 veces más rápido y consultar hasta 100 veces más rápido que otras soluciones, debido a su motor optimizado para manejar alta cardinalidad.

La herramienta permite consultar datos utilizando SQL a través de un editor visual intuitivo que facilita la creación de scripts, sin necesidad de configurar infraestructura. También se resalta su integración con Telegraf, que permite recopilar datos desde diferentes fuentes como sensores y aplicaciones. Su arquitectura basada en la nube ofrece escalabilidad automática y una reducción significativa del 90% en costos de almacenamiento. (InfluxData, 2024)

También es compatible con múltiples plataformas industriales. Por ejemplo, desde TIA Portal se pueden enviar datos a InfluxDB Cloud mediante un intermediario como Node-RED, una herramienta de programación visual que facilita la conexión entre dispositivos y servicios.

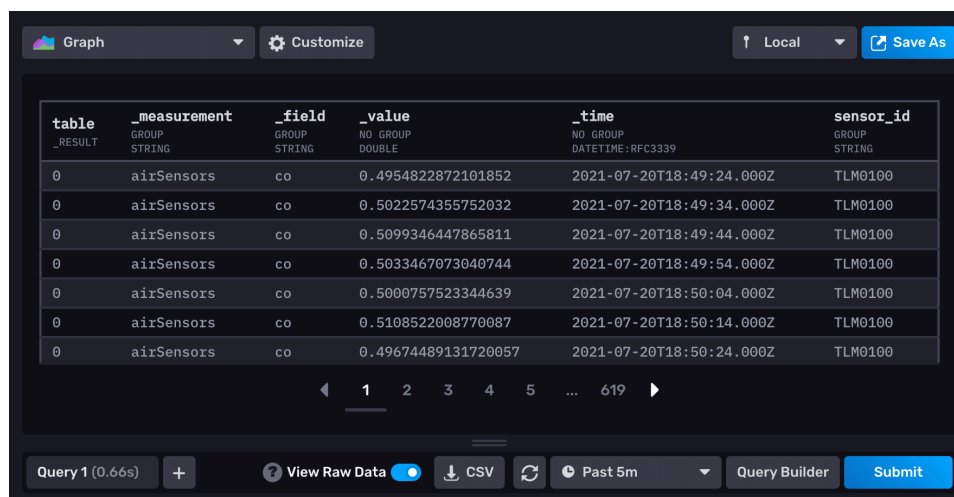


table	_measurement	_field	_value	_time	sensor_id
GROUP	GROUP	GROUP	NO GROUP	NO GROUP	GROUP
_RESULT	STRING	STRING	DOUBLE	DATETIME:RFC3339	STRING
0	airSensors	co	0.4954822872101852	2021-07-20T18:49:24.000Z	TLM0100
0	airSensors	co	0.5022574355752032	2021-07-20T18:49:34.000Z	TLM0100
0	airSensors	co	0.5099346447865811	2021-07-20T18:49:44.000Z	TLM0100
0	airSensors	co	0.5033467073040744	2021-07-20T18:49:54.000Z	TLM0100
0	airSensors	co	0.5000757523344639	2021-07-20T18:50:04.000Z	TLM0100
0	airSensors	co	0.5108522008770087	2021-07-20T18:50:14.000Z	TLM0100
0	airSensors	co	0.49674489131720057	2021-07-20T18:50:24.000Z	TLM0100

Figura 22. Datos en formato de tabla en InfluxDB Cloud. Fuente: InfluxData (2025).

II.3.5. Proceso de elaboración de bebidas

El proceso de elaboración de bebidas (Figura 23) se divide en varias etapas, iniciando en el establecimiento de suministro de agua potable, que luego es tratada para

ser apta para el consumo. De manera continua, se añaden edulcorantes, cafeína, zumos, vitaminas, azúcares, minerales, aromas, de acuerdo con la fórmula de cada producto. Si la bebida es gasificada, se le añade anhídrido carbónico, que cumple la función de preservante; en el caso de bebidas sin gas, se debe pasar por un proceso térmico de pasteurización con el fin de mantener las propiedades del producto.

Finalizado el proceso de mezclado, se procede a dosificar el producto y tapar las botellas, para luego verificar que el nivel sea adecuado. Con el producto sellado, se inicia el etiquetado, donde se visualizan los ingredientes y propiedades. Finalmente, las botellas son agrupadas de seis en seis en pallets para su distribución. (Asociación de Bebidas Refrescantes [ANFABRA], 2024)

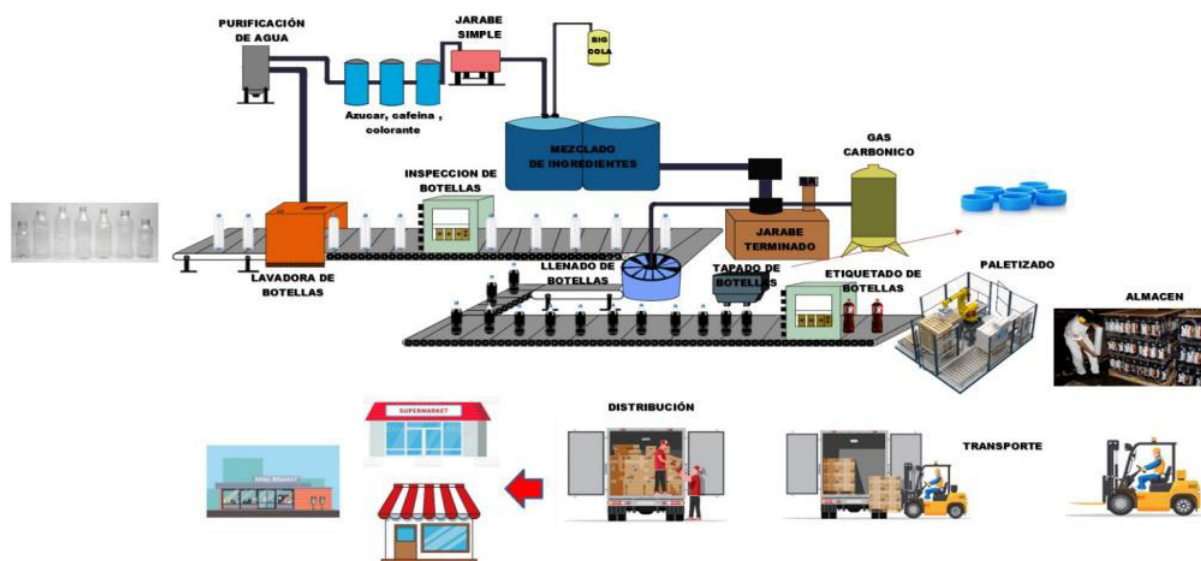


Figura 23. Proceso de elaboración de bebidas. Fuente: Chaname, Cruz, Hernández, Rojas y Tisnado (2022).

II.3.5.1. Llenado de botellas.

El proceso de llenado consiste en pasar líquido o producto desde un tanque hacia una botella o envase. Según OCME (s.f.), este procedimiento puede ejecutarse mediante distintos sistemas, que se distinguen por el método con el que miden y controlan la cantidad de producto a dosificar. Entre los más comunes se encuentran las llenadoras a

nivel, volumétricas y ponderales. El tipo de llenadora depende del tipo de producto a envasar, el material y diseño del envase, así como el grado de precisión requerido.

Las llenadoras a nivel son las más sencillas en cuanto a funcionamiento. La cantidad de producto es determinado por la altura del líquido en la botella, regulada por la longitud de la cánula insertada (véase la Figura 24). Para modificar ese nivel es necesario cambiar la cánula.



Figura 24. Sistema de llenado a nivel. Fuente: Adaptado de OCME (s.f.).

Las llenadoras volumétricas (véase la Figura 25) cuentan con sensores de caudal, másicos o magnéticos en cada boquilla para calcular el volumen dosificado en cada envase. Una variante de este sistema es el *piston filler*, que utiliza pistones neumáticos para desplazar un volumen exacto de producto. Estas llenadoras emplean bombas dosificadoras del tipo peristáltico o volumétrico, accionadas por servomotores, lo que garantiza una alta precisión. Cada bomba puede ajustarse de forma independiente mediante una interfaz HMI, y los servomotores facilitan la rápida adaptación del sistema a distintos productos. Asimismo, las versiones rotativas de estas máquinas están orientadas a procesos de alta velocidad y producción sincronizada a gran escala.

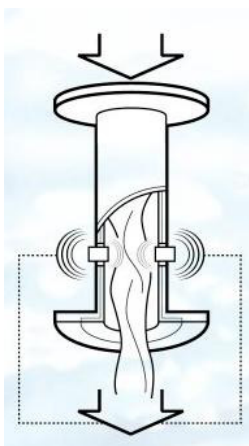


Figura 25. Boquillas con sensores en la llenadora volumétrica. Fuente: Adaptado de OCME (s.f.).

Las llenadoras ponderales (véase la Figura 26) se basan en el peso del contenido. Cada estación de llenado tiene una celda de carga que permite pesar el producto en tiempo real, tras haber realizado previamente la tara. Estas llenadoras pueden incorporar dosificadores de tornillo sinfín especializados para pesaje y la dosificación de polvos. Su microalimentador, accionado por servomotor, puede alcanzar exactitudes de hasta ± 0.003 g.

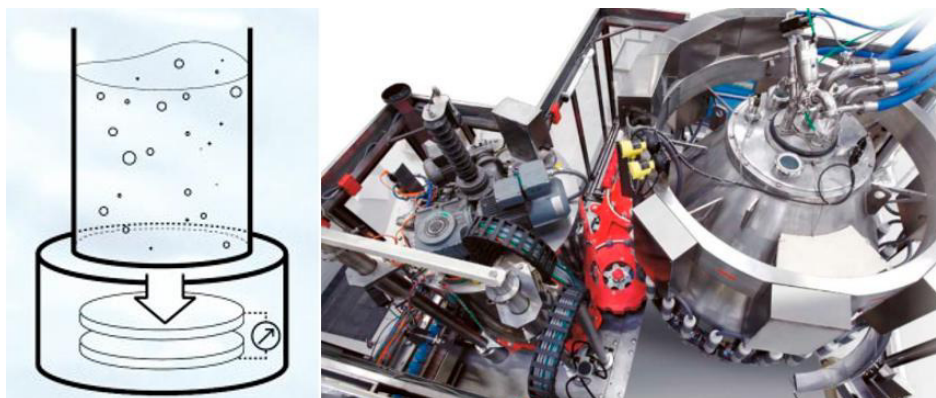


Figura 26. Sistema de llenadora ponderal. Fuente: Adaptado de OCME (s.f.).

II.4. Definición de términos

II.4.1. Plataformas remotas o en la nube

Son servicios tecnológicos que proporcionan a las organizaciones la capacidad de acceder de forma remota a recursos informáticos como servidores, almacenamiento de datos, herramientas de análisis o software, a través de internet. Esta modalidad elimina la necesidad de instalación o mantenimiento de equipos físicos en las propias instalaciones, ofreciendo escalabilidad, flexibilidad y accesibilidad desde cualquier punto geográfico. Además, el proveedor se encarga de las actualizaciones y mejoras de la plataforma, y en muchos casos, el cobro se realiza en función del uso real del usuario. Entre estas soluciones destacan Grafana Cloud e InfluxDB Cloud, dos plataformas en la nube que se complementan entre sí.

II.4.2. Node-RED

Es una herramienta de programación sencilla, basada en JavaScript, que permite crear sistemas conectados de forma visual. Para acceder al editor, sólo se debe ingresar la dirección IP del dispositivo donde está instalado, seguida del puerto 1880, a través de un navegador web. La interfaz mostrada en la Figura 27 permite arrastrar nodos desde el lado izquierdo, los cuales realizan tareas como recibir datos, procesarlos y enviarlos a otros destinos. Es ampliamente utilizada en proyectos de automatización y monitoreo, ya que permite conectar sensores, bases de datos, servicios web y otros dispositivos sin necesidad de escribir mucho código.

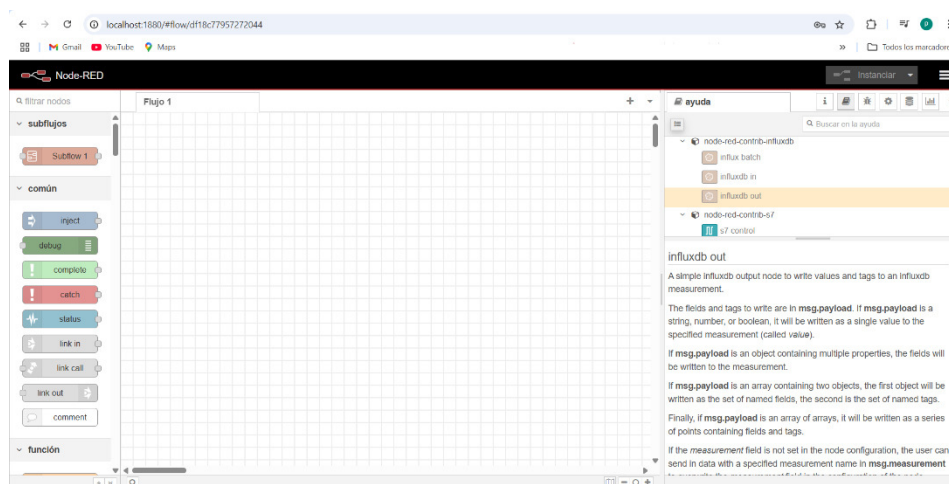


Figura 27. Interfaz de Node-RED.

II.4.3. Dashboard

Es una herramienta visual que permite mostrar información clave en una pantalla de forma clara y dinámica. Su función principal es agilizar el monitoreo y facilitar la comprensión de los datos, con el objetivo de apoyar la toma de decisiones acertadas. Estos paneles generalmente incluyen medidores, tablas o indicadores que reflejan el estado de un sistema. Los dashboards son ampliamente utilizados en sectores como la industria, la gestión de proyectos y el análisis del rendimiento, ya que proporcionan una visión general del funcionamiento del sistema. Por ejemplo, Grafana Cloud (véase la Figura 20), es una destacada herramienta de dashboard en la nube.

III. HIPÓTESIS Y VARIABLES

III.1. Hipótesis

III.1.1. Hipótesis general

Es posible el diseño e implementación de un sistema local y remoto para el monitoreo y almacenamiento de variables asociadas a servomotores y KPIs en el proceso de llenado de botellas.

III.1.2. Hipótesis específicas

- a. Es posible implementar un sistema local de monitoreo de las variables dinámicas de los servomotores, a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas.
- b. Es posible implementar un sistema local de almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas.
- c. Es posible implementar un sistema remoto de monitoreo de KPIs, mediante un dashboard en Grafana Cloud, en el proceso de llenado de botellas.
- d. Es posible implementar un sistema remoto de almacenamiento de las variables necesarias para el cálculo de KPIs en InfluxDB Cloud, en el proceso de llenado de botellas.

III.2. Variables

III.2.1. Variable independiente

La Variable Independiente de esta investigación es el Sistema Local y Remoto para el Monitoreo y Almacenamiento de Variables. Esta variable constituye una propuesta tecnológica central, que se compone de un nivel local, que incluye la interfaz de monitoreo desarrollada en WinCC Professional y el almacenamiento mediante SQL Server, y un nivel remoto, basado en el panel visual en Grafana Cloud y el almacenamiento en la nube a través de InfluxDB Cloud.

III.2.2. Variable dependiente

La Variable Dependiente está conformada los parámetros cuyo registro y análisis son posibles gracias al sistema implementado. Esta variable se divide en dos grupos principales. El primero son las Variables Dinámicas de los Servomotores, que comprenden datos esenciales como la Velocidad, la Posición y el Torque. El segundo grupo son los Indicadores Clave de Rendimiento (KPIs), que son métricas de alto nivel que evalúan la eficiencia global del proceso de llenado, tales como la Tasa de disponibilidad, la Tasa de calidad, la Tasa de utilización de capacidad y la Productividad.

III.3. Operacionalización de las variables

A continuación, se muestran las variables con sus respectivas dimensiones e indicadores.

Cuadro 1: Operacionalización de las variables.

Variable de estudio	Indicadores
Variable independiente:	1. Implementar un sistema local de monitoreo mediante una interfaz desarrollada en WinCC Professional.
Sistema local y remoto	2. Implementar un sistema local de almacenamiento utilizando SQL Server.
	3. Implementar un sistema remoto de monitoreo mediante un panel visual (dashboard) en Grafana Cloud.
	4. Implementar un sistema remoto de almacenamiento en la nube mediante InfluxDB Cloud.
Variable dependiente:	1.1 Velocidad (lineal).
	1.2 Posición (lineal).
Variables dinámicas de los servomotores	1.3 Torque (momento de fuerza).
	2.1 Tasa de utilización de capacidad.
KPIs (Indicadores claves de rendimiento)	2.2 Tasa de calidad.
	2.3 Tasa de disponibilidad.
	2.4 Productividad.

IV. MATERIALES Y MÉTODOS

IV.1. Área de estudio

La unidad de análisis seleccionada para el desarrollo y la validación de la presente investigación es el prototipo de llenado de botellas. Esta herramienta experimental fue desarrollada por el Grupo de Investigación TIC Aplicadas a la Sociedad (TAS), el cual se encuentra adscrito a la Facultad de Ingeniería Electrónica y Eléctrica (FIEE) de la Universidad Nacional Mayor de San Marcos (UNMSM). El prototipo constituyó el entorno experimental ideal, ya que su infraestructura posee la capacidad de replicar las condiciones de un proceso industrial real. Dispone de los servomotores y sistemas de control necesarios para simular la operación de llenado, lo que permitió disponer de un escenario idóneo para la adquisición, el monitoreo y el almacenamiento de las variables dinámicas y los KPIs fundamentales requeridos para validar el sistema propuesto.

IV.2. Diseño de la investigación

Según Hernández-Sampieri et al. (2014), fue un estudio en el que se manipularon intencionalmente una o más variables independientes para analizar las consecuencias que la manipulación tuvo sobre una o más variables dependientes. Esta investigación fue de tipo experimental. La variable independiente correspondió al sistema local y remoto, mientras que las dependientes fueron las relacionadas con los servomotores y los KPIs. Se implementó un sistema que, de manera local, registró en SQL Server las variables

dinámicas de los servomotores (velocidad, posición y torque) y las visualizó mediante una pantalla desarrollada en WinCC Professional. De forma remota, los KPIs de tasa de utilización, calidad, disponibilidad y productividad se almacenaron en InfluxDB Cloud y pudieron ser monitoreados a través de un dashboard en Grafana Cloud.

IV.3. Población y muestra

IV.3.1. Población

La población estuvo conformada por equipos de monitoreo y almacenamiento de variables de procesos industriales.

IV.3.2. Muestra

La muestra estuvo conformada por el prototipo de llenado de botellas, desarrollado por el grupo de investigación TAS de la Facultad de Ingeniería Electrónica y Eléctrica (FIEE) de la Universidad Nacional Mayor de San Marcos.

IV.4. Procedimiento, técnicas e instrumentos de recolección de información

IV.4.1. Descripción del sistema propuesto

Se propuso implementar un sistema de monitoreo y almacenamiento basado en un prototipo de llenado de botellas (véase la Figura 28). A nivel local, el sistema estuvo compuesto por una pantalla desarrollada en WinCC Professional conectada a SQL Server, donde se registraron las variables de los servomotores, como la velocidad, posición y torque. En paralelo, a nivel remoto, las variables necesarias para el cálculo de

los KPIs fueron almacenadas en InfluxDB Cloud y, posteriormente, enviadas a Grafana Cloud, donde se realizó el cálculo automatizado de estos indicadores y su visualización gráfica mediante un dashboard interactivo. Cabe destacar que se utilizaron licencias de pruebas gratuitas.

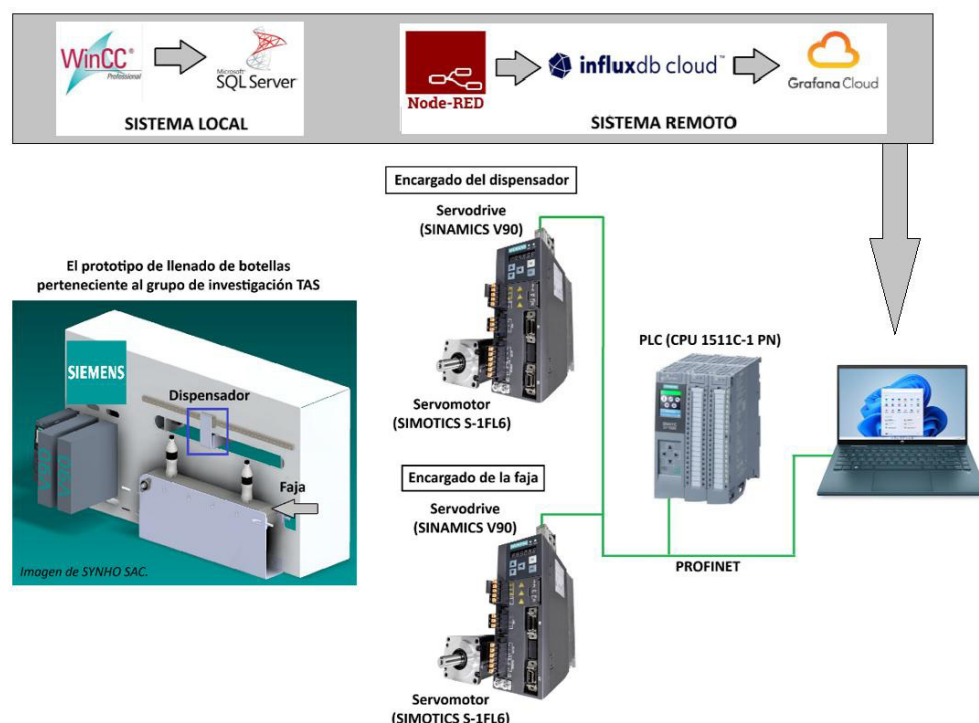


Figura 28. Sistema propuesto.

Según SYNHO SAC (2024), el prototipo de llenado de botellas, perteneciente al grupo de investigación TAS de la Facultad de Ingeniería Electrónica y Eléctrica, cuenta con dos servomotores: uno encargado de mover linealmente el dispensador y otro que controla el movimiento de la faja transportadora. En dicha demo se emplean dos sensores digitales (véase la Figura 29): uno ubicado en el extremo izquierdo, denominado sensor de sincronismo, cuya función es detectar la presencia de una botella para iniciar el proceso de sincronización entre el dispensador y la faja durante el llenado; y otro sensor ubicado en la boquilla del dispensador, encargado de detectar la botella antes de proceder al llenado.



Figura 29. Prototipo de llenado de botellas.

En la demo se destaca el uso del CPU 1511C-1 PN, dos SINAMICS V90 con sus respectivos SIMOTICS S-1FL6. A continuación, se describen dichos componentes:

CPU 1511C-1 PN (6ES7511-1CK01-0AB0)

Es un CPU de la marca Siemens, tal como se muestra en la Figura 30, que requiere una tensión de alimentación de 24V. Cuenta con 16 entradas digitales, 16 salidas digitales, 5 entradas analógicas, 2 salidas analógicas y E/S rápidas, con puerto PROFINET.



Figura 30. CPU 1511C-1 PN (6ES7511-1CK01-0AB0).

SINAMICS V90 (6SL3210-5FB10-2UF2)

El SINAMICS V90 (véase la Figura 31) cuenta con puerto PROFINET y acepta una tensión de entrada de 200-240 V AC, tanto monofásica como trifásica. Tiene un grado de protección IP20. Incluye un conector de alimentación de 24V/STO, un conector de alimentación de red, un conector de alimentación del motor, un conector para la resistencia de frenado, una interfaz mini USB, una ranura para tarjetas micro SD, una interfaz de control/estado y una interfaz de encoder.



Figura 31. SINAMICS V90 (6SL3210-5FB10-2UF2).

SIMOTICS S-1FL6 (1FL6032-2AF2x-xAA\Gx)

Empleó una tensión trifásica de 230 V, contó con un eje de 30 mm, una velocidad nominal de 3000 rpm, un *encoder* incremental de TTL de 2500 pulsos y un grado de protección IP65.



Figura 32. SIMOTICS S-1FL6 (1FL6032-2AF2x-xAA\Gx).

IV.4.1.1. Propuesta de sistema local

Para el presente sistema se utilizó una laptop HP Pavilion 15-cw1xxx (ver Figura 33), con sistema operativo Microsoft Windows 11 Home y 32 GB de RAM, la cual permitió realizar configuraciones entre los diferentes softwares locales, además de visualizar la pantalla desarrollada en WinCC Professional V17.

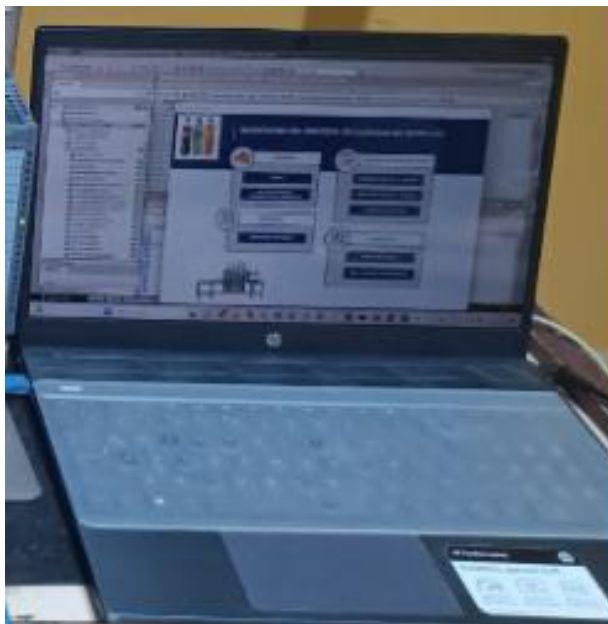


Figura 33. Laptop HP Pavilion.

En la laptop fue necesario contar con el software TIA Portal V17 (véase la Figura 34), desarrollado por Siemens, que integraba una serie de aplicaciones orientadas al control y monitoreo de sistemas automatizados. Este software permitió configurar y establecer comunicaciones entre el PLC y los servodrive. Además, se desarrolló una pantalla en WinCC Professional V17, la cual, mediante el uso de scripts, se comunicó con la información almacenada en SQL Server.

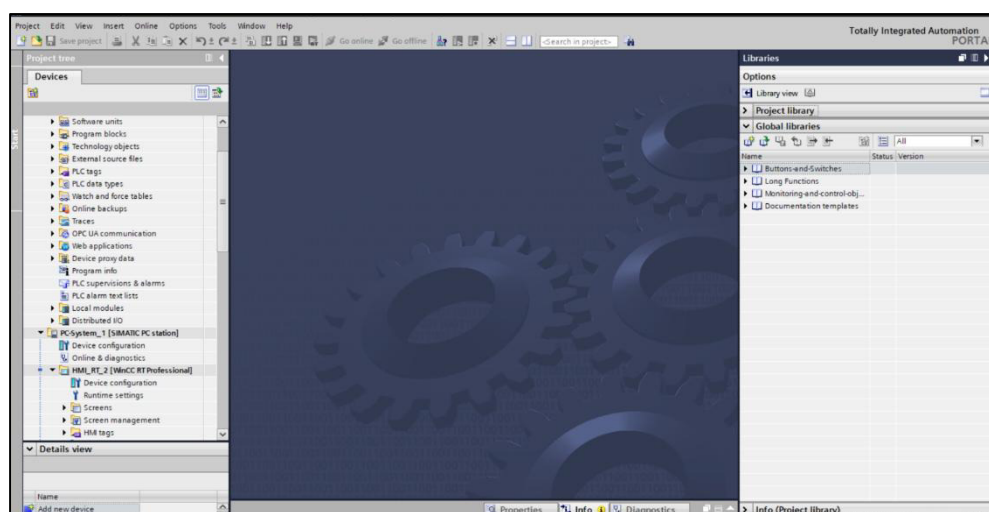


Figura 34. TIA Portal V17.

Para almacenar y organizar los datos generados por los servomotores se usará el sistema de gestión relacional de bases de datos SQL Server.



Figura 35. Logo de SQL Server.

Para interactuar con SQL Server de forma visual y práctica, se usará SQL Server Management Studio (SSMS) (véase la Figura 36), que es una herramienta de administración desarrollada por Microsoft que permite crear, gestionar y consultar bases de datos alojadas en SQL Server.

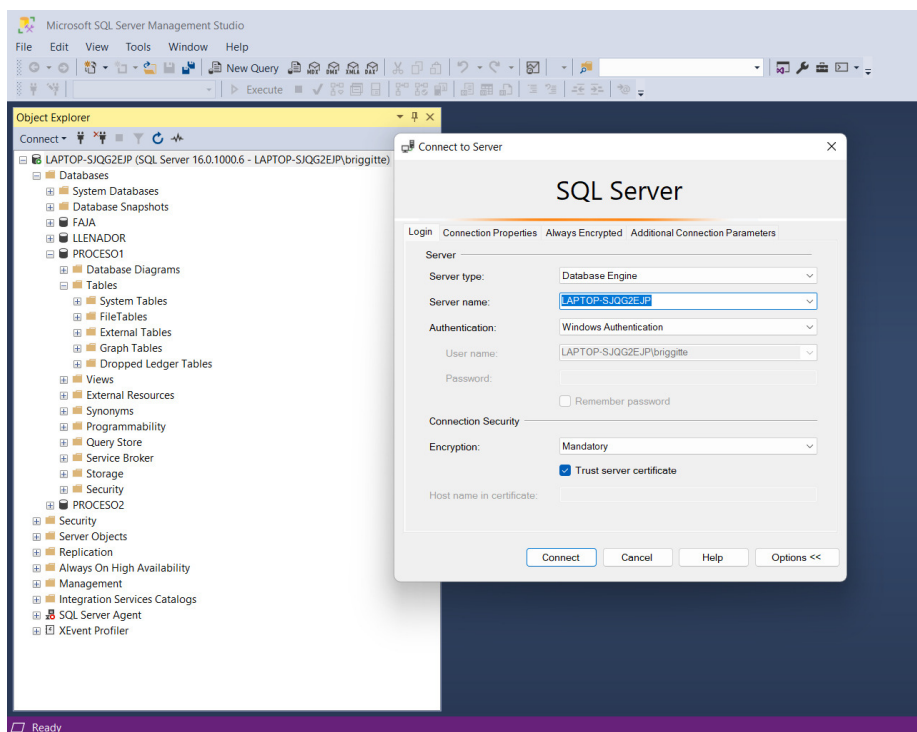


Figura 36. SQL Server Management Studio (SSMS).

Entonces, se propone implementar un sistema local (véase la Figura 37) que utiliza SQL Server para almacenar variables dinámicas de los servomotores, tales como velocidad, posición y torque. Estas variables provienen de los parámetros de los objetos tecnológicos configurados para cada servomotor en TIA Portal.

A través de una interfaz desarrollada en WinCC Professional, se monitorean dichas variables, además de permitir la creación o eliminación bases de datos y tablas, el inicio del almacenamiento de las variables dinámicas y la visualización de las tablas almacenadas en SQL Server.

Cada servomotor SIMOTICS S-1FL6 (1FL6032-2AF2x-xAA\Gx) se conecta a su respectivo servodrive SINAMICS V90 (6SL3210-5FB10-2UF2). Posteriormente, los dos servodrives se comunican con la CPU 1511C-1 PN (6ES7511-1CK01-0AB0) mediante PROFINET sobre Industrial Ethernet. Asimismo, se establece comunicación entre el PLC y la laptop HP Pavilion.

Los scripts de WinCC Professional se comunican con bases de datos mediante la tecnología ActiveX Data Objects (ADO). ADO funciona como un intermediario, facilitando la lectura, escritura y ejecución de comandos. Para establecer esta comunicación, ADO utiliza Open Database Connectivity (ODBC), que permite conectar diversas aplicaciones, como WinCC, con diferentes tipos de bases de datos como SQL Server. Finalmente, la conexión se concreta a través de un Data Source Name (DSN). Este DSN es una configuración guardada en Windows que contiene la información para acceder a la base de datos, como la dirección del servidor, el nombre de la base de datos, las credenciales de usuario y el tipo de controlador empleado.

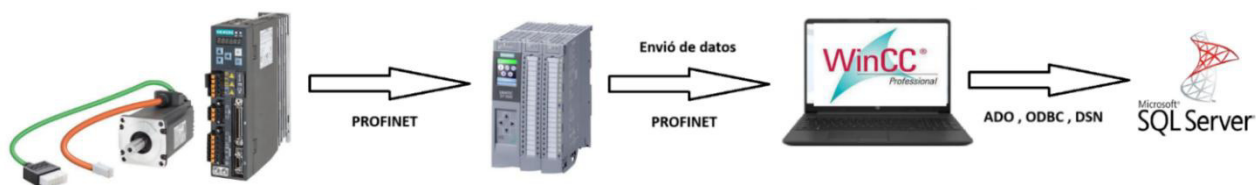


Figura 37. Sistema local.

IV.4.1.2. Propuesta de sistema remoto

De igual manera se empleará la laptop HP Pavilion 15-cw1xxx para realizar las configuraciones de las plataformas en la nube y TIA Portal V17. Para este sistema también será necesario usar Node-RED, que es una herramienta de programación visual de flujos desarrollada por IBM. Está construida sobre Node.js y proporciona una interfaz gráfica basada en navegador web, donde los usuarios pueden conectar nodos para desarrollar lógicas de automatización o flujos de intercambio de información entre diferentes sistemas.

En el contexto de la presente tesis, Node-RED funcionará como puente de comunicación (véase la Figura 38) entre el entorno local configurado en TIA Portal V17 y la base de datos en la nube InfluxDB Cloud, permitiendo la transferencia de datos generados durante la ejecución del programa.

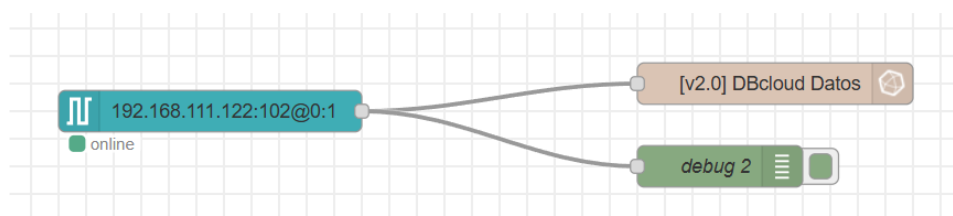


Figura 38. Enlace de Node-RED con TIA Portal y InfluxDB Cloud.

InfluxDB Cloud es un sistema de base de datos en la nube desarrollado por InfluxData, especializado en el manejo de datos de series temporales. En esta tesis, InfluxDB Cloud (véase la Figura 39) se utilizará como un repositorio remoto que almacenará las variables necesarias para el cálculo de los KPIs, los cuales serán procesados y visualizados en Grafana Cloud.

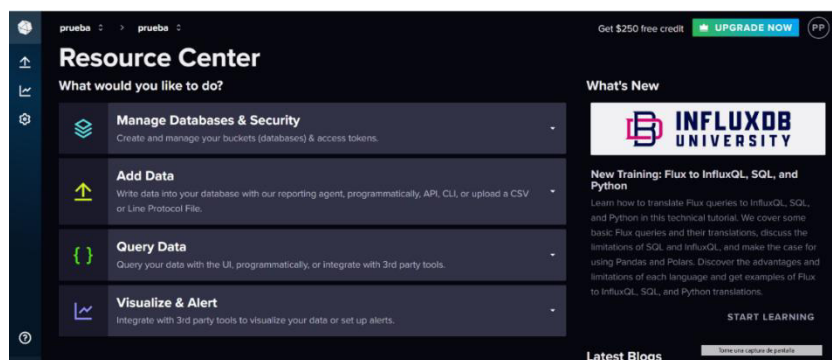


Figura 39. Centro de recursos de InfluxDB Cloud.

Grafana Cloud (véase la Figura 40) es una plataforma basada en la nube que proporciona potentes herramientas para visualizar y analizar datos. Su principal funcionalidad es la creación de dashboards interactivos y personalizables, ideales para el monitoreo y análisis de aplicaciones y procesos. En esta investigación, se empleará Grafana Cloud para calcular y visualizar los KPIs del proceso.

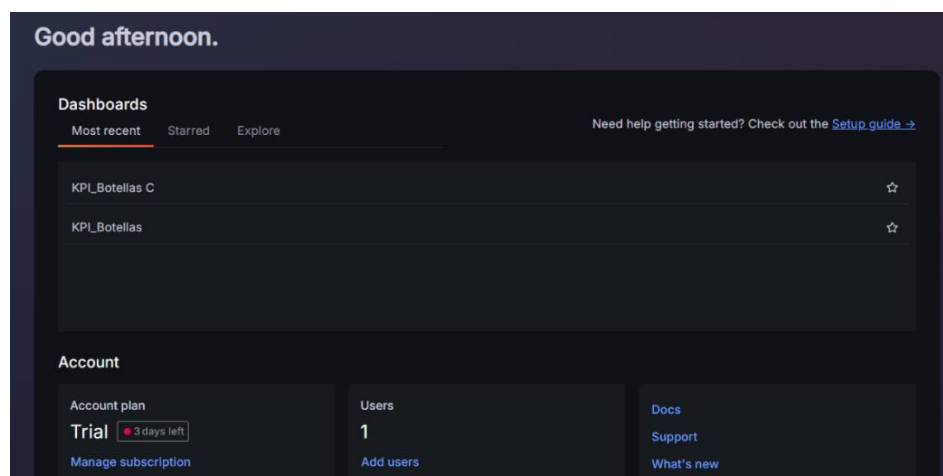


Figura 40. Recursos en Grafana Cloud.

Entonces, se implementará un sistema remoto, mostrado en la Figura 41, que permitirá el envío de datos tales como el tiempo, cantidad total de botellas, cantidad de botellas que cumplen las condiciones de calidad y cantidad de botellas consideradas como merma, desde TIA Portal hasta InfluxDB Cloud, usando como intermediario a Node-RED, el cual se comunicará con el PLC mediante el nodo S7.

Posteriormente, InfluxDB Cloud se integrará con Grafana Cloud, otra plataforma en la nube que aplicará las fórmulas correspondientes a los KPIs sobre los datos almacenados y permitirá visualizar dinámicamente los indicadores y sus variaciones a lo largo del tiempo mediante un dashboard. Para establecer esta conexión, será necesario especificar en Grafana Cloud los parámetros de conexión de InfluxDB Cloud: la URL, la organización, el bucket y el API token.



Figura 41. Sistema remoto.

IV.4.2. Implementación del sistema propuesto

IV.4.2.1. Implementación del sistema local

Red PROFINET

En el contexto de esta tesis (véase la Figura 43), la red se estableció mediante cables Ethernet (véase la Figura 42). A través de PROFINET –Industrial Ethernet– Plant bus, la red se estableció mediante cables entre la laptop, que funcionó como estación de monitoreo y configuración, y el PLC, encargado de gestionar el proceso de llenado de botellas. Además, se empleó PROFINET – Industrial ethernet entre el PLC y los servodrivres SINAMICS V90, lo que permitió controlar los servomotores responsables del transporte y llenado de las botellas.



Figura 42. Cable ethernet.

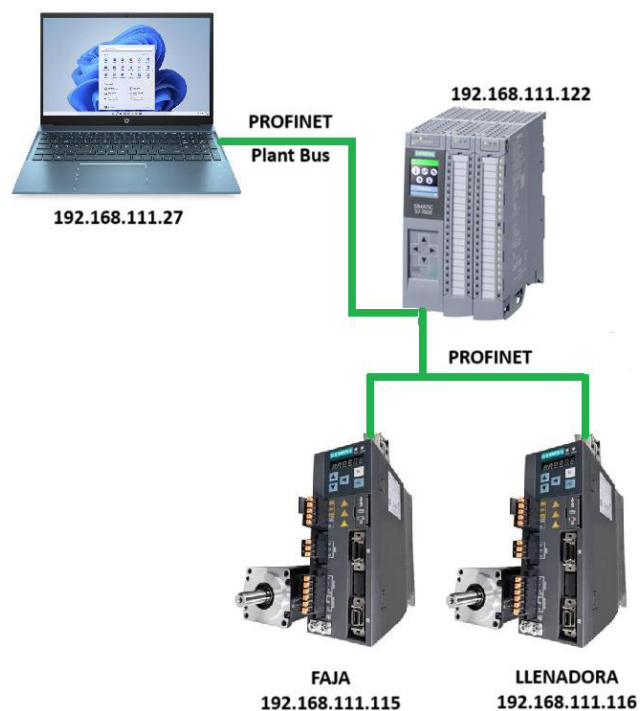


Figura 43. Esquema de red PROFINET.

Selección y configuración de los objetos tecnológicos

Para cada servomotor se seleccionó un objeto tecnológico en TIA Portal V17, ya que estos objetos facilitaron la programación de funciones tecnológicas utilizadas desde el controlador. Dentro de estos objetos se dispuso de una variedad de funciones relacionadas con el control de movimiento, que permitieron gestionar eficientemente los ejes del sistema.

Como se muestra en la Figura 44, para el servomotor que accionó la faja transportadora, se seleccionó el objeto tecnológico Eje de posicionamiento (TO_PositioningAxis). En el caso del servomotor que controló el dispensador, se utilizó el objeto tecnológico Eje sincronizado (TO_SynchronousAxis), ya que este eje debió coordinarse con el movimiento de la faja en cierto momento del proceso.

Cuando se activó el sensor de proximidad, denominado sensor de sincronismo, el dispensador reanudó su movimiento sincronizado. Además, contó con su propio sensor ubicado en la boquilla, encargado de detectar cuando una botella se encontró correctamente posicionada para iniciar el llenado. En ese momento, se inició la sincronización con la faja, de modo que mientras la botella se llenaba, la faja la transportaba a la misma velocidad a la que se desplazó el dispensador.

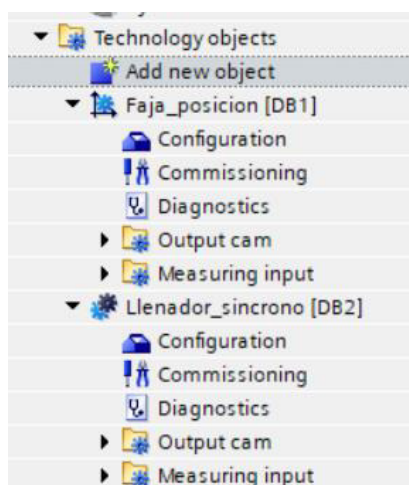


Figura 44. Objetos tecnológicos de cada servomotor.

Configuración de SQL Server

Se procedió a seleccionar la instancia predeterminada denominada MSSQLSERVER, ya que no se contempló realizar múltiples instalaciones del motor de base de datos en el mismo equipo.

Se configuró el modo de autenticación mixto, lo que permitió el acceso tanto mediante las credenciales del usuario de Windows como a través de usuarios específicos definidos en SQL Server. Durante la instalación, se estableció una contraseña para el usuario System Administrator (sa), quien actuó como administrador principal del sistema de base de datos.

Para verificar el correcto funcionamiento de SQL Server, se utilizó un cliente de administración de base de datos, en este caso SQL Server Management Studio 20 (SSMS20), el cual también fue instalado. Este software permitió establecer conexión con el motor de base de datos SQL Server para gestionar sus objetos y realizar consultas. Al momento de conectarse desde SSMS (véase la Figura 45), se seleccionó como tipo de servidor el Motor de base de datos. El nombre del servidor correspondió al nombre del equipo, y se mantuvo habilitado el modo de autenticación de Windows para facilitar el acceso inicial.

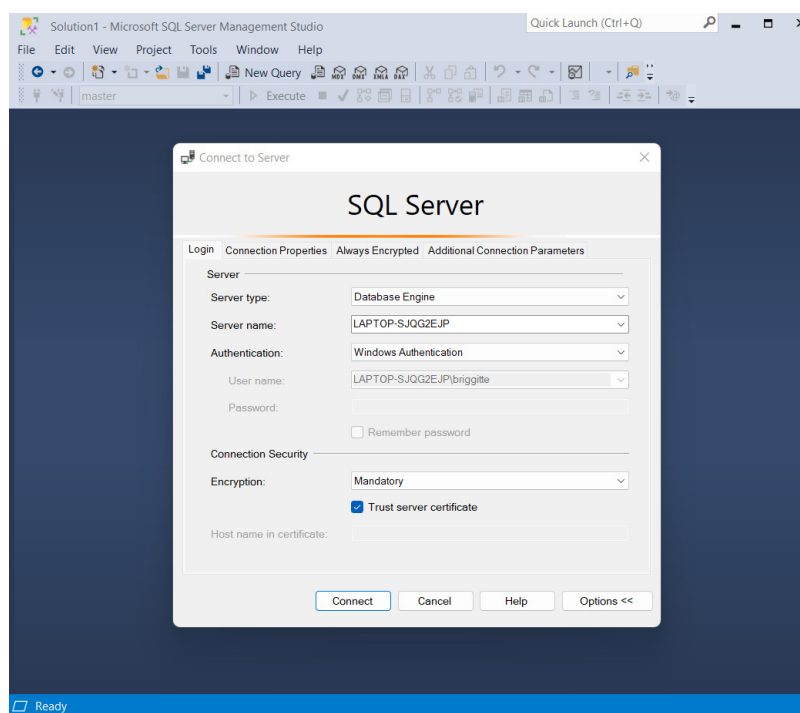


Figura 45. Inicio de sesión de SQL Server Management Studio 20 (SSMS20).

Luego de iniciar sesión, se pudieron observar las bases de datos y tablas creadas, como se observa en la Figura 46.

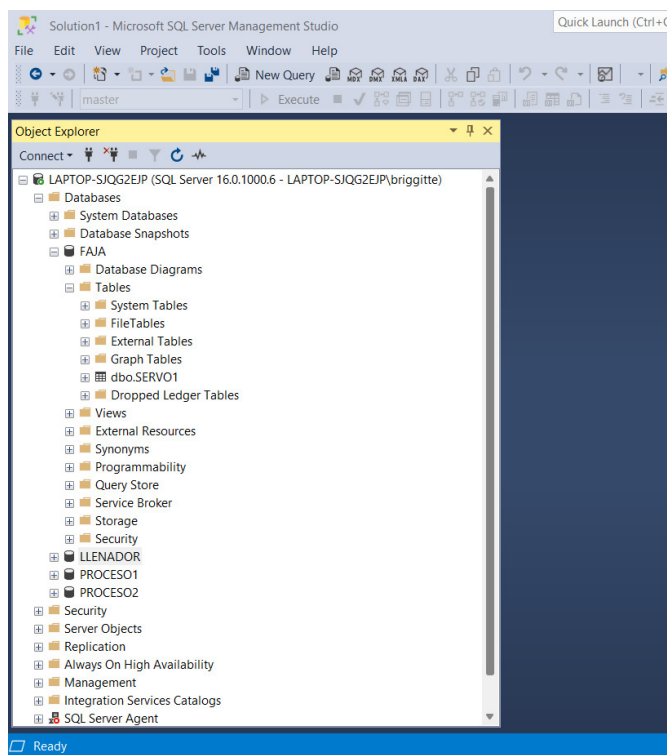


Figura 46. Bases de datos y tablas visualizadas desde SSMS20.

Desarrollo de la pantalla en WinCC Professional y de los scripts

Se seleccionó el dispositivo PC Station desde la lista de dispositivos disponibles al agregar un nuevo componente al proyecto de TIA Portal (véase la Figura 47).

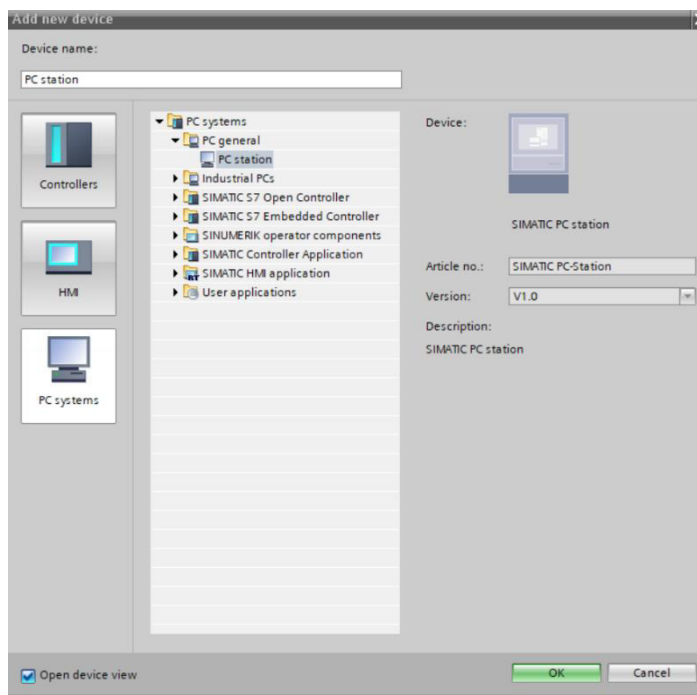


Figura 47. Añadir una PC station.

Luego se procedió a añadir un módulo de comunicación PROFINET, que en este caso fue una interfaz IE general. Esta interfaz permitió establecer la comunicación entre la estación PC y el PLC. Posteriormente, se seleccionó la aplicación SIMATIC HMI - WinCC Runtime Professional, que se integró dentro de la estación PC (véase la Figura 48). Esta aplicación fue la encargada de ejecutar las pantallas de visualización desarrolladas en WinCC, permitiendo el monitoreo y control de variables del sistema automatizado desde la PC.

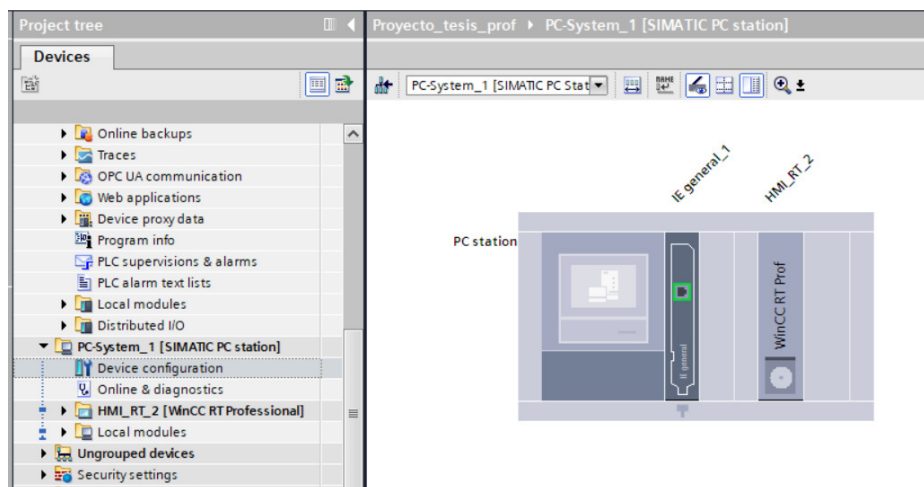


Figura 48. Configuración de la PC station.

Se procedió a configurar las siguientes imágenes:

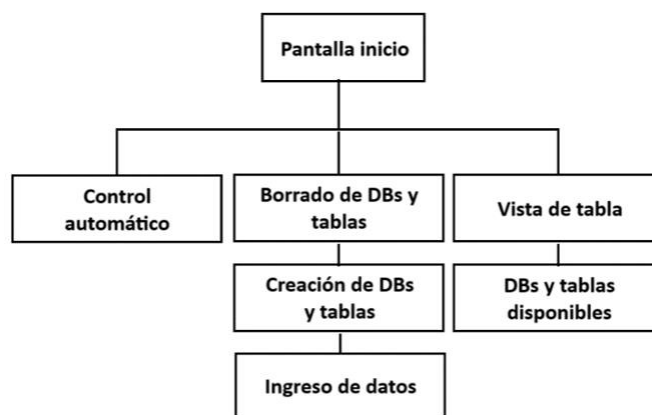


Figura 49. Diagrama de imágenes a usar.

Pantalla inicio

En esta pantalla se visualizaron todas las pantallas disponibles (véase la Figura 50) organizadas en tres grupos. Se pudo acceder a la pantalla de control automático, borrado de DBs y tablas, creación de DBs y tablas, ingreso de datos, vista de tabla, DBs y tablas disponibles.



Figura 50. Pantalla de inicio.

En el software TIA Portal, la pantalla definida como Imagen de Inicio se identifica visualmente mediante un triángulo verde (véase la Figura 51).



Figura 51. Designación de pantalla principal.

La velocidad y posición requeridas, que eran ingresadas por el usuario, se almacenaron en los respectivos bloques de datos (véase la Figura 52): DB_FAJA[DB20] y DB_LLENA[DB24], para después ser usadas en las instrucciones de control de movimiento.

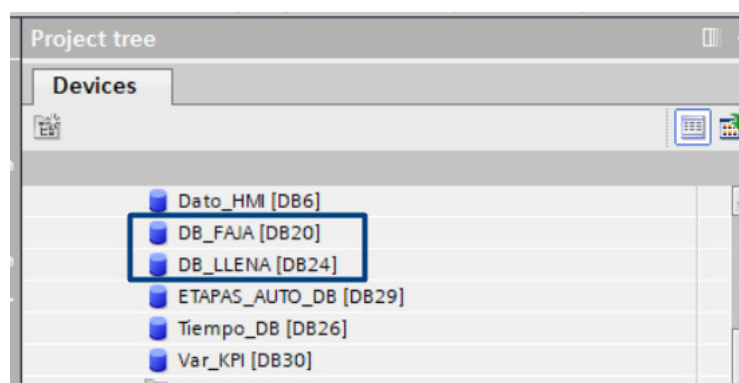


Figura 52. DB para almacenar variables de la faja y llenadora.

Los valores actuales que se mostraban en la pantalla provenían de DB_FAJA[DB20] y DB_LLENA[DB24], los cuales almacenaban los datos de los bloques de datos tecnológicos, como se visualiza en los siguientes segmentos de programación (véanse las Figuras 53 y 54).

```
2 "DB_FAJA".VelocidadActual := "Faja_posicion".ActualVelocity;
3 "DB_FAJA".PosicionActual := "Faja_posicion".ActualPosition;
4 "DB_FAJA".TorqueActual := "Faja_posicion".StatusTorqueData.ActualTorque;
```

Figura 53. Almacenamiento de los valores actuales de la velocidad, torque y posición de la faja.

```

2  "DB_LLENA".VelocidadActualS := "Llenador_sincrono".ActualVelocity;
3  "DB_LLENA".PosicionActualS := "Llenador_sincrono".ActualPosition;
4  "DB_LLENA".TorqueActualS := "Llenador_sincrono".StatusTorqueData.ActualTorque;

```

Figura 54. Almacenamiento de los valores actuales de la velocidad, torque y posición del llenador o dispensador.

Pantalla de control automático

Por medio de esta pantalla, que se muestra en la Figura 55, se inició el funcionamiento del prototipo de llenado de botellas. El campo de “Velocidad requerida” indicó la velocidad deseada para el transporte de las botellas, mientras que los valores actuales de la velocidad, aceleración y torque provinieron de los bloques DB_FAJA[DB20] y DB_LLENA[DB24]. Cuando se presionó el botón “Activar modo”, se inició el movimiento automático. La faja se desplazó continuamente a la velocidad establecida en el campo “Requerida”. Paralelamente, el llenador o dispensador comenzó su proceso únicamente cuando se activó el sensor de sincronismo y se detectó una botella mediante el sensor ubicado en la boquilla del llenador.

Durante el llenado, tanto la faja como el dispensador se movieron sincronizadamente a la misma velocidad, lo que permitió el llenado mientras la botella estuvo en movimiento. Finalizado el llenado, el dispensador retornó a su posición de origen a velocidad máxima, quedando a la espera de una nueva activación de sensores

para reiniciar el ciclo. En la parte superior de la pantalla se visualizó el número total de botellas contabilizadas durante el proceso.

The screenshot displays the 'MODO AUTOMÁTICO' (Automatic Mode) interface. At the top, there is a blue header with a gear icon and the text 'MODO AUTOMÁTICO'. Below this, a row contains three elements: a blue button labeled 'ACTIVAR MODO', a grey box labeled 'N° DE BOTELLA LLENADAS' with a white input field showing '0', and a blue button labeled 'RESETEAR CONTADOR'. The interface is divided into two main sections, each with a yellow header: 'FAJA' and 'LLENADOR'. Each section contains a status indicator on the left (a yellow box with a red 'X' and the text 'ESTADO Inoperativo' and a blue 'RESETEO' button) and a data panel on the right. The 'FAJA' data panel shows 'VELOCIDAD (mm/s)' with 'Requerida' and 'Actual' both at 0.00, 'POSICIÓN (mm)' at 0.00, and 'TORQUE (Nm)' at 0.00. The 'LLENADOR' data panel shows 'VELOCIDAD (mm/s)' with 'Maxima' at 200.00 and 'Actual' at 0.00, 'POSICIÓN (mm)' at 0.00, and 'TORQUE (Nm)' at 0.00.

Figura 55. Pantalla de control automático.

Pantalla de creación de DBs y tablas

En esta pantalla, que se muestra en la Figura 56, el usuario pudo especificar el nombre de la base de datos que deseaba crear, definir el nombre de la tabla y seleccionar la cantidad de columnas (máximo de cinco). Una vez definido el número de columnas, se desplegaron opciones para cada una. El usuario pudo asignar a cada columna una de las siguientes variables: Velocidad Faja (mm/s), Torque Faja (Nm), Posición Faja (mm), Velocidad Llenador (mm/s), Torque Llenador (Nm) o Posición Llenador (mm).

Esta pantalla permitió realizar la creación conjunta de una base de datos y una tabla, o bien crear solo una de ellas (ya sea la base o la tabla), utilizando una estructura existente. No fue obligatorio realizar ambas operaciones simultáneamente.

Al presionar los botones “CREAR”, se ejecutaron los scripts “CrearDB” y “CrearTabla”, según correspondió. Los resultados del proceso (ya fueran errores o confirmaciones de éxito) se mostraron en un recuadro inferior de la pantalla.

CREACIÓN DE DBS Y TABLAS

BASE DE DATOS **CREAR**

TABLA **Nº DE COLUMNAS**

COLUMNAS
Fecha
hor...

CREAR

INICIO **BORRADO DE DBs Y TABLAS** **INGRESO DE DATOS**

Figura 56. Pantalla de creación de DBs y tablas.

En el cuadro 2, se presenta la estructura del script “CrearDB”, donde se detalla el funcionamiento del mismo. En el Anexo B se encuentran las líneas del código.

Cuadro 2: Pseudocódigo del script CrearDB.

1. Inicio	
2. Inicializar mensaje de error	SmartTags("MensajeError") = "".
3. Crear objetos de conexión	connect = CreateObject("ADODB.Connection"). record = CreateObject("ADODB.Recordset").
4. Verificar si SmartTags("NombreDB") está vacío	Sí: Asignar mensaje "No se ha proporcionado nombre de la base de datos a crear." y finalizar. No: Continuar.
5. Conectar al servidor	Provider=MSDASQL; DSN=" & DSN & ";
6. Verificar si hubo error en la conexión	Sí: Asignar mensaje "No se pudo conectar a la base de datos." y finalizar. No: Continuar.
7. Verificar existencia de la base de datos	Ejecutar: SELECT COUNT (*) FROM sys.databases WHERE name = 'NombreDB'.
8. ¿La base de datos ya existe?	Sí: Asignar mensaje "La base de datos 'NombreDB' ya existe." y finalizar. No: Continuar.
9. Crear la base de datos	Ejecutar CREATE DATABASE NombreDB.
10. Verificar si hubo error al crear	Sí: ¿Error -2147217900?. Sí: Asignar mensaje "Nombre no válido del DB, contiene caracteres no válidos." y finalizar. No: Asignar mensaje "No se pudo crear la base de datos." y finalizar. No: Asignar mensaje "La base de datos 'NombreDB' se ha creado correctamente.".
11. Finalización	Cerrar conexión con connect.Close, liberar objetos Set connect = Nothing, Set record = Nothing.
12. Fin	

En el cuadro 3, se muestra la composición del script “CrearTabla”, describiendo su modo de operación. En el Anexo C se encuentran las líneas del código.

Cuadro 3: Pseudocódigo del script CrearTabla.

1. Inicio	
2. Inicializar mensaje de error	SmartTags("MensajeError") = "".
3. Crear objetos de conexión	connect = CreateObject("ADODB.Connection"). record = CreateObject ("ADODB.Recordset").
4. Validar campos obligatorios	Si NombreDB, NombreTabla están vacíos o Ncolumnas = 0, asignar mensaje de error correspondiente y finalizar.
5. Conectar a base de datos master usando DSN	connect.Open "Provider=MSDASQL;Initial Catalog=master;DSN=...". Si hay error: asignar mensaje "No se pudo conectar." y finalizar.
6. Verificar si existe la base de datos	Ejecutar: SELECT COUNT (*) FROM sys.databases WHERE name = 'NombreDB'. Si no existe: asignar mensaje "La base de datos 'NombreDB' no existe." y finalizar.
7. Conectar a la base de datos indicada (NombreDB)	connect.Open "Provider=MSDASQL;Initial Catalog=NombreDB;DSN=...". Si hay error: asignar mensaje "No se pudo conectar a la base de datos 'NombreDB'." y finalizar.
8. Verificar si la tabla ya existe	Ejecutar: SELECT COUNT (*) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'NombreTabla'. Si ya existe: asignar mensaje "La tabla 'NombreTabla' ya existe. Cambie el nombre." y finalizar.
9. Construir dinámicamente la lista de columnas	Empezando con "Fecha_hora DATETIME", según el número de columnas (Ncolumnas) y los valores seleccionados en Columna1, Columna2, etc.
10. Armar sentencia SQL	CREATE TABLE NombreTabla (Fecha_hora DATETIME, ColumnaX FLOAT, ...).
11. Ejecutar la creación de tabla	Si hay error: ¿Error -2147217900?: asignar mensaje "Nombre no válido de la tabla". Otro error: asignar mensaje "No se pudo crear la tabla." y finalizar. Si no hay error: asignar mensaje "La tabla 'NombreTabla' ha sido creada exitosamente en la base de datos 'NombreDB'.".
12. Finalización	Cerrar conexión con connect.Close, liberar objetos Set connect = Nothing, Set record = Nothing.
13. Fin	

Pantalla de borrado de DBs y tablas

En esta pantalla, que se muestra en la Figura 57, se pudo especificar la base de datos que se deseaba eliminar presionando el botón “Borrar”, el cual ejecutó el script “BorrarDB”. De igual forma, al indicar el nombre de una tabla y presionar el botón “Borrar”, ubicado a la derecha del campo donde se ingresó dicho nombre, se ejecutó el script “Borrar tabla”.

The screenshot displays a web application interface for deleting databases and tables. The title bar is blue with a yellow database icon and the text "BORRADO DE DBS Y TABLAS". The main area has a light gray dotted background. It features two yellow horizontal bands. The first band is labeled "BASE DE DATOS" and contains a white text input field followed by a blue button labeled "BORRAR". The second band is labeled "TABLA" and also contains a white text input field followed by a blue button labeled "BORRAR". Below these bands is a large white rectangular area for a message or log, preceded by a blue speech bubble icon. At the bottom, a black bar contains three gray buttons: "INICIO", "CREACIÓN DE DBs Y TABLAS", and "INGRESO DE DATOS".

Figura 57. Pantalla de borrado de DBs y tablas.

En el cuadro 4, se muestra la composición del script “BorrarDB”, describiendo su modo de operación. En el Anexo D se encuentran las líneas del código.

Cuadro 4: Pseudocódigo del script BorrarDB.

1. Inicio	
2. Inicializar mensaje de error	SmartTags("MensajeError10") = "".
3. Crear objetos de conexión	conn = CreateObject("ADODB.Connection"). rst = CreateObject("ADODB.Recordset").
4. Validar campo obligatorio	Si NombreDB10 está vacío, asignar mensaje "No se ha proporcionado nombre de la base de datos a borrar." y finalizar.
5. Conectar al servidor de base de datos usando DSN	conn.Open "Provider=MSDASQL; DSN=..." Si hay error: asignar mensaje "No se pudo conectar a la base de datos." y finalizar.
6. Armar sentencia SQL para eliminar base de datos	DROP DATABASE NombreDB10.
7. Ejecutar la eliminación de base de datos	Si hay error: ¿Error -2147217865?: asignar mensaje "La base de datos 'NombreDB10' no existe." Otro error: asignar mensaje "No se pudo borrar la base de datos." y finalizar.
8. Confirmar eliminación exitosa	Si no hay error: asignar mensaje "La base de datos 'NombreDB10' se ha borrado correctamente."
9. Finalización	Cerrar conexión con conn.Close, liberar objetos Set conn = Nothing, Set rst = Nothing.
10. Fin	

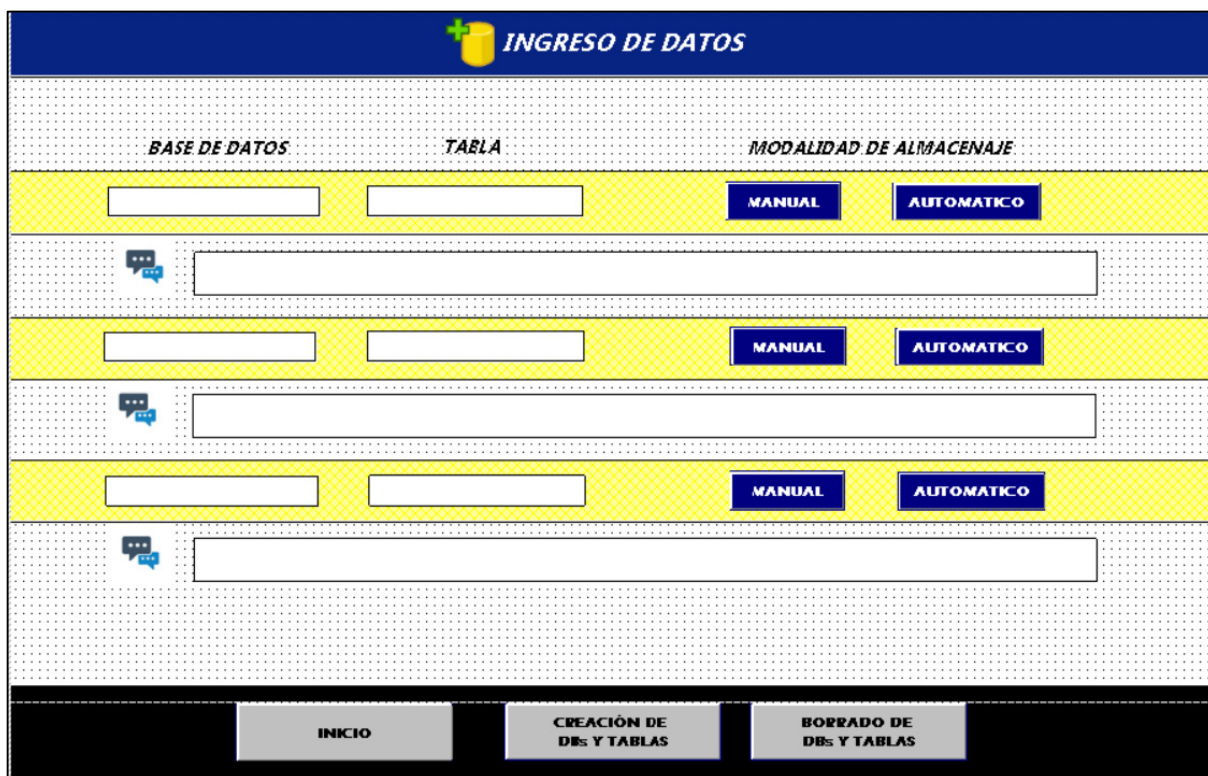
En el cuadro 5, se muestra la composición del script “BorrarTabla”, donde se detalla el funcionamiento del mismo. En el Anexo E se encuentran las líneas del código.

Cuadro 5: Pseudocódigo del script BorrarTabla.

1. Inicio	
2. Inicializar mensaje de error	SmartTags("MensajeError10") = "".
3. Crear objetos de conexión	Set connect = CreateObject("ADODB.Connection"). Set record = CreateObject("ADODB.Recordset").
4. Verificar si se ha proporcionado el nombre de la base de datos y la tabla	Si SmartTags("NombreDB10") o SmartTags("NombreTabla10") están vacíos, asignar mensaje de error y finalizar. connect.Open "Provider=MSDASQL;Initial
5. Conectar a la base de datos "master"	Catalog=master;DSN=" & DSN & ";".
6. Verificar error en la conexión	Si hay error, asignar mensaje de error y finalizar.
7. Verificar si la base de datos existe	Ejecutar SELECT COUNT (*) FROM sys.databases WHERE name = 'NombreDB10'.
8. ¿La base de datos existe?	Si no existe, asignar mensaje de error y finalizar.
9. Conectar a la base de datos indicada	connect.Open "Provider=MSDASQL;Initial Catalog=" & SmartTags("NombreDB10") & ";DSN=" & DSN & ";".
10. Verificar error en la conexión a la base de datos específica	Si hay error, asignar mensaje de error y finalizar.
11. Verificar si la tabla existe	Ejecutar SELECT COUNT (*) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'NombreTabla10'.
12. ¿La tabla existe?	Si no existe, asignar mensaje de error y finalizar.
13. Borrar la tabla	Ejecutar SQLDropTable = "DROP TABLE " & SmartTags("NombreTabla10") y connect.Execute(SQLDropTable).
14. Verificar error al borrar la tabla	Si hay error, asignar mensaje de error y finalizar. Si no, asignar mensaje de éxito.
15. Cerrar la conexión	connect.Close , Set connect = Nothing, Set record = Nothing.
16. Fin	

Pantalla de ingreso de datos

Se seleccionó la base de datos y la tabla correspondiente. Si el usuario presionaba el botón “Manual” (véase la Figura 58), el almacenamiento de datos se ejecutaba una sola vez. En cambio, si se presionaba el botón “Automático”, el script “InsertarDataAUTO” comenzaba a ejecutarse de forma periódica. Al volver a presionar el mismo botón, se detenía la ejecución del script, ya que funcionaba como un interruptor.



INGRESO DE DATOS		
BASE DE DATOS	TABLA	MODALIDAD DE ALMACENAJE
		MANUAL AUTOMATICO
		
		MANUAL AUTOMATICO
		
		MANUAL AUTOMATICO
		

Figura 58. Pantalla de ingreso de datos.

En el cuadro 6, se muestra la composición del script “InsertarDataMAN”, describiendo su modo de operación para almacenar datos cada vez que se presionaba el botón “Manual”. En el Anexo F se encuentran las líneas del código.

Cuadro 6: Pseudocódigo del script InsertarDataMAN.

1. Inicio	
2. Inicializar mensaje de error	SmartTags("MensajeError1") = "".
3. Verificar si AutoMode está activado	Si SmartTags("AutoMode1") = True, finalizar, ya que se ejecuta en modo automático.
4. Obtener los datos de la base de datos y tabla desde SmartTags	DBName = SmartTags("NombreDB1"). TableName = SmartTags("NombreTabla1").
5. Crear objetos de conexión	connect = CreateObject("ADODB.Connection"). record = CreateObject("ADODB.Recordset").
6. Establecer conexión con la base de datos	Conectar a la base de datos usando connect.Open "Provider=MSDASQL;Initial Catalog=" & DBName & ";DSN=" & DSN & ";". Si hay error al conectar, asignar mensaje de error y finalizar. Si la conexión es exitosa, mostrar mensaje "Conexión exitosa a la base de datos: " & DBName.
7. Verificar si la tabla existe en la base de datos	Ejecutar consulta SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = "" & TableName & ". Si la tabla no existe, asignar mensaje de error y finalizar.
8. Preparar listas de columnas y valores	Inicializar listas: columnList = "fecha_hora", valueList = "GETDATE()". Recorrer las columnas de la tabla y obtener valores de SmartTags. Asignar valores a las columnas correspondientes como "Velocidad_Faja_mm_s", "Torque_Faja_Nm", etc.
9. Construir la sentencia SQL de inserción	Crear la sentencia SQLinsert = "INSERT INTO " & TableName & " (" & columnList & ") VALUES (" & valueList & ")".
10. Ejecutar la sentencia SQL	Ejecutar la sentencia SQLinsert para insertar los datos en la tabla. Si hay error al insertar, asignar mensaje de error y finalizar. Si la inserción es exitosa, mostrar mensaje "Datos insertados correctamente en la tabla "" & TableName & ""."
11. Finalización	Cerrar la conexión con connect.Close. Liberar objetos: Set connect = Nothing, Set record = Nothing.
12. Fin	

En el cuadro 7, se muestra la composición del script “InsertarDataAUTO”, describiendo su modo de operación para almacenar datos cada vez que se presionaba el botón “Automático”. En el Anexo G se encuentran las líneas del código.

Cuadro 7: Pseudocódigo del script InsertarDataAUTO.

1. Inicio	
2. Inicializar mensaje de error	Se inicializa el mensaje de error en blanco mediante la instrucción: SmartTags("MensajeError1") = "".
3. Verificar si AutoMode está activado	Si SmartTags("AutoMode1") = 1, entonces se procede con la ejecución automática. En caso contrario, no se ejecuta el script.
4. Obtener los datos de la base de datos y tabla desde SmartTags	Se recuperan los valores de SmartTags("NombreDB1") y SmartTags("NombreTabla1") y se asignan a las variables DBName y TableName respectivamente.
5. Crear objetos de conexión	Se crean los objetos connect y record usando CreateObject("ADODB.Connection") y CreateObject("ADODB.Recordset").
6. Establecer conexión con la base de datos	Se intenta abrir la conexión a través de la cadena: connect.Open "Provider=MSDASQL;Initial Catalog=" & DBName & ";DSN=" & DSN & ";". Si ocurre un error al conectar, se asigna un mensaje de error y se termina el procedimiento. Si la conexión es exitosa, se muestra el mensaje: "Conexión exitosa a la base de datos: [nombre de la base]".
7. Verificar si la tabla existe en la base de datos	Se ejecuta la consulta SQL "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'nombreTabla'". Si el recordset está vacío (es decir, record.EOF = True), significa que la tabla no existe, se asigna el mensaje de error correspondiente y se finaliza el script.
8. Preparar listas de columnas y valores	Se inicializa columnList con "fecha_hora" y valueList con "GETDATE ()". Luego, se recorre cada columna del recordset. Si el nombre de la columna coincide con alguna variable de TIA Portal (por ejemplo, "Velocidad_Faja_mm_s", "Torque_Llenador_Nm", etc.), se obtiene el valor de la respectiva SmartTag. Si la columna tiene un valor válido, se agrega a las listas de columnas y valores para la sentencia SQL.
9. Construir la sentencia SQL de inserción	Con las listas generadas de columnas y valores, se construye la instrucción: INSERT INTO nombreTabla (columnas) VALUES (valores).
10. Ejecutar la sentencia SQL	Se ejecuta la instrucción de inserción. Si ocurre un error durante la ejecución, se asigna un mensaje de error con el número correspondiente. Si la ejecución es exitosa, se asigna el mensaje: "Datos insertados correctamente en la tabla '[nombre de la tabla]'".
11. Finalización	Se cierra la conexión con connect.Close y se liberan los objetos connect y record.
12. Fin	

Pantalla de vista de tabla

Cuando el usuario presionó el botón “Mostrar data” (véase la Figura 59), se visualizó todo el contenido de las tablas correspondientes, ordenado de manera ascendente.

The screenshot displays a web-based interface for viewing database tables. The top section features a yellow header with labels for 'BASE DE DATOS' and 'TABLA', each followed by a text input field. Below this is a blue navigation bar containing two buttons: 'MOSTRAR DATA' and 'OSTRAR DATA'. The main content area includes a table with two columns: 'Nº' and 'Hora y fecha'. The table contains several rows, each starting with '***' and followed by the timestamp '12/31/2002 10:59:59.010 AM'. To the right of the table, there are five buttons labeled 'Seleccione columna'. Below the table is a large, empty yellow-bordered rectangular box. In the center of the interface, there is a magnifying glass icon over a grid pattern, and to its right are two database cylinder icons. The bottom of the interface is a black bar with two buttons: 'INICIO' and 'DBs Y TABLAS DISPONIBLES'.

Figura 59. Pantalla de vista de tabla.

En el cuadro 8, se muestra la composición del script “MostrarTabla”, describiendo su modo de operación para mostrar toda la información almacenada en vistas de 8 filas en cada consulta. En el Anexo H se encuentran las líneas del código.

Cuadro 8: Pseudocódigo del script *MostrarTabla*.

1. Inicio	
2. Obtener las fechas mínima y máxima de la tabla	Ejecutar consulta: SELECT MIN (fecha_hora) AS FechaInicio, MAX (fecha_hora) AS FechaFin FROM " & tableName. Si la consulta tiene resultados: Asignar valores a SmartTags: SmartTags("Inicio1") = rs.Fields("FechaInicio").Value, SmartTags("Fin1") = rs.Fields("FechaFin").Value. Si no hay resultados: Asignar valores de error: SmartTags("Inicio1") = "No hay datos", SmartTags("Fin1") = "No hay datos".
3. Consultar las columnas de la tabla (excluyendo fecha_hora):	Ejecutar consulta: SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = " & tableName & " AND COLUMN_NAME <> 'fecha_hora' ORDER BY ORDINAL_POSITION. Si no se encuentran columnas: Asignar mensaje de error: SmartTags("MensajeMuestra2") = "No se encontraron información.".
4. Asignar valores a las columnas	Inicializar contador i = 1. Recorrer las columnas obtenidas y asignar valores a SmartTags: Para cada columna, usar Select Case para asignar valores específicos de cada columna (Velocidad_Faja_mm_s, Torque_Faja_Nm, etc.) a las SmartTags correspondientes. Incrementar el contador de columnas: NcolumnasMuestra = NcolumnasMuestra + 1.
5. Obtener los datos de la tabla, ordenados por fecha_hora:	Ejecutar consulta: SELECT * FROM " & tableName & " ORDER BY [fecha_hora] ASC. Si no se obtienen registros: Asignar mensaje de error: SmartTags("MensajeMuestra2") = "No se encontraron registros.".
6. Control de la paginación	Recuperar valor de Muestra desde SmartTags("Muestra1"). Calcular la fila de inicio: startRow = Muestra * 8 + 1. Mover el cursor de resultados hasta la fila de inicio: Usar un bucle For para mover el cursor startRow - 1 veces.
7. Mostrar las filas de la página (máximo 8 filas):	Inicializar contador de filas visibles visibleRowCount = 0. Usar un ciclo Do While Not rs.EOF And i <= 8 para recorrer las filas y asignar los valores a SmartTags: Asignar el valor de fecha_hora a SmartTags("HoraFecha" & i & "T"). Asignar los valores de las columnas según el número de columna presente en la tabla. Incrementar visibleRowCount. Mover al siguiente registro con rs.MoveNext.
8. Verificar si hemos llegado a la última fila	Si rs.EOF (fin de los registros), establecer: SmartTags("IsLastRow1") = True. Si no, establecer: SmartTags("IsLastRow1") = False.
9. Fin	

Pantalla de DBs y tablas disponibles

A través de esta pantalla (véase la Figura 60), el usuario pudo consultar las bases de datos disponibles y seleccionar una de ellas para explorar sus tablas y columnas asociadas. Para habilitar estas funcionalidades, se emplearon tres scripts específicos que gestionaron la recuperación y presentación de esta información.

The screenshot shows a web application interface for managing databases and tables. The header is blue with a database icon and the text "DBS Y TABLAS DISPONIBLES". The main area has a yellow background and contains three sections, each with a refresh icon and a text input field:

- BASE DE DATOS DISPONIBLES**: The first input field.
- BASE DE DATOS** and **TABLAS DISPONIBLES**: The second input field.
- TABLA** and **COLUMNAS DISPONIBLES**: The third input field.

At the bottom, there are two buttons: "INICIO" and "VISTA DE TABLA".

Figura 60. Pantalla de DBs y tablas disponibles.

En el cuadro 9, se muestra la composición del script “DBrefres”, describiendo su modo de operación para mostrar todos los DBs disponibles cada vez que se pulsaba el botón de consulta. En el Anexo I se encuentran las líneas del código.

Cuadro 9: Pseudocódigo del script DBrefres.

1. Inicio	
2. Crear objetos de conexión y recordset	Set connect = CreateObject("ADODB.Connection"). Set record = CreateObject("ADODB.Recordset").
3. Verificar si la conexión fue exitosa	Si connect.State = 1 entonces: Si la conexión es exitosa, asignar: SmartTags("Resultado") = "Conexión exitosa". Si la conexión falla: Asignar SmartTags("Resultado") = "Error al conectar a la base de datos". Liberar objeto de conexión: Set connect = Nothing. Finalizar la ejecución.
4. Obtener la lista de bases de datos disponibles	Asignar consulta a SQLDatabases: "SELECT name FROM sys.databases WHERE state_desc = 'ONLINE' AND name NOT IN ('master', 'tempdb', 'model', 'msdb');". Ejecutar consulta y guardar en record: Set record = connect.Execute(SQLDatabases).
5. Verificar si se obtuvieron bases de datos	Si record.EOF es True: Asignar SmartTags("Resultado") = "No se encontraron bases de datos." Liberar objeto de conexión: Set connect = Nothing. Finalizar la ejecución con Exit Sub.
6. Recorrer los resultados y construir la lista de bases de datos	Inicializar dbList como cadena vacía. Usar bucle Do While Not record.EOF: Asignar dbName = record.Fields("name").Value. Concatenar a dbList: dbList = dbList & dbName & " - ". Avanzar al siguiente registro: record.MoveNext.
7. Eliminar el último guion agregado	Si Len(dbList) > 2 entonces: dbList = Left (dbList, Len(dbList) - 2).
8. Mostrar la lista de bases de datos al usuario	Asignar resultado a SmartTags("Resultado") = dbList.
9. Desactivar la bandera de refresco	Asignar SmartTags("Refrescar") = False.
10. Finalización	Cerrar conexión y recordset: connect.Close y record.Close. Liberar objetos: Set connect = Nothing y Set record = Nothing.
11. Fin	

En el cuadro 10, se muestra la composición del script “TablaporDBRefresh”, describiendo su modo de operación para mostrar todas las tablas disponibles según el DB seleccionado. En el Anexo J se encuentran las líneas del código.

Cuadro 10: Pseudocódigo del script TablaporDBRefresh.

1. Inicio	
2. Inicializar conexión a la base de datos	Crear objetos de conexión y recordset: Set conn = CreateObject("ADODB.Connection"). Set record = CreateObject("ADODB.Recordset"). Intentar abrir conexión con: conn.Open "Provider=MSDASQL; DSN=" & DSN & ";". Si la conexión falla (conn.State <> 1): Asignar mensaje: SmartTags("Resultado2") = "Error al conectar a la base de datos." Finalizar ejecución con Exit Sub.
3. Cambiar el contexto a la base de datos especificada	Ejecutar comando SQL: conn.Execute "USE " & DBName.
4. Ejecutar consulta para obtener tablas	Asignar consulta SQL: SQLTablas = "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE';". Ejecutar la consulta: Set record = conn.Execute(SQLTablas).
5. Procesar resultado de la consulta	Si se encuentran tablas (Not record.EOF), recorrer los resultados: Concatenar los nombres de las tablas en tableNames, separados por -. Si no hay tablas: Asignar mensaje: tableNames = "No se encontraron tablas."
6. Asignar el resultado al HMI	SmartTags("Resultado2") = tableNames.
7. Finalización	Cerrar la conexión: conn.Close. Liberar objetos: Set conn = Nothing, Set record = Nothing. Actualizar variable de control: SmartTags("Refrescar2") = False.
8. Fin	

En el cuadro 11, se muestra la composición del script “*ColumnaRefreshDBdinamico*”, describiendo su modo de operación para mostrar todas las columnas disponibles según la tabla seleccionada. En el Anexo K se encuentran las líneas del código.

Cuadro 11: Pseudocódigo del script *ColumnRefreshDBdinamico*.

1. Inicio	
2. Obtener lista de bases de datos disponibles	Crear consulta: SQLDatabases = "SELECT name FROM sys.databases WHERE state_desc = 'ONLINE' AND name NOT IN ('master', 'tempdb', 'model', 'msdb');". Ejecutar la consulta: Set rsDatabases = conn.Execute(SQLDatabases).
3. Verificar si se obtuvieron bases de datos	Si conn.State = 0, entonces: SmartTags("Resultado3") = "Error al conectar a la base de datos.". Liberar conexión: Set conn = Nothing. Finalizar el script con Exit Sub.
4. Inicializar variables para columnas	columnList = "". foundTable = False.
5. Recorrer cada base de datos para buscar la tabla	Para cada base de datos en rsDatabases: Obtener nombre: dbName = rsDatabases.Fields("name").Value. Crear nueva conexión: Set conn = CreateObject("ADODB.Connection"). conn.Open "Provider=SQLOLEDB;Data Source=LAPTOP-SJQG2EJP; Initial Catalog=" & dbName & "; Integrated Security=SSPI; ". Crear consulta para columnas: sqlQuery = "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = " & tableName & "". Ejecutar consulta: Set rs = conn.Execute(sqlQuery).
6. Verificar si se encontraron columnas	Si Not rs.EOF, entonces: Recorrer resultados: Para cada columna: columnList = columnList & rs.Fields("COLUMN_NAME").Value & " - " rs.MoveNext. Establecer foundTable = True. Salir del bucle con Exit Do. Si no se encontraron columnas, continuar con la siguiente base de datos.
7. Evaluar si se encontró la tabla en alguna base de datos	Si foundTable = False, entonces: columnList = "No se encontraron columnas para la tabla " & tableName & " en ninguna base de datos.". Si se encontraron columnas: Eliminar el último guion y espacio: columnList = Left(columnList, Len(columnList) - 3).
8. Asignar resultado	SmartTags("Resultado3") = columnList.
9. Fin	

IV.4.2.2. Implementación del sistema remoto

Red PROFINET

Se utilizó la misma red PROFINET descrita en el sistema local (véase la Figura 43), donde el PLC S7-1500 se comunicó con la laptop mediante este protocolo. En este caso, esta conexión formó parte del Plant Bus, uno de los niveles de comunicación establecidos en una red industrial. Esto es posible porque en la laptop se ejecutó el entorno TIA Portal, que se enlazó con Node – RED para transmitir las variables necesarias para el cálculo de los KPIs.

Enlace de TIA PORTAL y Node-RED

En TIA Portal se creó el bloque de datos (DB) (véase la Figura 61) Var_KPI [DB30] en donde se tuvieron las variables necesarias para calcular los KPIs .

	Name	Data type	Offset	Start value	Retain
1	Static				☐
2	Botellas	Int	0.0	0	☐
3	ProductoBueno	Int	2.0	0	☐
4	Merma	Int	4.0	0	☐
5	Minutos	Real	6.0	0.0	☐

Figura 61. Bloque de datos (DB) creado en TIA Portal.

Se creó un contador para calcular las botellas procesadas de manera general ya sea en buenas o malas condiciones. Como se observa en la Figura 62, la cantidad total de botellas fue recepcionada por la variable Var_KPI.Botellas, que se encuentra dentro del DB30.

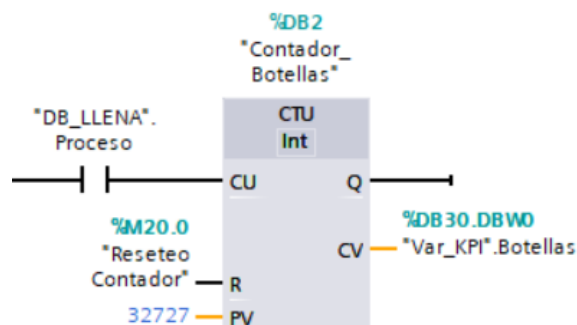


Figura 62. Contador de total de botellas.

Al total de botellas se le restó la merma para obtener el número de botellas en buen estado, valor que fue recepcionado por Var_KPI.ProductoBueno (véase la Figura 63).

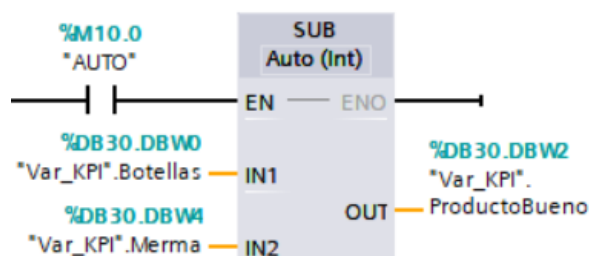


Figura 63. Total de botellas en buen estado.

Para calcular los minutos transcurridos durante la producción fue necesario crear un bloque de función (véase la Figura 64), en el cual se inició con el cálculo de segundos a través de un temporizador con retardo a la conexión (TON) (véase la Figura 65). Cada pulso que emitió fue contabilizado por el contador ascendente (CTU) que, al tener un total de 60 pulsos, envió un pulso al CTU del segmento dos (véase la Figura 66), encargado de contabilizar los minutos.

Si bien los contadores trabajan con salidas de tipo entero o int, fue necesario convertirlos a reales por medio de la instrucción CONV para realizar el cálculo del total de minutos mediante la instrucción CALCULATE (véase la Figura 67), el cual sólo

recibió sin conflictos variables del tipo real. Dicho cálculo se realizó para obtener los minutos de manera más precisa con decimales.



Figura 64. FB Minutos.

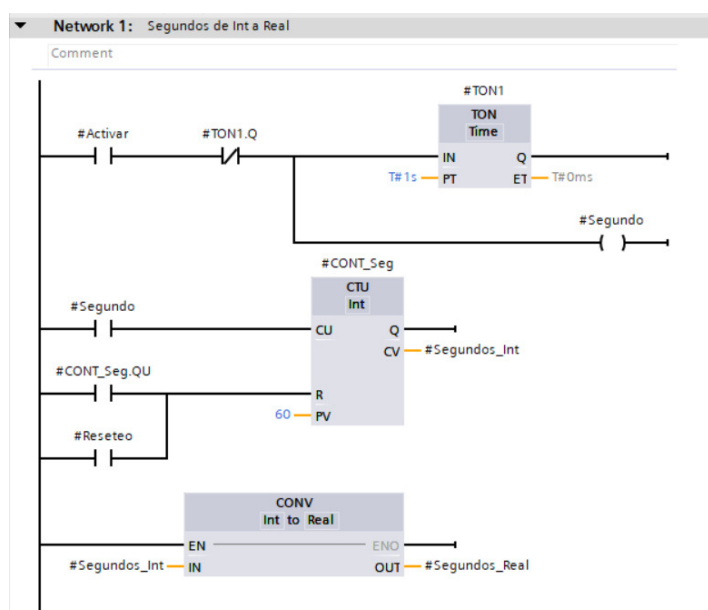


Figura 65. Segmento 1 del FB Minutos.

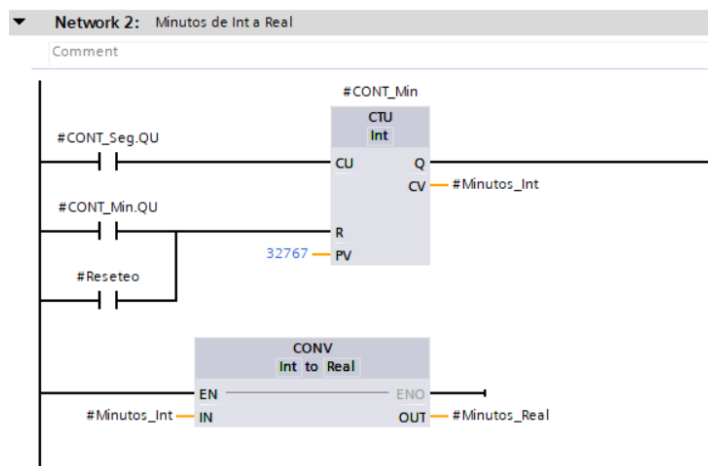


Figura 66. Segmento 2 del FB Minutos.

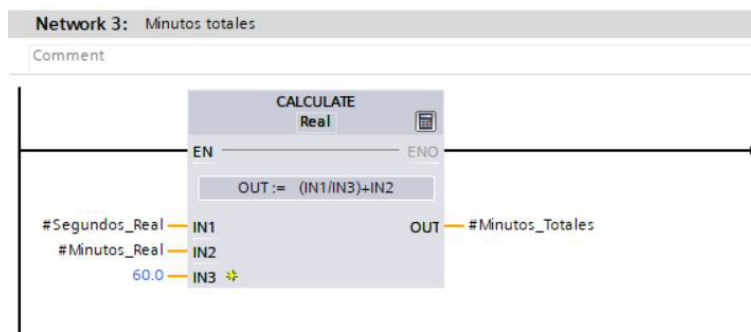


Figura 67. Segmento 3 del FB Minutos.

Se inició con el envío de las variables que contenían la cantidad de botellas producidas, las botellas que cumplían con los estándares deseados, las botellas clasificadas como merma, así como la variable de tiempo transcurrido. Este envío se realizó desde TIA Portal a través de Node-RED, con el objetivo de que InfluxDB Cloud pudiera almacenar estos datos.

Node-RED utilizó el nodo S7 para establecer la comunicación con el PLC. En dicho nodo se configuró la dirección IP del PLC (véanse las Figuras 68 y 69), las variables enviadas a Node-RED junto con sus direcciones (véase la Figura 70), de acuerdo con la estructura del bloque de datos (DB) creado en TIA Portal (véase la Figura 61), además del intervalo de tiempo para la lectura periódica.

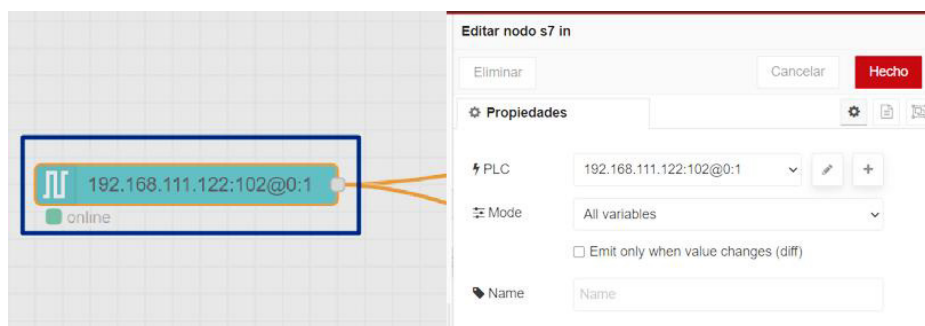


Figura 68. Configuración del nodo S7 – Parte 1.

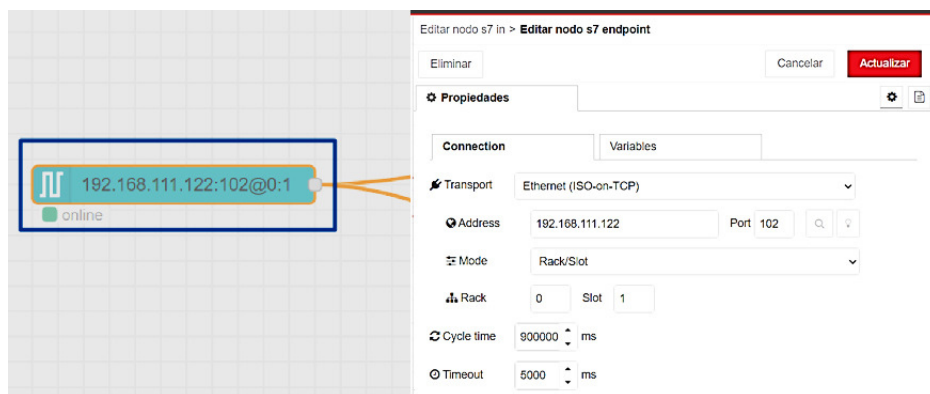


Figura 69. Configuración del nodo S7 – Parte 2.

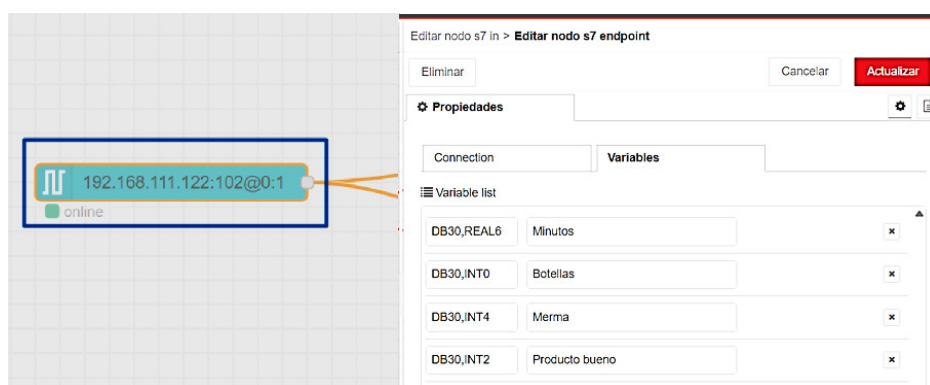
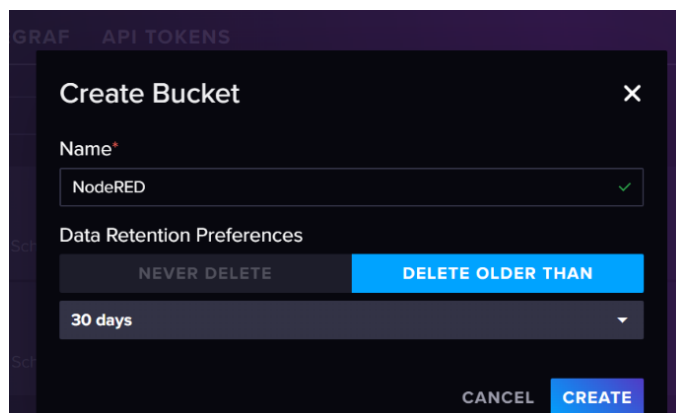


Figura 70. Configuración del nodo S7 – Parte 3.

Enlace de Node-RED y InfluxDB Cloud

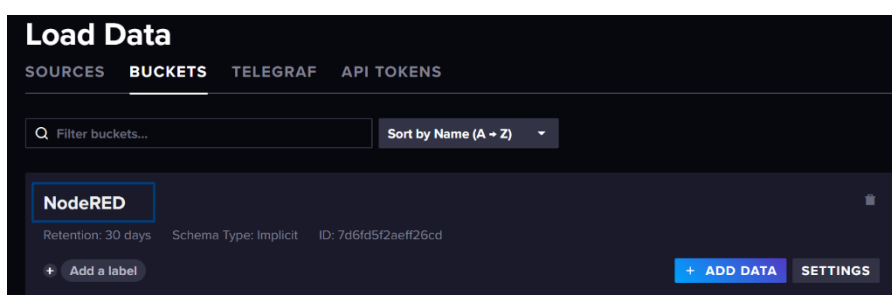
Se utilizó el nodo Influxdb Out para enviar y registrar los datos en una base de datos de series temporales alojada en InfluxDB Cloud. Para la configuración de este nodo (véanse las Figuras 75 y 76), fue necesario especificar el nombre del bucket previamente creado en InfluxDB Cloud (véanse las Figuras 71 y 72), la URL del servicio, el API Token (véanse las Figuras 73 y 74), que permitió la autenticación y conexión segura, así como el nombre de la organización correspondiente.

El API Token es importante para autenticar las solicitudes realizadas a la API. Por otro lado, el bucket es la ubicación donde se almacenaron los datos.



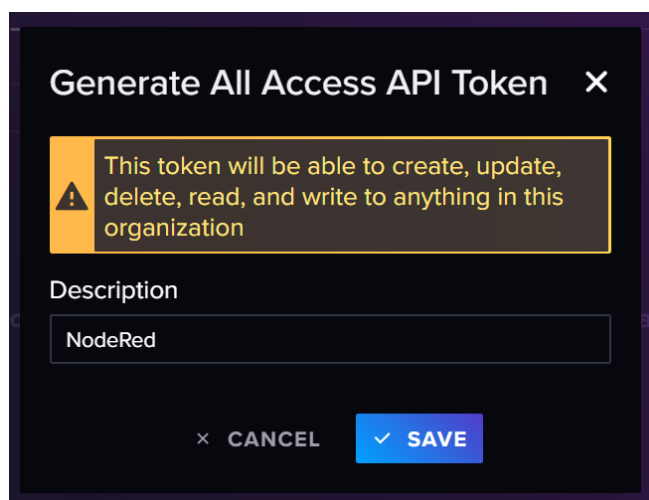
The screenshot shows a 'Create Bucket' modal window. At the top, there's a title bar with 'GRAF' and 'API TOKENS' on the left, and a close button 'X' on the right. The main title is 'Create Bucket'. Below it, there's a 'Name' field with 'NodeRED' entered and a green checkmark icon. Underneath is the 'Data Retention Preferences' section, which has two buttons: 'NEVER DELETE' and 'DELETE OLDER THAN' (which is highlighted in blue). Below these buttons is a dropdown menu showing '30 days'. At the bottom right, there are two buttons: 'CANCEL' and 'CREATE' (highlighted in blue).

Figura 71. Creación del bucket en InfluxDB Cloud.



The screenshot shows the 'Load Data' page with tabs for 'SOURCES', 'BUCKETS', 'TELEGRAP', and 'API TOKENS'. The 'BUCKETS' tab is active. There's a search bar 'Filter buckets...' and a 'Sort by Name (A - Z)' dropdown. A list of buckets is shown, with 'NodeRED' selected and highlighted with a blue border. Below the bucket name, it shows 'Retention: 30 days', 'Schema Type: Implicit', and 'ID: 7d6fd5f2aef26cd'. At the bottom, there are three buttons: '+ Add a label', '+ ADD DATA' (highlighted in blue), and 'SETTINGS'.

Figura 72. Bucket disponible en InfluxDB Cloud.



The screenshot shows a 'Generate All Access API Token' modal window. It has a title bar with 'Generate All Access API Token' and a close button 'X'. Below the title, there's a warning box with a yellow background and a black border, containing a warning icon and the text: 'This token will be able to create, update, delete, read, and write to anything in this organization'. Underneath is a 'Description' field with 'NodeRed' entered. At the bottom, there are two buttons: 'CANCEL' and 'SAVE' (highlighted in blue).

Figura 73. Generación del API Token en InfluxDB Cloud.

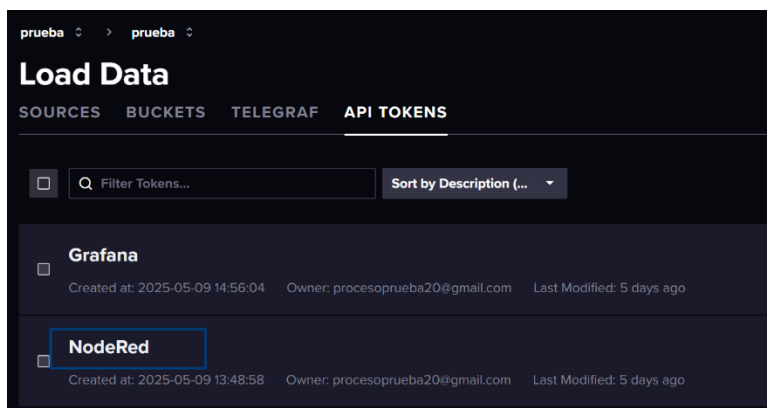


Figura 74. API Token en InfluxDB Cloud.

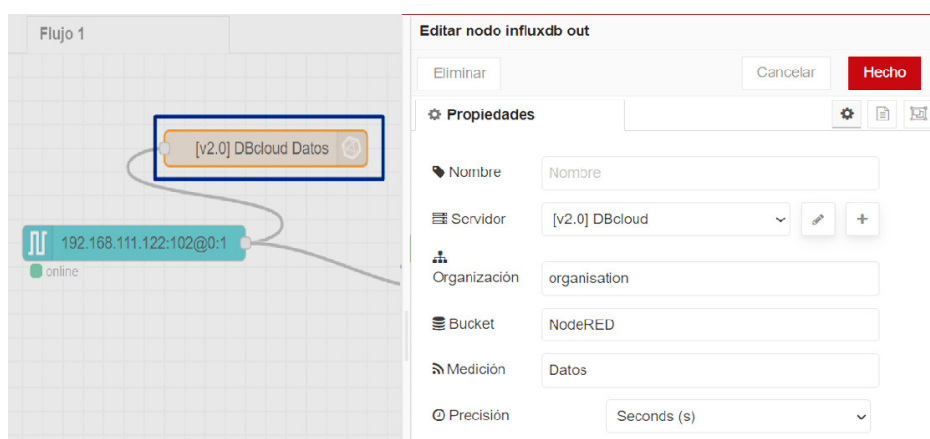


Figura 75. Configuración del nodo Influxdb out– Parte 1.

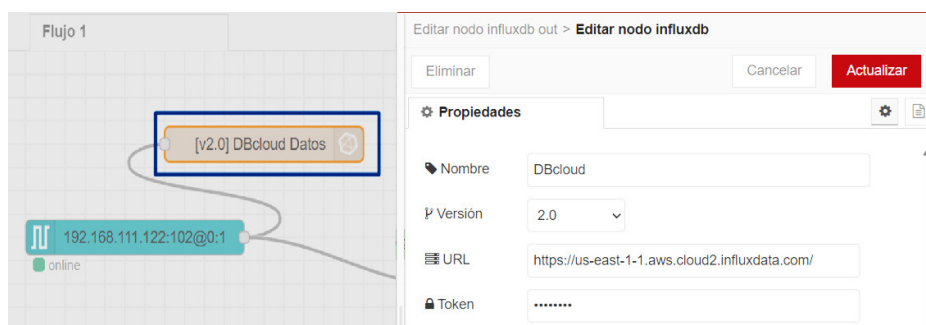


Figura 76. Configuración del nodo Influxdb out– Parte 2.

Enlace de InfluxDB Cloud y Grafana Cloud

Los datos almacenados en InfluxDB Cloud se transmitieron a Grafana Cloud para calcular y visualizar los KPIs mediante gráficos dinámicos e interactivos. Previamente, es necesario generar un API Token en InfluxDB Cloud (véase la Figura 77). Luego, en Grafana Cloud se configuró la fuente de datos seleccionando InfluxDB Cloud (véase la Figura 78), en donde se eligió el lenguaje Flux (véase la Figura 79) para realizar consultas, compatible con la versión del InfluxDB configurado. Durante las configuraciones para enlazar Grafana Cloud e InfluxDB Cloud se proporcionaron la URL del servicio, el API Token, el nombre de la organización y el bucket correspondiente (véase la Figura 80).

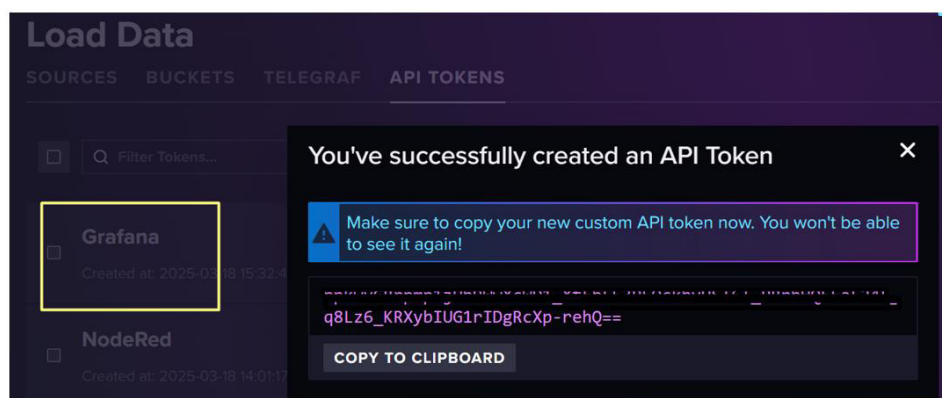


Figura 77. Generación de un segundo API Token.

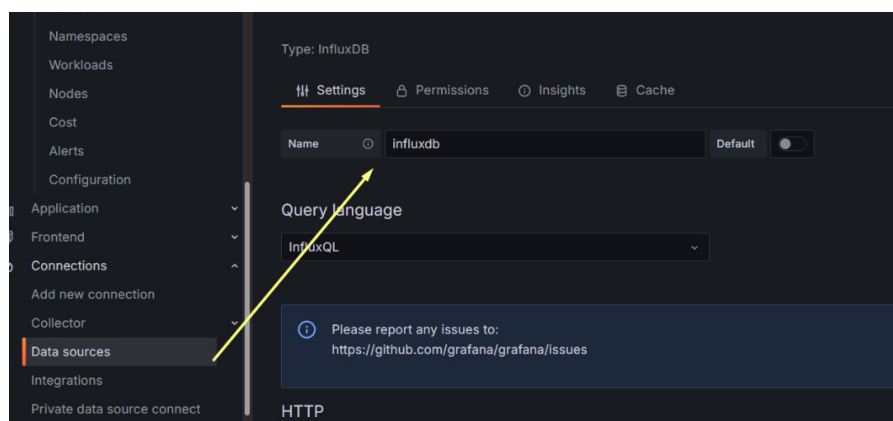


Figura 78. Configuración de fuente de datos en Grafana Cloud – Parte 1.

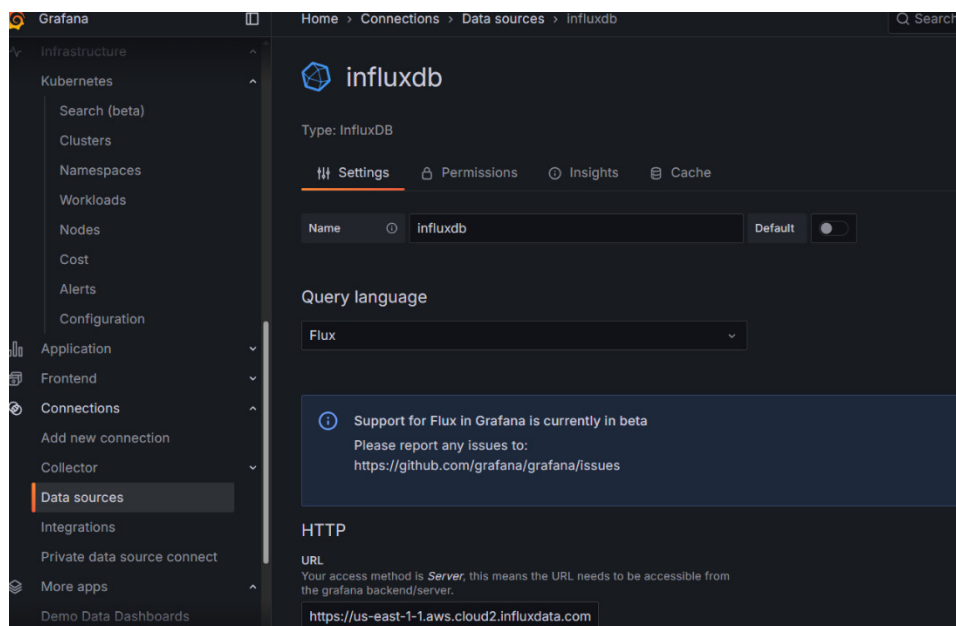


Figura 79. Configuración de fuente de datos en Grafana Cloud – Parte 2.

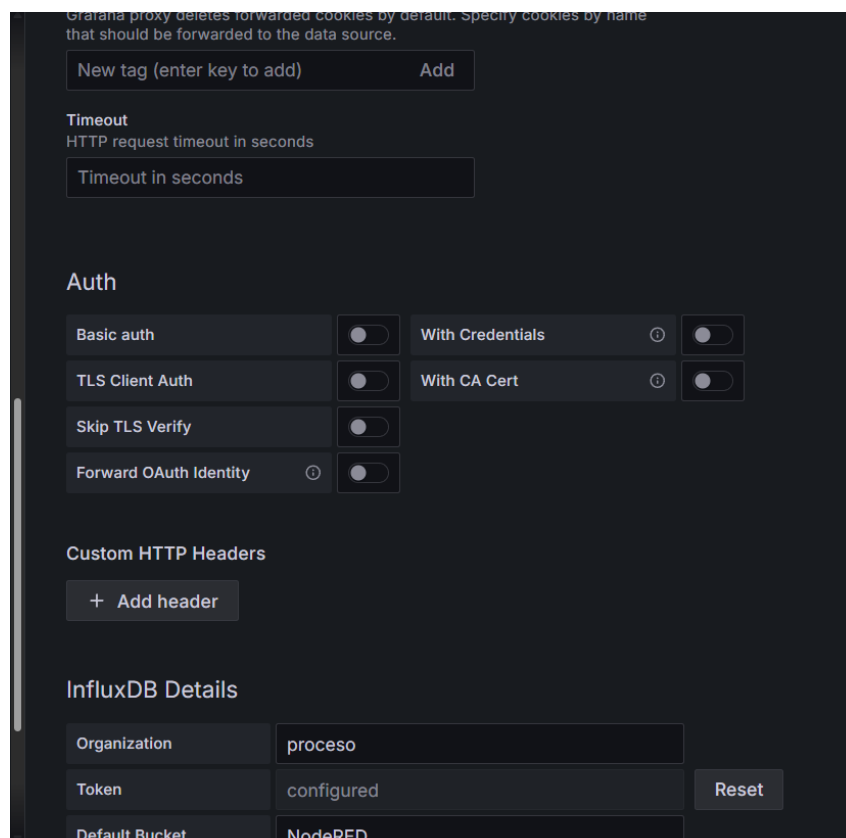


Figura 80. Configuración de fuente de datos en Grafana Cloud – Parte 3.

Scripts para calcular los KPIs

Después de recepcionar los datos de InfluxDB Cloud, se procedió a elaborar scripts en lenguaje Flux que fueron capaces de calcular los KPIs en base a estos datos. A continuación, se explicará el funcionamiento de cada script.

Script para el cálculo de la tasa de utilización de capacidad

```
1 from(bucket: "NodeRED")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r["_measurement"] == "Datos")
4   |> filter(fn: (r) => r["_field"] == "Botellas")
5   |> sort(columns: ["_time"]) // Ordenar por tiempo
6   |> difference(columns: ["_value"], keepFirst: true)
7   |> map(fn: (r) => ({
8     _time: r._time,
9     "Tasa de utilización de capacidad": (r._value / 450.0) * 100.0
10  })))
```

Figura 81. Script para el cálculo de la tasa de utilización de capacidad.

Se comenzó extrayendo los registros del bucket llamado “NodeRED” en InfluxDB Cloud, dentro del intervalo definido por v.timeRangeStart y v.timeRangeStop en Grafana Cloud. Luego, se filtraron únicamente los datos del campo “Botellas” en la serie “Datos” (véase la Figura 81) y se ordenaron cronológicamente. A continuación, se calculó la diferencia entre cada lectura y la anterior para obtener la producción de botellas en cada intervalo de tiempo, cuyo máximo podía ser de 15 minutos, y dicha diferencia se dividió entre la capacidad máxima teórica, que es de 450 botellas, multiplicando el resultado por 100 para obtener un valor porcentual.

Script para el cálculo de la tasa de calidad

```
1 botellas = from(bucket: "NodeRED")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r["_measurement"] == "Datos" and r["_field"] == "Botellas")
4 merma = from(bucket: "NodeRED")
5   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
6   |> filter(fn: (r) => r["_measurement"] == "Datos" and r["_field"] == "Merma")
7 join(
8   tables: {botellas: botellas, merma: merma},
9   on: ["_time"],
10  method: "inner")
```

Figura 82. Script para el cálculo de la tasa de calidad – Parte 1.

```

10   method: "inner"
11   )
12   |> map(fn: (r) => ({
13     _time: r._time,
14     Tasa_de_calidad: (r._value_botellas - r._value_merma) / r._value_botellas * 100.0,
15   }))
16   |> rename(columns: {
17     "Tasa_de_calidad": "Tasa de calidad",
18   })
19   |> yield(name: "Tasa de calidad")

```

Figura 83. Script para el cálculo de la tasa de calidad – Parte 2.

El proceso inició extrayendo los datos de “Botellas” y “Merma” del bucket “NodeRED” (véase la Figura 82) dentro de un intervalo definido por Grafana Cloud. Luego, se utilizó la función join para unir ambos flujos por su misma marca temporal, de modo que cada punto contuviera tanto el valor de botellas como el de merma. La función map calculó la tasa de calidad restando la merma del total de botellas, dividiendo ese resultado entre el total de botellas y multiplicándolo por 100 (véase la Figura 83) para obtener un valor porcentual.

Script para el cálculo de la tasa de disponibilidad

```

1   from(bucket: "NodeRED")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r["_measurement"] == "Datos")
4   |> filter(fn: (r) => r["_field"] == "Minutos")
5   |> sort(columns: ["_time"]) // Asegúrate de que los datos estén ordenados por tiempo
6   |> difference(columns: ["_value"], keepFirst: true)
7   |> map(fn: (r) => ({
8     _time: r._time,
9     tasa_disponibilidad: (
10      (if r._value + 0.1 > 15.0 then 15.0 else r._value + 0.1) / 15.0) * 100.0
11    ))
12   |> keep(columns: ["_time", "tasa_disponibilidad"])
13   |> yield(name: "Tasa de disponibilidad")
14

```

Figura 84. Script para el cálculo de la tasa de disponibilidad.

Este script en Flux trabajó con la medición “Datos” y el campo “Minutos” (véase la Figura 84). Primero, ordenó los registros por su marca de tiempo y calculó la diferencia entre cada valor y el anterior. Después, se añadió 0.1 unidades a la diferencia resultante para considerar un tiempo de retardo en el envío de datos entre plataformas, limitando el resultado a un máximo de 15 minutos. Con este valor corregido, se calculó la tasa de disponibilidad, expresada como el porcentaje respecto a los 15 minutos, que

representan el total del intervalo. Cabe destacar que esta consideración de margen de retardo se aplicó en los diferentes scripts según sea necesario.

Script para el cálculo de la productividad

```

1 producto_bueno = from(bucket: "NodeRED")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r._measurement == "Datos" and r._field == "Producto bueno")
4   |> sort(columns: ["_time"])
5   |> difference(nonNegative: true)
6   |> rename(columns: {_value: "producto_bueno_diff"})
7 minutos = from(bucket: "NodeRED")
8   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
9   |> filter(fn: (r) => r._measurement == "Datos" and r._field == "Minutos")
10  |> sort(columns: ["_time"])
11  |> difference()
12  |> map(fn: (r) => ({
13    _time: r._time,
14    minutos_diff_ajustado: if (r._value + 0.1) > 15.0 then 15.0 else r._value + 0.1
15  }))
16 join(
17   tables: {producto: producto_bueno, tiempo: minutos},
18   on: ["_time"],
19 )

```

Figura 85. Script para el cálculo de la productividad – Parte 1.

```

20   |> map(fn: (r) => ({
21     _time: r._time,
22     _value: r.producto_bueno_diff / r.minutos_diff_ajustado
23   }))
24   |> set(key: "_field", value: "Productividad")
25   |> yield(name: "Productividad")
26

```

Figura 86. Script para el cálculo de la productividad – Parte 2.

A partir de dos series de datos almacenadas en InfluxDB Cloud, una con el campo “Producto bueno” y otra con el campo “Minutos” (véase la Figura 85), se procedió a extraer sus valores según el intervalo seleccionado en Grafana Cloud. Luego, se calculó la diferencia entre los valores consecutivos para obtener la cantidad de productos buenos y minutos generados en cada intervalo. En el caso de los minutos, se le sumó un margen de 0.1 (véase la Figura 85) y se limitó a un valor máximo de 15 minutos. Posteriormente, ambas series se combinaron usando join, que emparejó los datos que compartieron la misma marca temporal. Finalmente, se dividió el número de productos buenos entre los minutos ajustados para obtener la productividad (véase la Figura 86).

Finalmente, el desarrollo implicó articular la gestión de datos en una doble estrategia de almacenamiento y visualización. Para las variables dinámicas de los servomotores, los datos recolectados fueron almacenados directamente en las bases SQL Server, lo cual sustentó su visualización desde la interfaz local de WinCC Professional. En paralelo, para la gestión remota, las variables esenciales para el cálculo de los indicadores clave de rendimiento (KPIs) se registraron en InfluxDB Cloud. Este proceso se complementó con la ejecución de scripts específicos directamente en Grafana Cloud, que no solo permitieron el cálculo de los KPIs, sino también su visualización inmediata en el dashboard de la plataforma. Este procesamiento integral constituyó la fase final del estudio, cuyo propósito fue obtener y presentar los valores representativos del proceso. Los resultados, mostrados en las interfaces local y remota, demostraron de forma concluyente la capacidad operativa del sistema para gestionar y exhibir eficazmente todos los valores e indicadores seleccionados.

V. RESULTADOS

V.1. Presentación y análisis de los resultados

V.1.1. Sistema local

Se inició creando la base de datos y su respectiva tabla que recibirían los datos a almacenar en SQL Server. Se indicó el nombre de la base de datos (DB), ya sea para crearla o para crear una tabla dentro de una base de datos existente. En caso de que se decidiera crear una base de datos y se cumpliera la condición de que no existiera otra con el mismo nombre, se procedió a crearla y se mostró un mensaje indicando que fue creada exitosamente.

En este caso, se creó un base de datos llamada “FAJA” (véase la Figura 87), que contenía la tabla “SERVO1” (véanse las Figuras 88 y 89) con tres columnas: Velocidad Faja (mm/s), Torque Faja (Nm) y Posición Faja (mm).

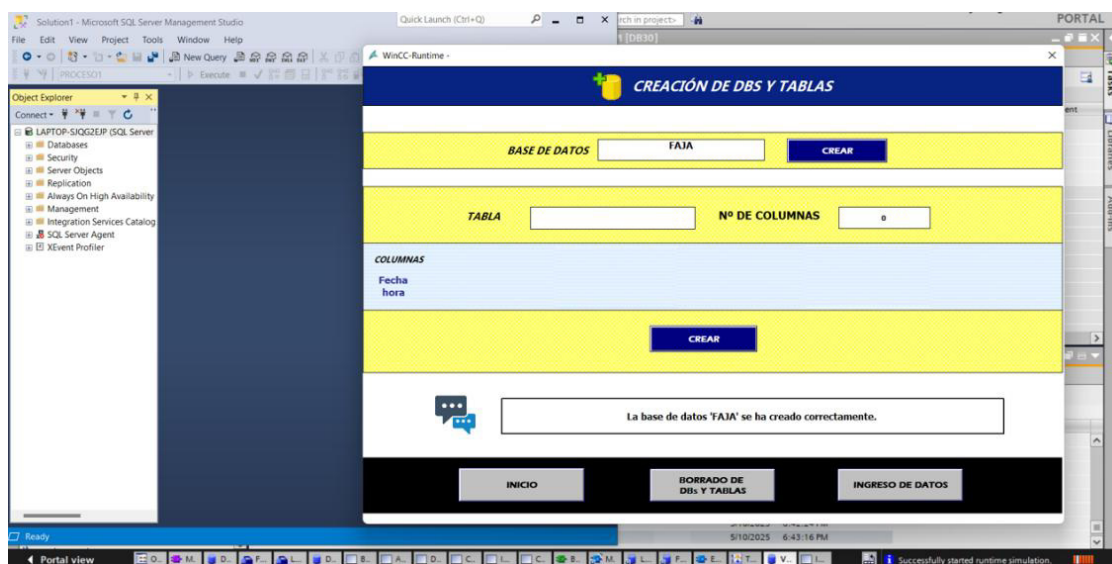


Figura 87. Creación de la base de datos FAJA.

Este sistema permitió contar con hasta 5 columnas y decidir de manera práctica qué dato se podía almacenar.

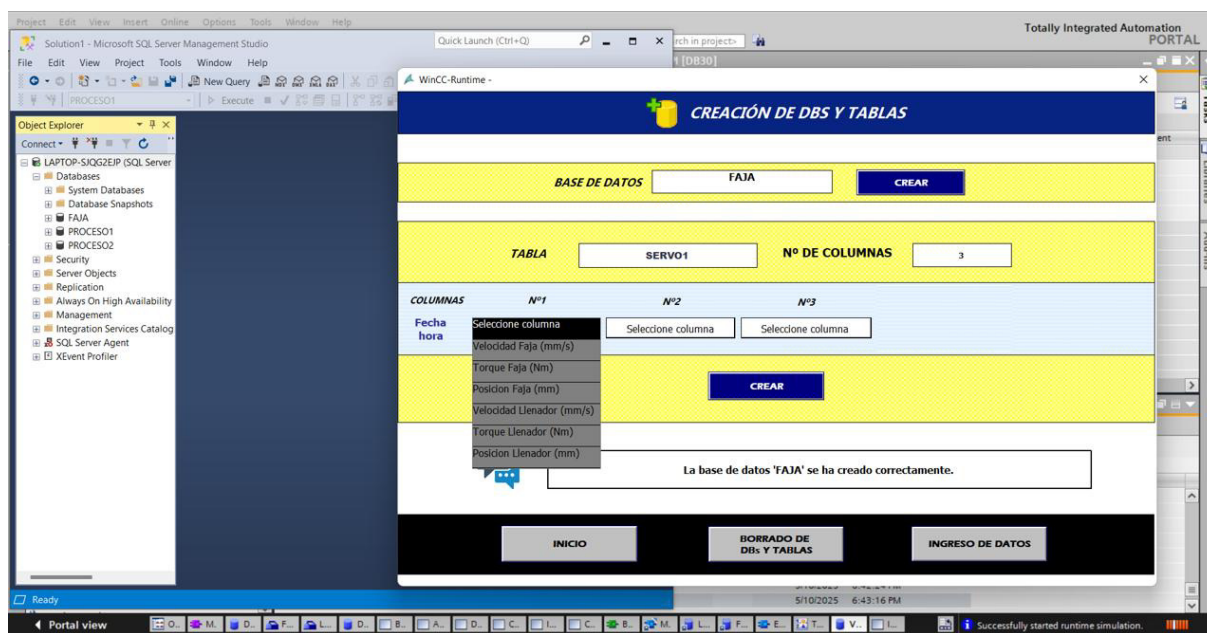


Figura 88. Selección de columnas para la tabla SERVO1 perteneciente a la base de datos FAJA.

La creación de la tabla fue confirmada mediante un mensaje de validación; en caso de que el proceso no se completara exitosamente, el sistema emitió una notificación detallando las incidencias registradas.

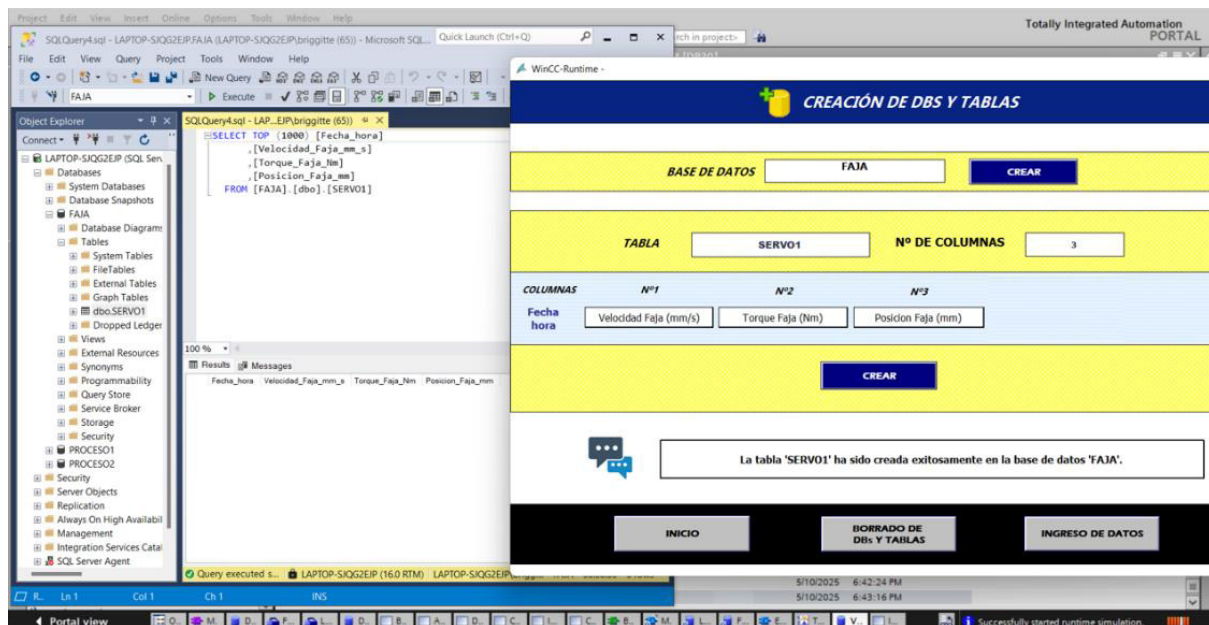


Figura 89. Creación de la tabla *SERVO1* perteneciente a la base de datos *FAJA*.

De la misma manera, se creó la base de datos y tabla que almacenaron las variables dinámicas del servomotor del dispensador, también denominado llenador (véanse las Figuras 90, 91 y 92).

Luego, se procedió a crear la base de datos o DB “LLENADOR” (véase la Figura 90).

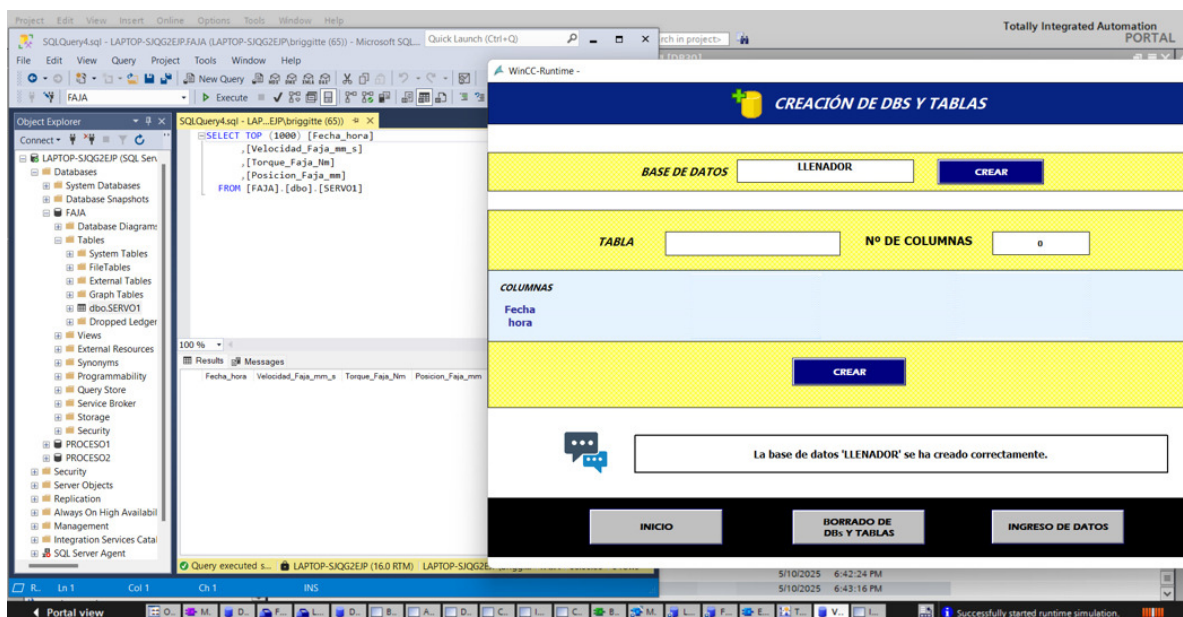


Figura 90. Creación de la base de datos *LLENADOR*.

Dentro de esta DB se creó la tabla “SERVO2”, a la cual también se le asignaron tres columnas (véase la Figura 91). Finalmente, se mostró un mensaje que confirmó la creación exitosa de la tabla (véase la Figura 92).

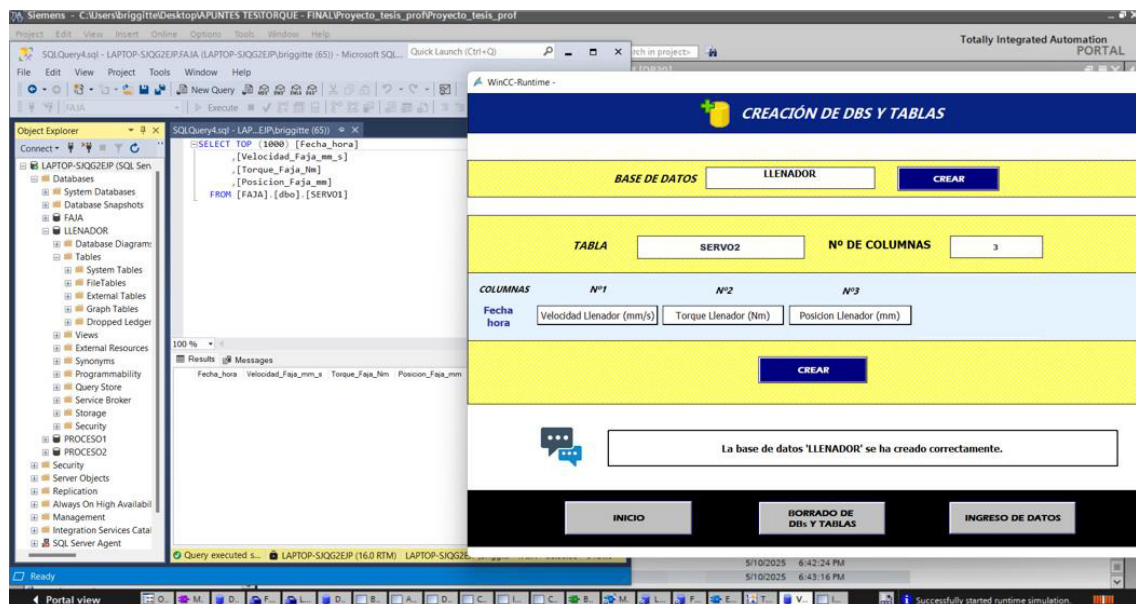


Figura 91. Selección de columnas para la tabla *SERVO2* perteneciente a la base de datos *LLENADOR*.

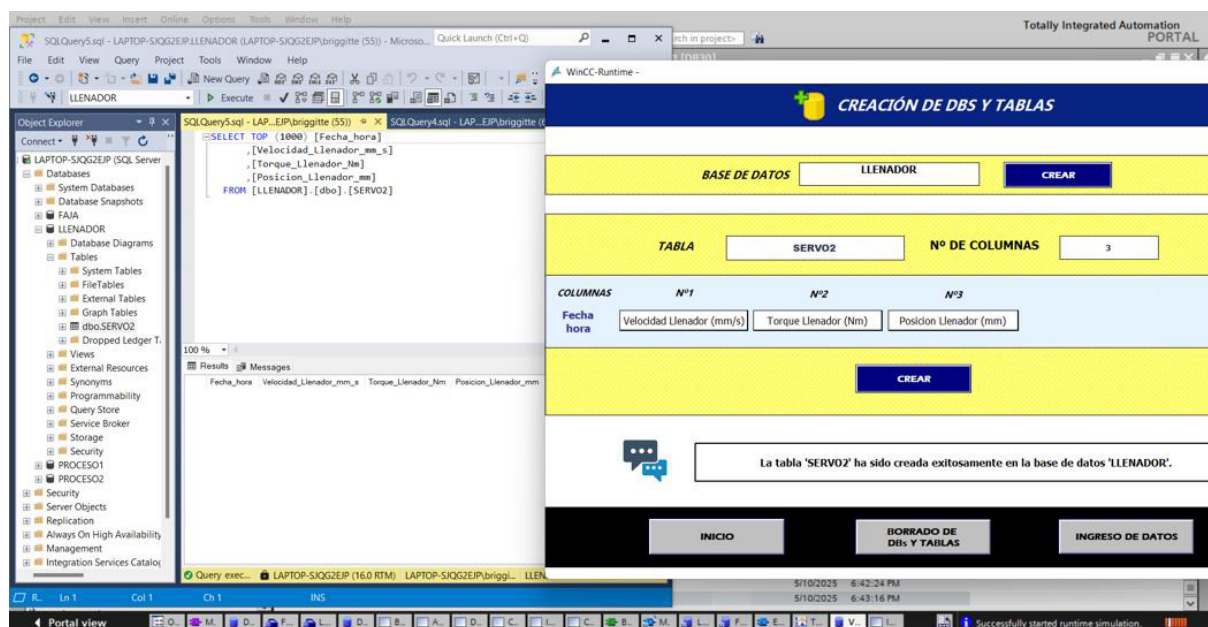


Figura 92. Creación de la tabla *SERVO2* perteneciente a la base de datos *LLENADOR*.

A continuación, se ejecutó la puesta en marcha de la demo, tal como se muestra en la Figura 93.

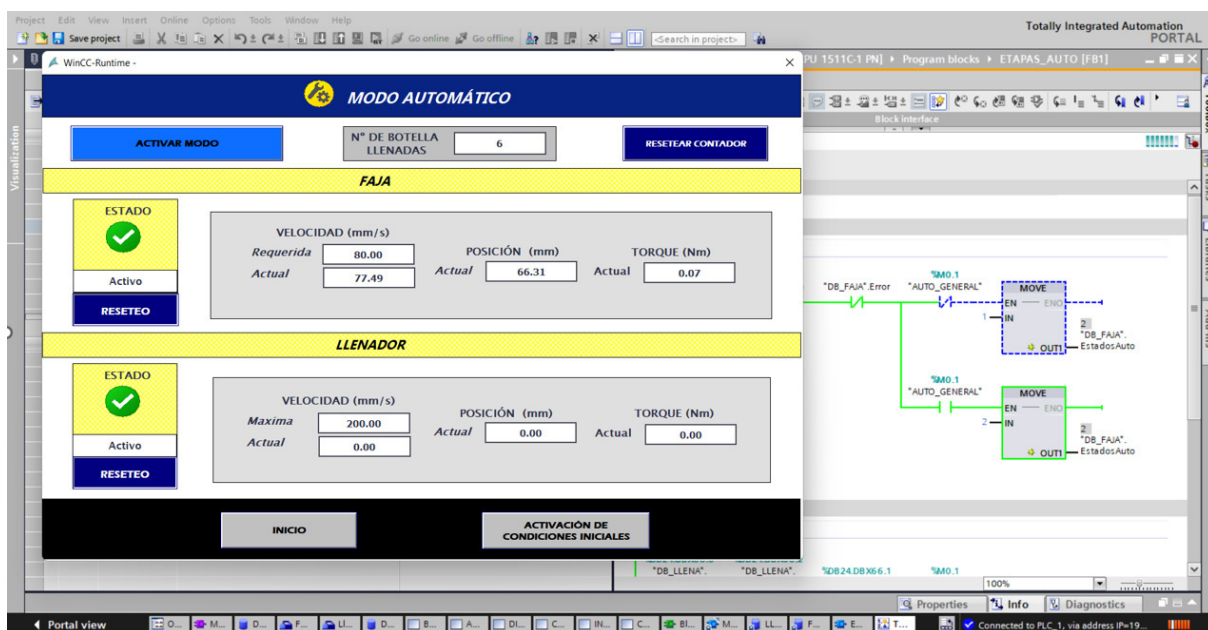


Figura 93. Demo en funcionamiento.

Se procedió a realizar el almacenamiento de variables, ya fuera en modo manual o automático. En el modo manual (véase la Figura 94), con un solo pulso del operador se almacenó un grupo de datos, mientras que en el modo automático (véase la Figura 95) un pulso activó el almacenamiento y otro lo desactivó, permitiendo así un registro continuo y adaptable según la necesidad. Como se muestra en la Figura 94, con un solo pulso únicamente se logró almacenar un dato por columna.

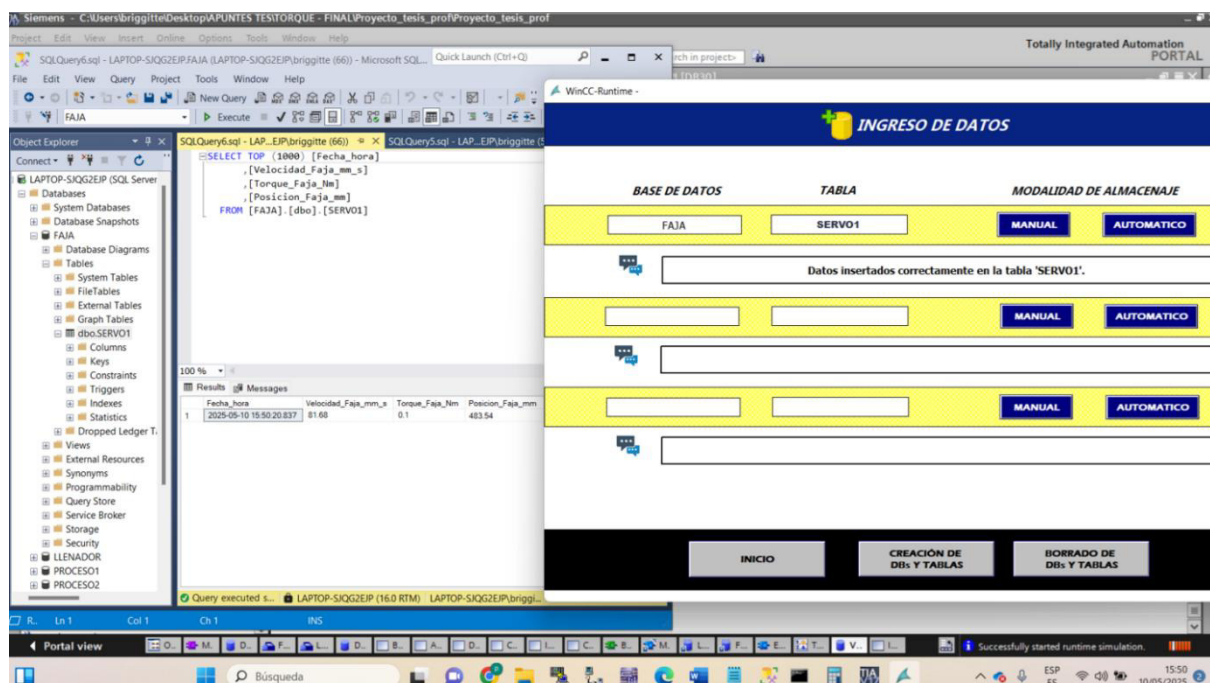


Figura 94. Almacenamiento de datos en modo manual.

En el modo automático, mostrado en la Figura 95, los datos se almacenaron de forma continua en intervalos regulares. Se observó además que, a medida que se realizaba el almacenamiento, el botón cambiaba de color azul a verde y aparecía un mensaje de confirmación.

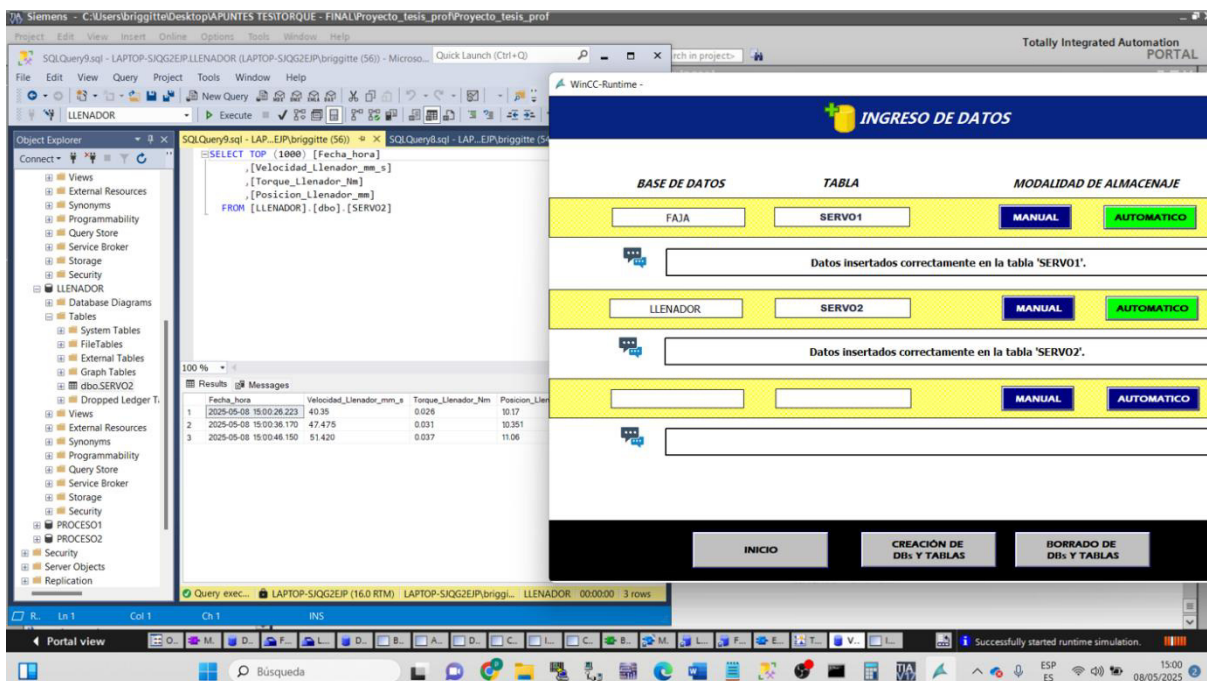


Figura 95. Almacenamiento de datos en modo automático.

A través de la pantalla desarrollada en WinCC Professional, se pudieron visualizar las variables dinámicas de los servomotores. Se compararon los parámetros mostrados en pantalla con los de los objetos tecnológicos (véanse las Figuras 96, 98, 100, 102, 104 y 106) y con los datos almacenados (véanse las Figuras 97, 99, 101, 103, 105 y 107) con el fin de confirmar que la información es veraz y confiable.

En la Figura 96 se observa que los valores coincidían tanto en la base de datos como en la interfaz de parámetros de objetos tecnológicos. Esta misma correspondencia se aprecia en las Figuras 98 y 100, en el caso de las variables dinámicas del servomotor encargado de la faja.

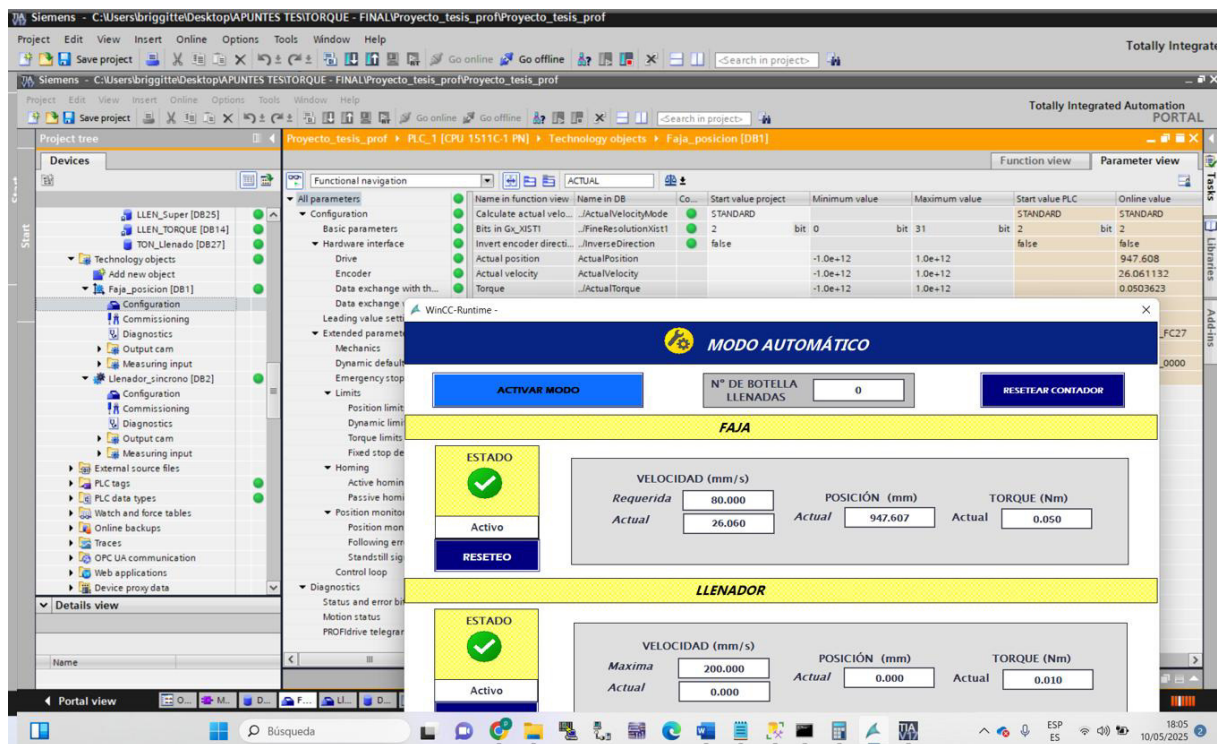


Figura 96. Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 1.

SQLQuery17.sql - LAPTOP-SJQG2EJP.master (LAPTOP-SJQG2EJP\brigitte (57)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

master Execute

Object Explorer

Connect LAPTOP-SJQG2EJP (SQL Server 16.0.10)

- Databases
 - System Databases
 - Database Snapshots
 - FAJA
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERV01
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
 - LLENADOR
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERV02
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store

SQLQuery17.sql - LA...EJP\brigitte (57)

```
SELECT [Fecha_hora]
,[Velocidad_Faja_mm_s]
,[Torque_Faja_Nm]
,[Posicion_Faja_mm]
FROM [FAJA].[dbo].[SERV01]]
ORDER BY [Fecha_hora] ASC
```

100 %

Results Messages

	Fecha_hora	Velocidad_Faja_mm_s	Torque_Faja_Nm	Posicion_Faja_mm
124	2025-05-10 18:05:03.203	23.127	0.043	635.514
125	2025-05-10 18:05:03.337	23.571	0.044	659.595
126	2025-05-10 18:05:03.597	23.737	0.045	683.547
127	2025-05-10 18:05:04.053	23.966	0.045	707.557
128	2025-05-10 18:05:04.200	24.177	0.045	731.577
129	2025-05-10 18:05:04.407	24.383	0.046	755.593
130	2025-05-10 18:05:04.660	24.592	0.046	779.565
131	2025-05-10 18:05:04.913	24.817	0.047	803.337
132	2025-05-10 18:05:05.170	25.012	0.048	827.122
133	2025-05-10 18:05:05.437	25.222	0.048	851.643
134	2025-05-10 18:05:05.673	25.433	0.048	875.524
135	2025-05-10 18:05:05.913	25.641	0.049	899.781
136	2025-05-10 18:05:06.167	25.851	0.049	923.476
137	2025-05-10 18:05:06.407	26.06	0.05	947.607
138	2025-05-10 18:05:06.657	26.245	0.051	971.637
139	2025-05-10 18:05:06.910	26.479	0.051	995.605
140	2025-05-10 18:05:07.153	26.611	0.051	19.613
141	2025-05-10 18:05:07.407	26.853	0.052	43.611
142	2025-05-10 18:05:07.667	27.117	0.053	67.612
143	2025-05-10 18:05:07.917	27.317	0.053	91.613
144	2025-05-10 18:05:08.163	27.523	0.054	115.614
145	2025-05-10 18:05:08.420	27.736	0.054	139.615
146	2025-05-10 18:05:08.660	27.945	0.055	166.725
147	2025-05-10 18:05:08.910	28.152	0.055	187.612
148	2025-05-10 18:05:09.160	28.364	0.056	211.615
149	2025-05-10 18:05:09.410	28.574	0.056	235.619
150	2025-05-10 18:05:09.663	28.784	0.057	259.62
151	2025-05-10 18:05:09.917	28.958	0.058	283.425

Query executed successfully.

Ready

Figura 97. Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 1.

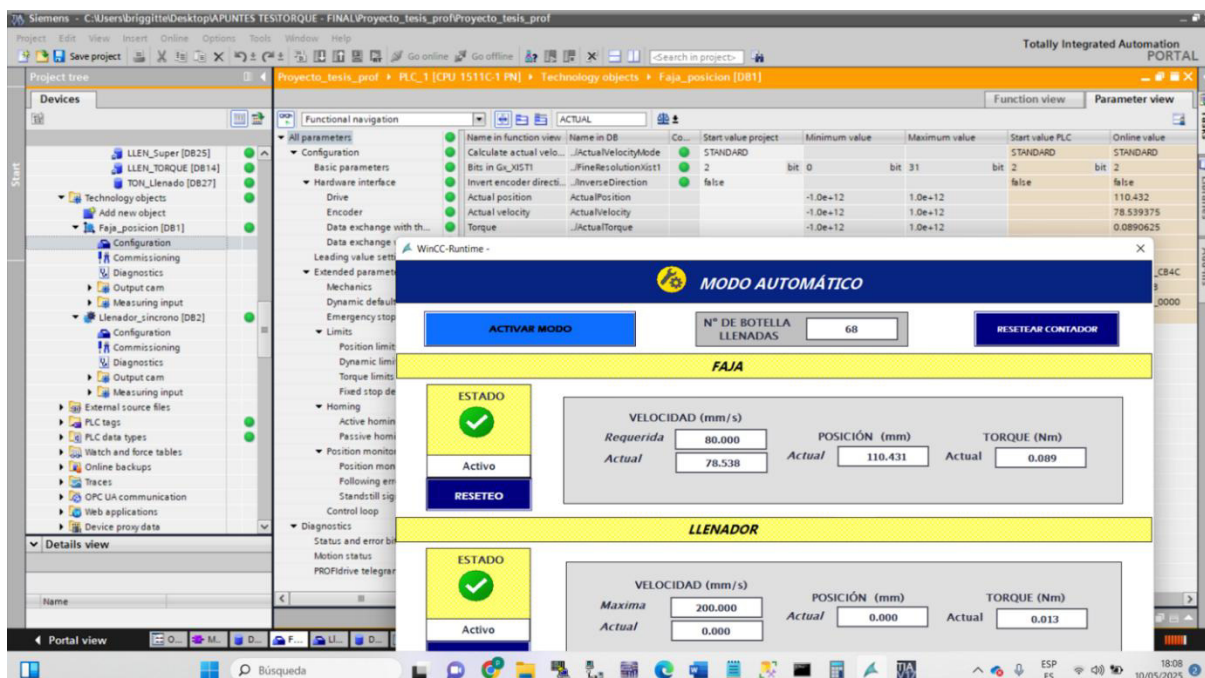


Figura 98. Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 2.

SQLQuery18.sql - LAPTOP-SJQG2EJP.FAJA (LAPTOP-SJQG2EJP\briggitte (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

FAJA Execute

Object Explorer

Connect LAPTOP-SJQG2EJP (SQL Server 16.0.10)

Databases

- System Databases
- Database Snapshots
- FAJA
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERVO1
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
- LLENADOR
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERVO2
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store

SQLQuery18.sql - LA...EJP\briggitte (58)

```
SELECT [Fecha_hora]
      ,[Velocidad_Faja_mm_s]
      ,[Torque_Faja_Nm]
      ,[Posicion_Faja_mm]
FROM [FAJA].[dbo].[SERVO1]
ORDER BY [Fecha_hora] ASC
```

100 %

Results Messages

	Fecha_hora	Velocidad_Faja_mm_s	Torque_Faja_Nm	Posicion_Faja_mm
711	2025-05-10 18:08:41.937	77.492	0.072	915.481
712	2025-05-10 18:08:42.150	77.627	0.073	939.484
713	2025-05-10 18:08:42.407	77.523	0.073	963.482
714	2025-05-10 18:08:42.653	77.887	0.074	987.478
715	2025-05-10 18:08:42.897	78.015	0.077	11.476
716	2025-05-10 18:08:43.147	78.144	0.079	35.474
717	2025-05-10 18:08:43.403	78.274	0.081	59.472
718	2025-05-10 18:08:43.650	78.474	0.088	83.475
719	2025-05-10 18:08:43.907	78.538	0.089	110.431
720	2025-05-10 18:08:44.143	78.661	0.09	131.466
721	2025-05-10 18:08:44.420	78.813	0.091	155.464
722	2025-05-10 18:08:44.673	78.937	0.091	179.462
723	2025-05-10 18:08:44.910	79.063	0.092	203.46
724	2025-05-10 18:08:45.153	79.199	0.093	227.458
725	2025-05-10 18:08:45.400	79.327	0.093	251.456
726	2025-05-10 18:08:45.647	79.451	0.093	275.454
727	2025-05-10 18:08:45.937	79.585	0.095	299.452
728	2025-05-10 18:08:46.160	79.719	0.096	323.45
729	2025-05-10 18:08:46.410	79.849	0.97	347.448
730	2025-05-10 18:08:46.657	79.974	0.973	371.446
731	2025-05-10 18:08:46.907	80.116	0.975	395.444
732	2025-05-10 18:08:47.157	80.247	0.976	419.442
733	2025-05-10 18:08:47.407	80.378	0.977	443.44
734	2025-05-10 18:08:47.660	80.58	0.979	467.438
735	2025-05-10 18:08:47.917	80.673	0.98	491.436
736	2025-05-10 18:08:48.160	80.765	0.981	515.434
737	2025-05-10 18:08:48.420	80.896	0.982	539.432
738	2025-05-10 18:08:48.733	81.027	0.984	563.43

Query executed successfully.

Ready

Figura 99. Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 2.

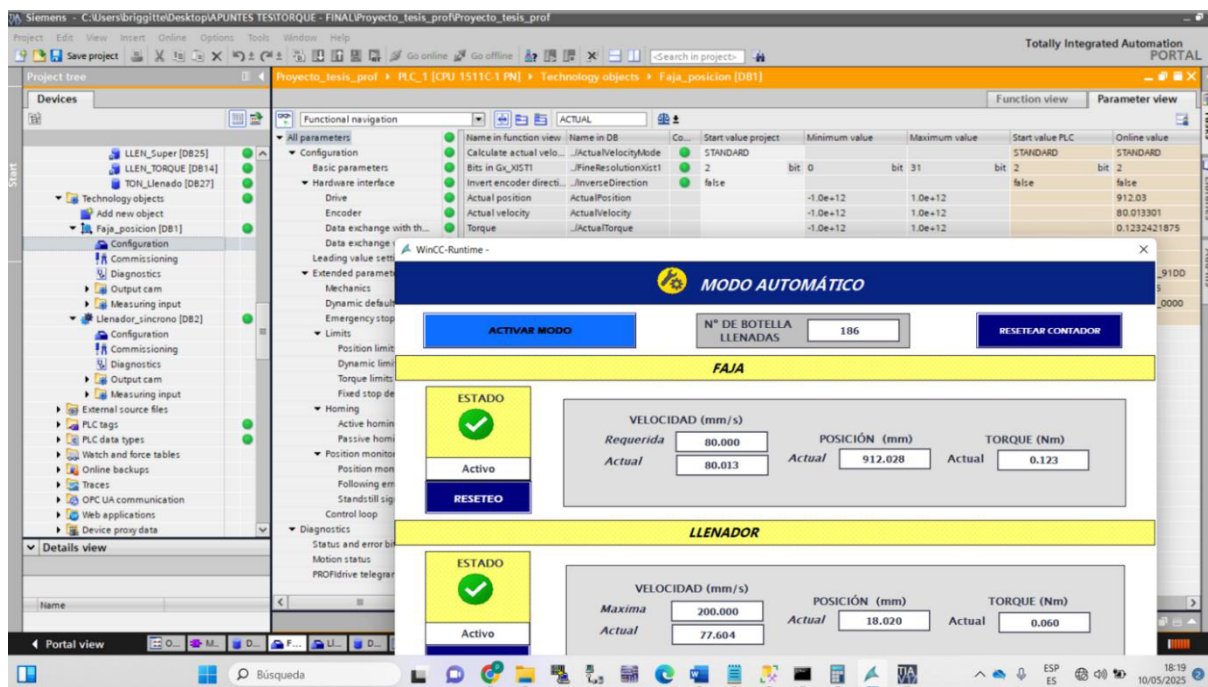


Figura 100. Visualización de variables dinámicas de la faja en la pantalla desarrollada en WinCC Professional – Caso 3.

SQLQuery17.sql - LAPTOP-SJQG2EJP.master (LAPTOP-SJQG2EJP\brigitte (57)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

master

Object Explorer

Connect

LAPTOP-SJQG2EJP (SQL Server 16.0.10)

Databases

System Databases

Database Snapshots

FAJA

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.SERVO1

Dropped Ledger Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

LLENADOR

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.SERVO2

Dropped Ledger Tables

Views

External Resources

Synonyms

Programmability

Query Store

SQLQuery17.sql - LA...EJP\brigitte (57))

SQLQuery18.sql - LA...EJP\brigitte (71))

```
SELECT [Fecha_hora]
,[Velocidad_Faja_mm_s]
,[Torque_Faja_Nm]
,[Posicion_Faja_mm]
FROM [FAJA].[dbo].[SERVO1]
ORDER BY [Fecha_hora] ASC
```

100 %

Results Messages

	Fecha_hora	Velocidad_Faja_mm_s	Torque_Faja_Nm	Posicion_Faja_mm
1400	2025-05-10 18:19:53.400	74.776	0.099	432.012
1401	2025-05-10 18:19:53.643	74.986	0.101	456.011
1402	2025-05-10 18:19:53.900	75.196	0.101	480.017
1403	2025-05-10 18:19:54.140	75.406	0.102	504.011
1404	2025-05-10 18:19:54.437	75.616	0.103	528.012
1405	2025-05-10 18:19:54.697	75.827	0.104	552.017
1406	2025-05-10 18:19:54.923	76.035	0.106	576.013
1407	2025-05-10 18:19:55.160	76.249	0.107	600.017
1408	2025-05-10 18:19:55.423	76.453	0.108	624.016
1409	2025-05-10 18:19:55.667	76.665	0.109	648.017
1410	2025-05-10 18:19:55.907	76.873	0.11	672.019
1411	2025-05-10 18:19:56.177	77.085	0.111	696.017
1412	2025-05-10 18:19:56.400	77.299	0.113	720.023
1413	2025-05-10 18:19:56.653	77.503	0.114	744.021
1414	2025-05-10 18:19:56.900	77.715	0.115	768.022
1415	2025-05-10 18:19:57.143	77.926	0.117	792.023
1416	2025-05-10 18:19:57.400	78.133	0.118	816.024
1417	2025-05-10 18:19:57.660	78.347	0.119	840.025
1418	2025-05-10 18:19:57.907	78.551	0.119	864.026
1419	2025-05-10 18:19:58.183	78.766	0.12	888.027
1420	2025-05-10 18:19:58.417	80.013	0.123	912.028
1421	2025-05-10 18:19:58.657	80.094	0.124	936.029
1422	2025-05-10 18:19:58.907	80.176	0.126	960.037

Query executed successfully.

Ready

Figura 101. Visualización de variables dinámicas de la faja en SQL Server Management Studio 20 (SSMS20) – Caso 3.

En la Figura 102 se observa que los valores coincidían tanto en la base de datos como en la interfaz de parámetros de objetos tecnológicos. Esta misma correspondencia se aprecia en las Figuras 104 y 106, en el caso de las variables dinámicas del servomotor encargado del dispensador o llenador.

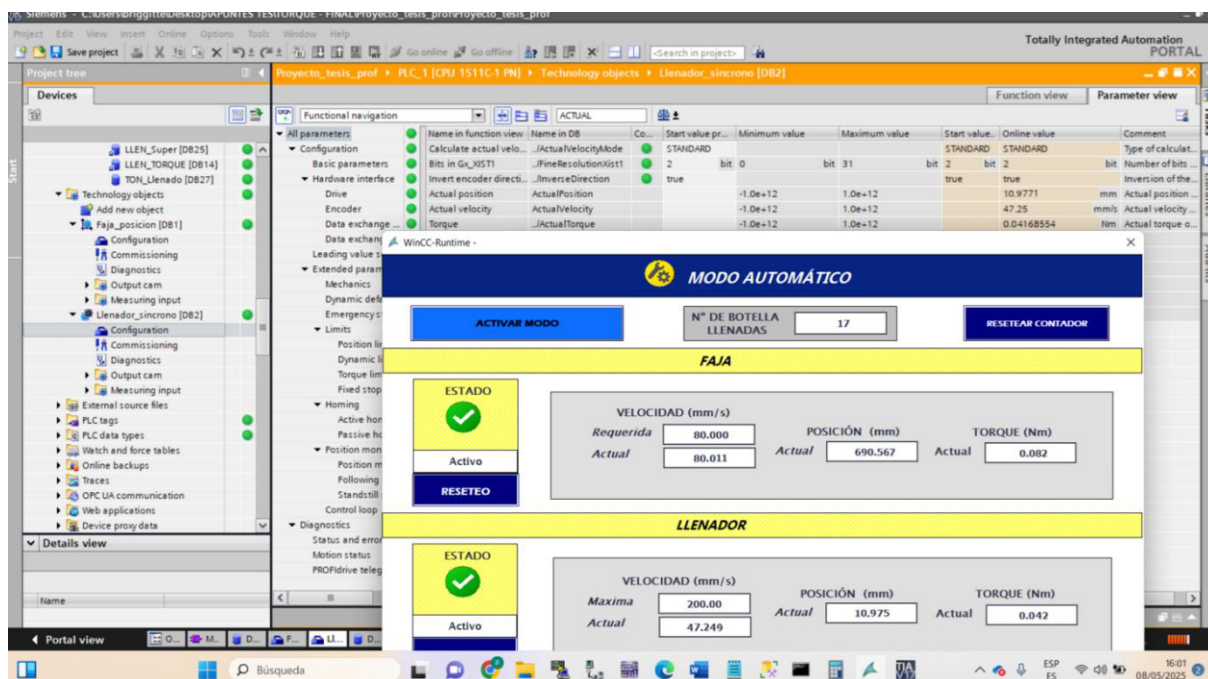


Figura 102. Visualización de variables dinámicas del llenador en la pantalla desarrollada en WinCC Professional – Caso 1.

SQLQuery20.sql - LAPTOP-SJQG2EJP.LLENADOR (LAPTOP-SJQG2EJP\briggitte (81)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Connect LAPTOP-SJQG2EJP (SQL Server 16.0.10)

Databases

- System Databases
- Database Snapshots
- FAJA
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERV01
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
- LLENADOR
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.SERV02
 - Dropped Ledger Tables
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store

SQLQuery18.sql - LA...EJP\briggitte (71)

```
SELECT [Fecha_hora]
      ,[Velocidad_Llenador_mm_s]
      ,[Torque_Llenador_Nm]
      ,[Posicion_Llenador_mm]
FROM [LLENADOR].[dbo].[SERVO2]
ORDER BY [Fecha_hora] ASC
```

SQLQuery17.sql - LA...EJP\briggitte (57)

Results

	Fecha_hora	Velocidad_Llenador_mm_s	Torque_Llenador_Nm	Posicion_Llenador_mm
40	2025-05-08 16:01:03.950	43.425	0.036	10.131
41	2025-05-08 16:01:04.130	43.695	0.036	10.185
42	2025-05-08 16:01:04.333	43.952	0.037	10.241
43	2025-05-08 16:01:04.587	44.204	0.037	10.312
44	2025-05-08 16:01:04.840	44.458	0.038	10.356
45	2025-05-08 16:01:05.090	44.712	0.038	10.415
46	2025-05-08 16:01:05.347	44.965	0.039	10.463
47	2025-05-08 16:01:05.593	45.219	0.039	10.525
48	2025-05-08 16:01:05.840	45.473	0.039	10.579
49	2025-05-08 16:01:06.087	45.725	0.04	10.632
50	2025-05-08 16:01:06.330	45.987	0.04	10.692
51	2025-05-08 16:01:06.583	46.234	0.04	10.743
52	2025-05-08 16:01:06.837	46.485	0.041	10.811
53	2025-05-08 16:01:07.083	46.741	0.041	10.861
54	2025-05-08 16:01:07.333	46.995	0.041	10.918
55	2025-05-08 16:01:07.587	47.249	0.042	10.975
56	2025-05-08 16:01:07.840	47.503	0.042	11.031
57	2025-05-08 16:01:08.087	47.756	0.043	11.088
58	2025-05-08 16:01:08.337	48.01	0.043	11.144
59	2025-05-08 16:01:08.587	48.264	0.044	11.212
60	2025-05-08 16:01:08.837	48.518	0.044	11.256
61	2025-05-08 16:01:09.083	48.772	0.045	11.312
62	2025-05-08 16:01:09.340	49.025	0.045	11.369
63	2025-05-08 16:01:09.587	49.279	0.046	11.425
64	2025-05-08 16:01:09.840	49.533	0.046	11.481

Query executed successfully.

Ready Ln 6

Figura 103. Variables dinámicas del llenador mostradas en SQL Server Management Studio 20 (Caso 1).

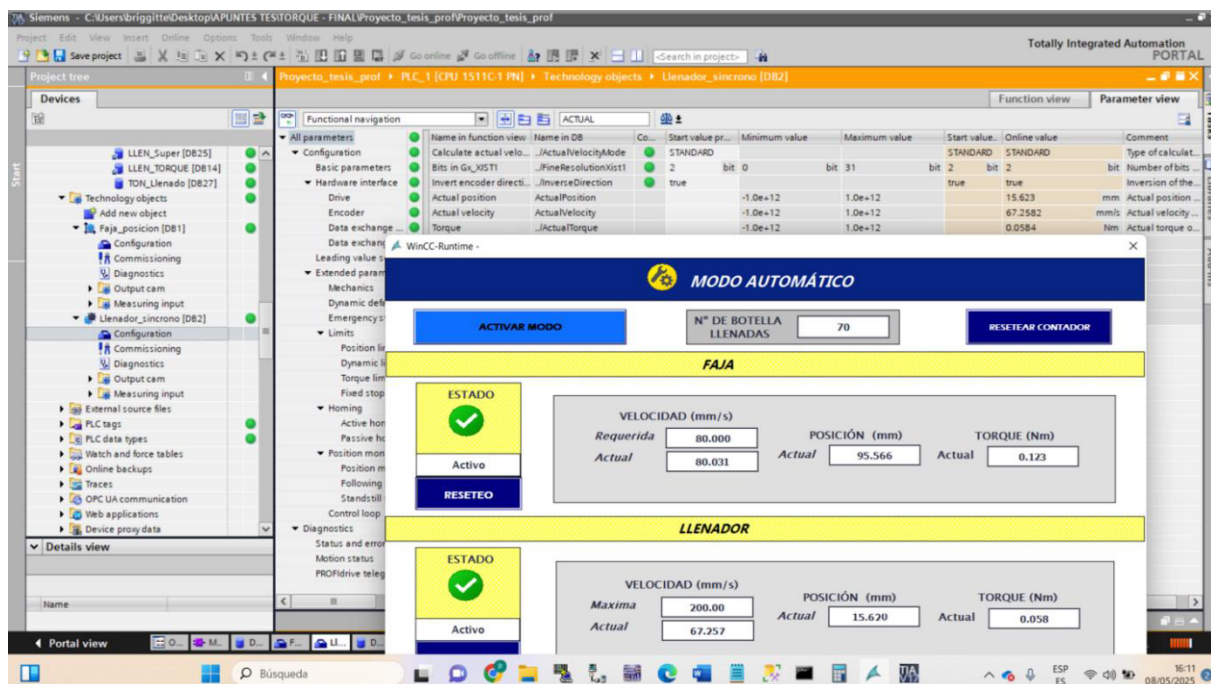


Figura 104. Visualización de variables dinámicas del llenador en la interfaz desarrollada en WinCC Professional (Caso 2).

SQLQuery20.sql - LAPTOP-SJQG2EJP.LLENADOR (LAPTOP-SJQG2EJP\briggite (81)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

LEENADOR Execute

Object Explorer

Connect

LAPTOP-SJQG2EJP (SQL Server 16.0.10)

Databases

System Databases

Database Snapshots

FAJA

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.SERVO1

Dropped Ledger Tables

Views

External Resources

Synonyms

Programmability

Query Store

Service Broker

Storage

Security

LLENADOR

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.SERVO2

Dropped Ledger Tables

Views

External Resources

Synonyms

Programmability

Query Store

SQLQuery18.sql - LA...EJP\briggite (71))

SQLQuery17.sql - LA...EJP\briggite (57))

```
SELECT [Fecha_hora]
, [Velocidad_Llenador_mm_s]
, [Torque_Llenador_Nm]
, [Posicion_Llenador_mm]
FROM [LLENADOR].[dbo].[SERVO2]
ORDER BY [Fecha_hora] ASC
```

100 %

Results Messages

	Fecha_hora	Velocidad_Llenador_mm_s	Torque_Llenador_Nm	Posicion_Llenador_mm
139	2025-05-08 16:11:12.093	52.036	0.043	14.776
140	2025-05-08 16:11:12.333	53.047	0.044	14.832
141	2025-05-08 16:11:12.587	54.063	0.045	14.873
142	2025-05-08 16:11:12.837	55.711	0.046	14.948
143	2025-05-08 16:11:13.087	56.052	0.047	15.001
144	2025-05-08 16:11:13.337	57.123	0.048	15.055
145	2025-05-08 16:11:13.583	58.122	0.049	15.117
146	2025-05-08 16:11:13.837	59.137	0.051	15.17
147	2025-05-08 16:11:14.087	60.153	0.051	15.226
148	2025-05-08 16:11:14.337	61.163	0.052	15.282
149	2025-05-08 16:11:14.590	62.188	0.053	15.339
150	2025-05-08 16:11:14.833	63.197	0.054	15.395
151	2025-05-08 16:11:15.087	64.212	0.055	15.451
152	2025-05-08 16:11:15.333	65.221	0.056	15.507
153	2025-05-08 16:11:15.583	66.245	0.057	15.564
154	2025-05-08 16:11:15.837	67.257	0.058	15.62
155	2025-05-08 16:11:16.083	68.272	0.059	15.673
156	2025-05-08 16:11:16.343	69.287	0.06	15.731
157	2025-05-08 16:11:16.590	70.302	0.061	15.781
158	2025-05-08 16:11:16.833	71.317	0.062	15.844
159	2025-05-08 16:11:17.083	72.332	0.063	15.902
160	2025-05-08 16:11:17.337	73.347	0.064	15.953
161	2025-05-08 16:11:17.590	74.362	0.065	16.019
162	2025-05-08 16:11:17.833	75.377	0.066	16.068
163	2025-05-08 16:11:18.087	76.392	0.066	16.129

Query executed successfully.

Ready Ln 6

Figura 105. Visualización de variables dinámicas del llenador en SQL Server Management Studio 20 (SSMS 20) (Caso 2).

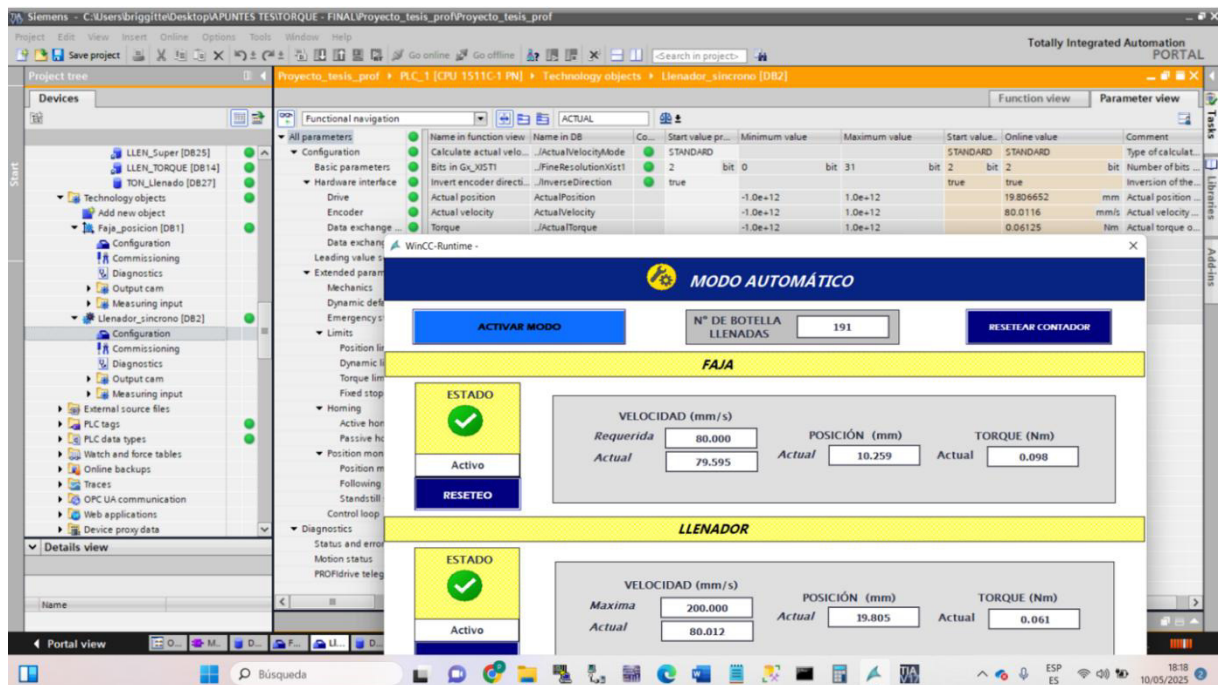


Figura 106. Visualización de variables dinámicas del llenador en la interfaz desarrollada en WinCC Professional (Caso 3).

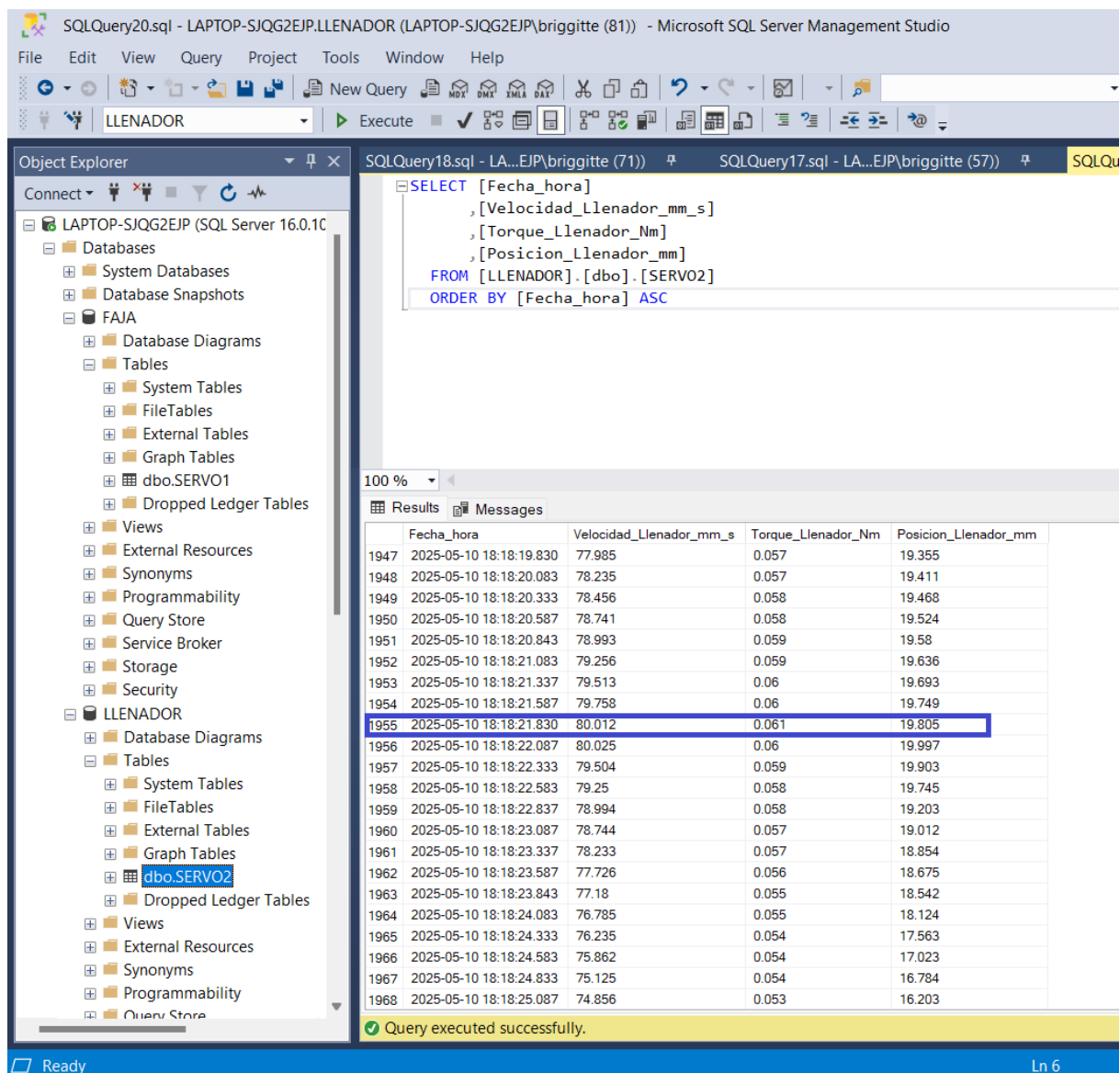


Figura 107. Visualización de variables dinámicas del llenador en SQL Server Management Studio 20 (SSMS 20) (Caso 3).

La pantalla desarrollada en WinCC Professional también permitió visualizar las bases de datos, tablas y sus respectivas columnas que estaban disponibles. En la Figura 108 se observan las bases de datos disponibles según lo mostraba SMSS, lo que confirmó el funcionamiento óptimo del script ejecutado.

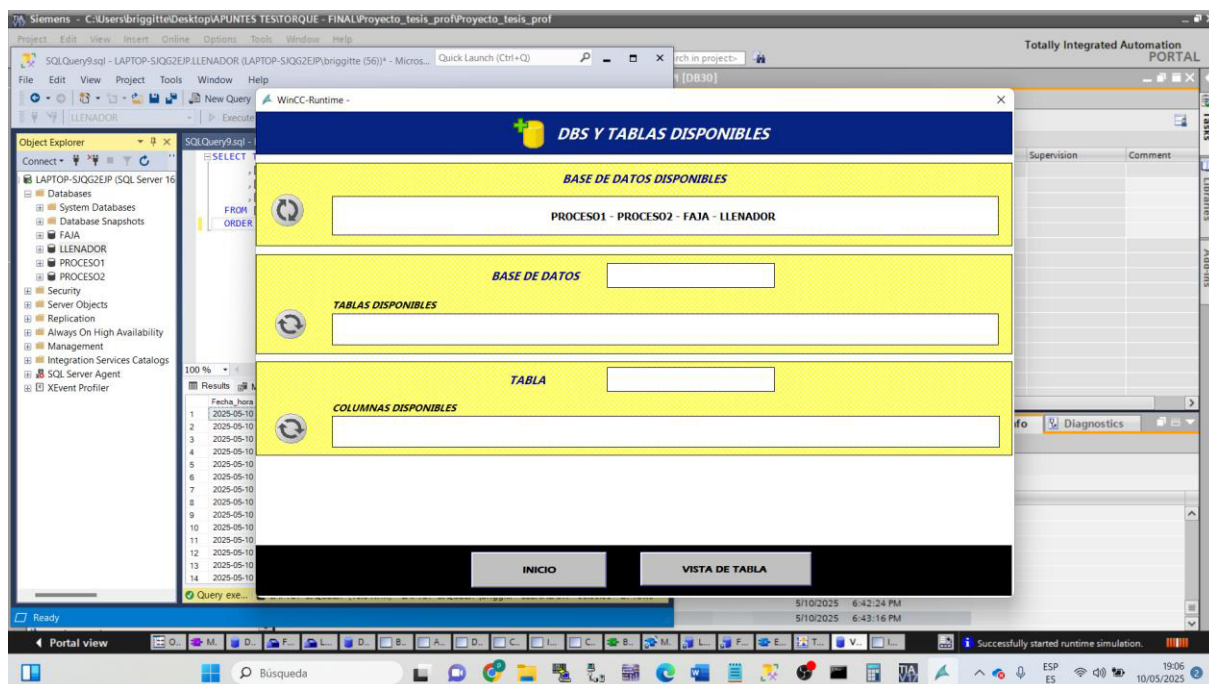


Figura 108. Bases de datos disponibles.

Al indicar una base de datos, se pudieron obtener todas sus tablas, tal como se muestra en la Figura 109. Se confirmó que el resultado era correcto, ya que en el SMSS se verificó que la base de datos “FAJA” contenía únicamente a “SERVO1” como tabla.

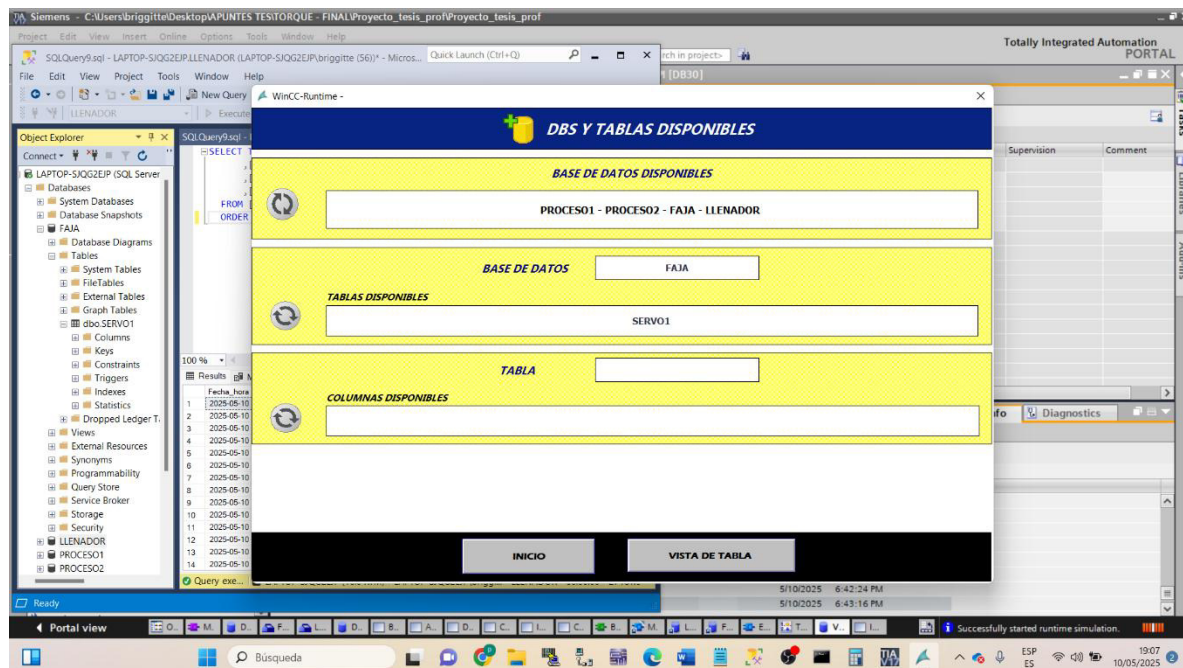


Figura 109. Tablas disponibles según base de dato seleccionada.

Si el usuario omitió ingresar el nombre de la base de datos o de las tablas a buscar, se mostró inmediatamente un mensaje informando el error para que pudiera ser corregido (véase la Figura 110).

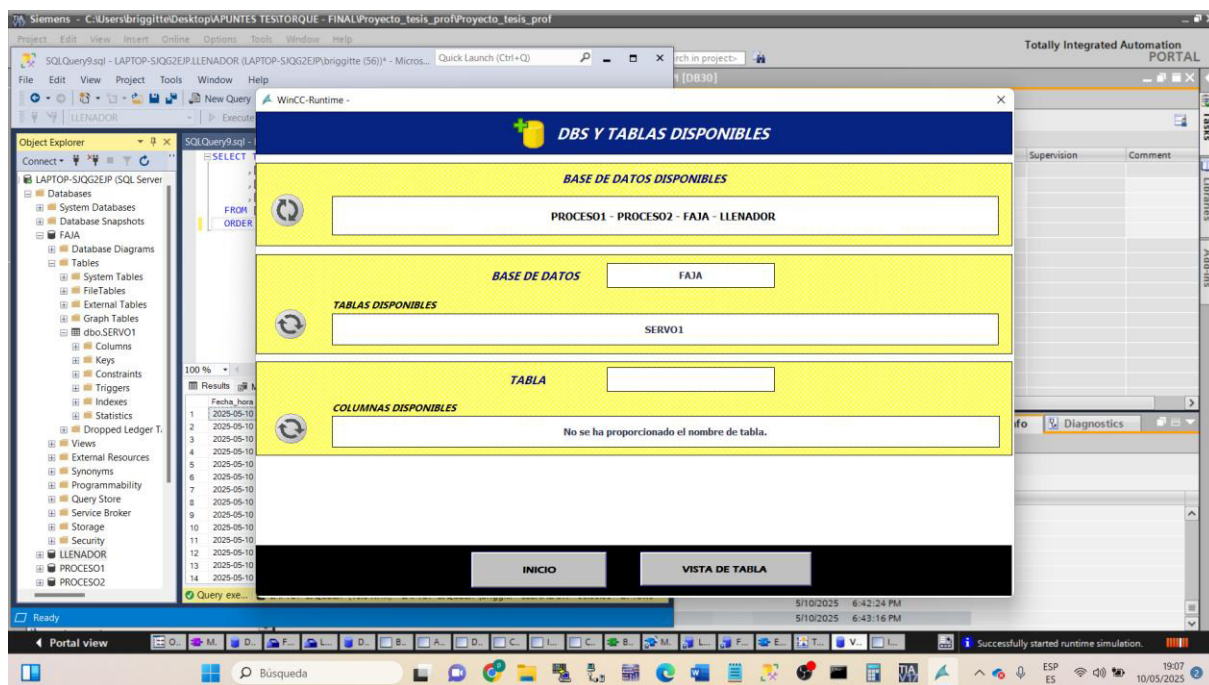


Figura 110. Caso en que no se brinde datos de búsqueda.

Al indicar una tabla, se obtuvieron todas sus columnas, tal como se muestra en la Figura 111.

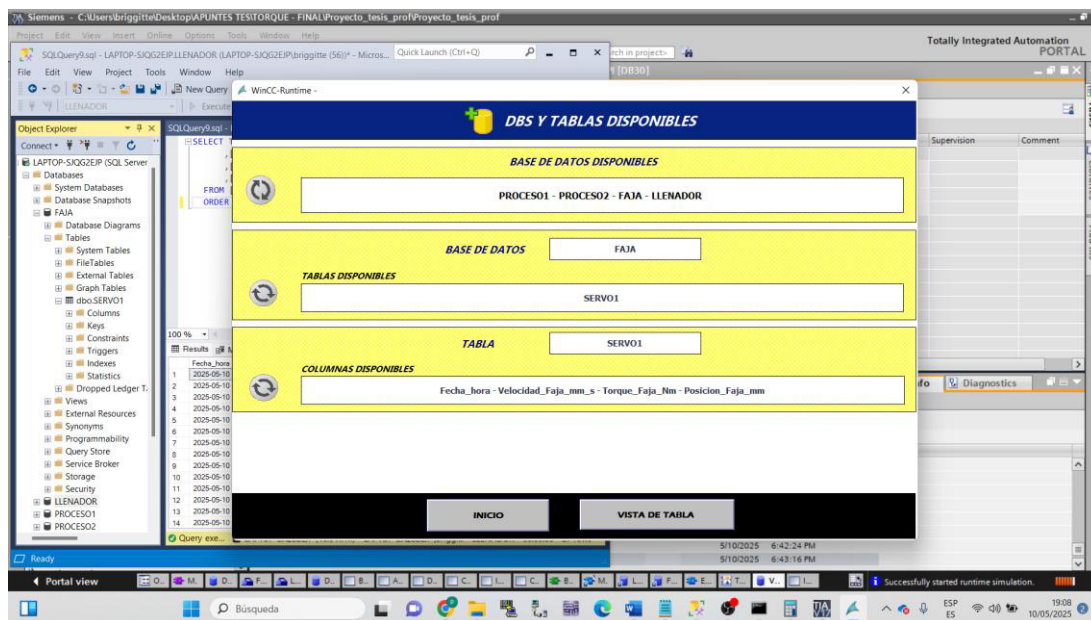


Figura 111. Columnas disponibles según tabla seleccionada.

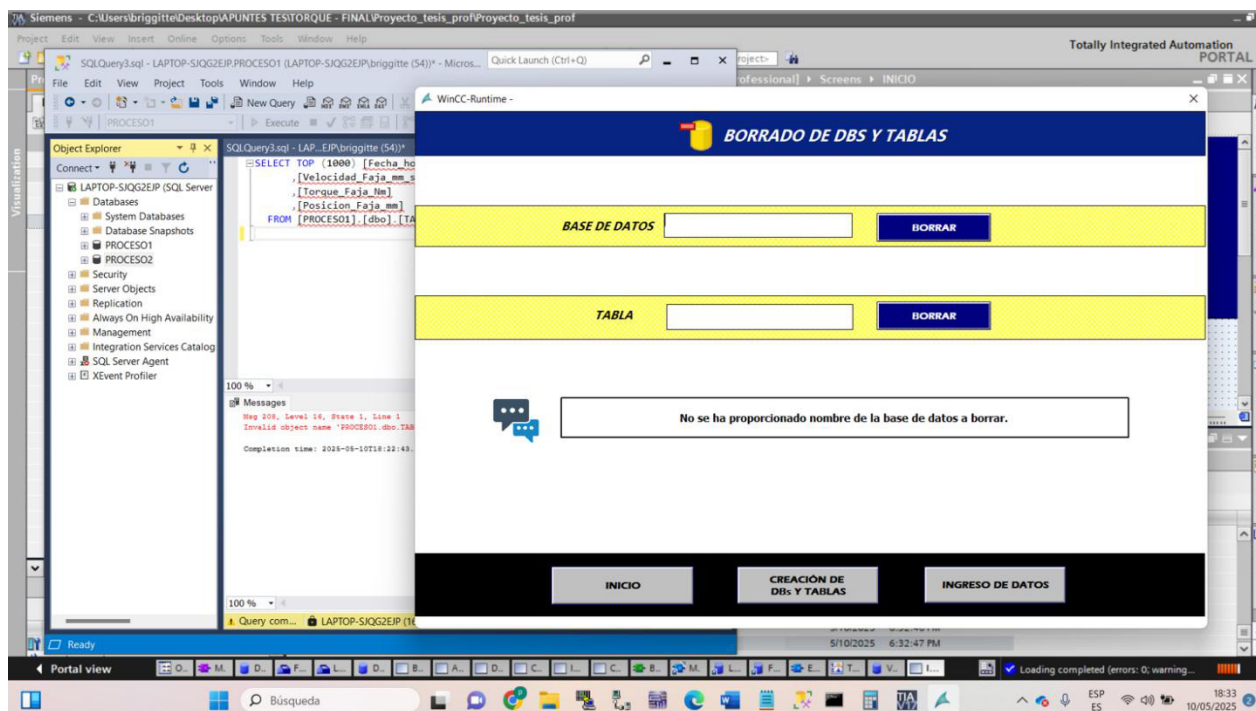


Figura 113. Falta de datos para iniciar el borrado de una base de datos o tablas - Caso 1.

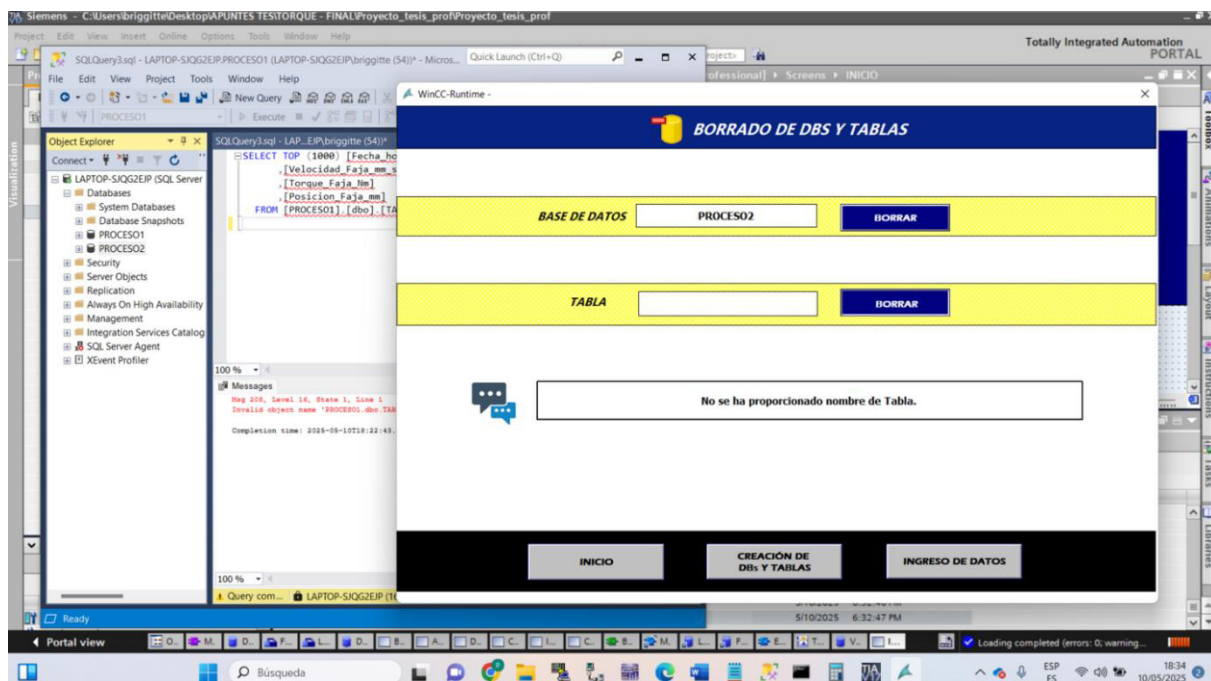


Figura 114. Falta de datos para iniciar el borrado de una base de datos o tablas - Caso 2.

En caso de que el usuario intentara borrar una tabla que ya no existiera porque había sido eliminada previamente, el sistema le informó mediante un mensaje (véase la Figura 115).

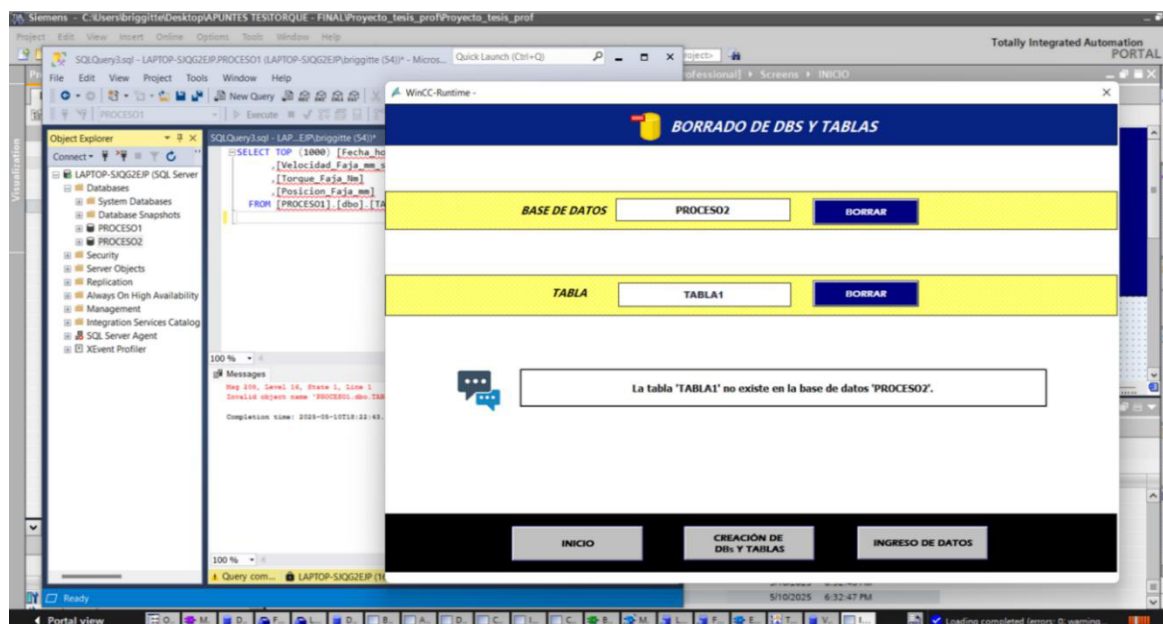


Figura 115. Borrado de una tabla inexistente.

La pantalla permitió mostrar la totalidad de los datos en grupos de 8 filas. El usuario indicó previamente la base de datos y la tabla. Una vez obtenidos los resultados (véase la Figura 116), pudo desplazarse por todos los datos utilizando los botones superiores.

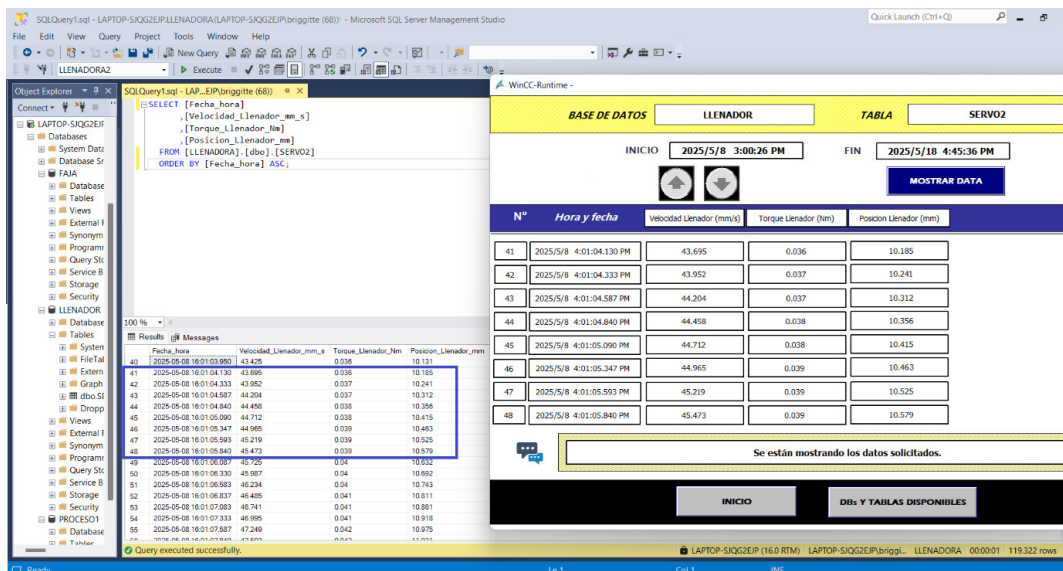


Figura 116. Tabla de datos mostrados en una pantalla desarrollada en WinCC Professional.

V.1.2. Sistema remoto

Se inició con el envío de las variables necesarias para el cálculo de los KPIs. A continuación, se muestran los datos que se enviaron desde TIA Portal a Node - RED para su posterior almacenamiento en InfluxDB Cloud. En TIA Portal, se designó el bloque de datos “Var_KPI” [DB30] (véase la Figura 117) para agrupar las variables de interés.

The screenshot displays the TIA Portal interface. The top window shows the 'Var_KPI' data block (DB30) with the following data:

Name	Data type	Offset	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
Static											
Botellas	Int	0.0	0	1483		☒	☒	☒	☐		
ProductoBueno	Int	2.0	0	1474		☒	☒	☒	☐		
Merma	Int	4.0	0	91		☒	☒	☒	☐		
Minutos	Real	6.0	0.0	122.0667		☒	☒	☒	☐		

Overlaid on this is the 'WinCC-Runtime' window, which is in 'MODO AUTOMÁTICO'. It displays the following information:

- ACTIVAR MODO** button
- N° DE BOTELLA LLENADAS**: 1483
- RESETEAR CONTADOR** button
- FAJA** section:
 - ESTADO**: ☒ (Green checkmark icon)
 - Activo** button
 - RESETEO** button
 - VELOCIDAD (mm/s)**: Requerida: 80.00, Actual: 80.03
 - POSICIÓN (mm)**: Actual: 674.28
 - TORQUE (Nm)**: Actual: 0.120
- LLENADOR** section:
 - ESTADO**: ☒ (Green checkmark icon)
 - VELOCIDAD (mm/s)**: Maxima: 200.00
 - POSICIÓN (mm)**
 - TORQUE (Nm)**

Figura 117. Bloque de datos Var_KPI [DB30] en TIA Portal.

En Node – RED se visualizó que todos los datos de “Var_KPI” [DB30] fueron transmitidos a InfluxDB Cloud (véase la Figura 118).



Figura 118. Envío de datos a Influx DB Cloud desde Node - RED.

En InfluxDB Cloud se mostraron los datos a medida que eran recibidos, ordenados de forma ascendente para facilitar un mejor control de la información (véase la Figura 119).

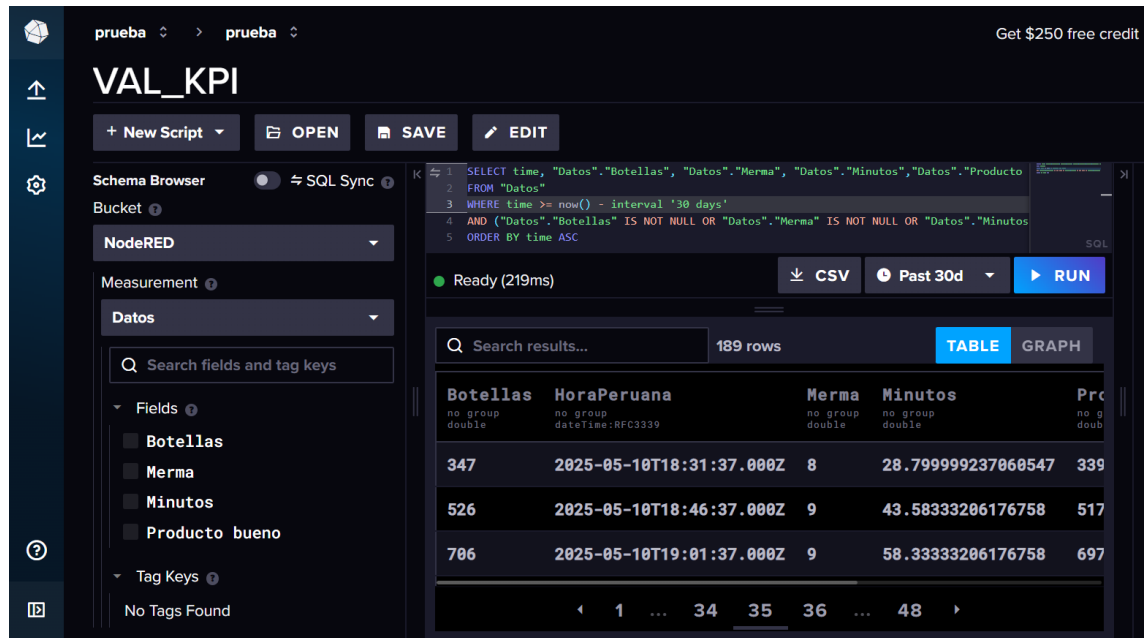


Figura 119. Datos recepcionados por InfluxDB Cloud.

Luego, estos datos fueron utilizados por Grafana Cloud para el cálculo de los KPIs y la elaboración de gráficas. Tomando como referencia algunos periodos de trabajo (véase la Figura 120), se aplicaron las fórmulas correspondientes con el fin de comprobar el correcto funcionamiento del sistema.



Figura 120. Períodos de toma de datos en Node-RED a analizar.

Se observó como los datos fueron recibidos en InfluxDB Cloud (véanse las Figuras 121 y 122).

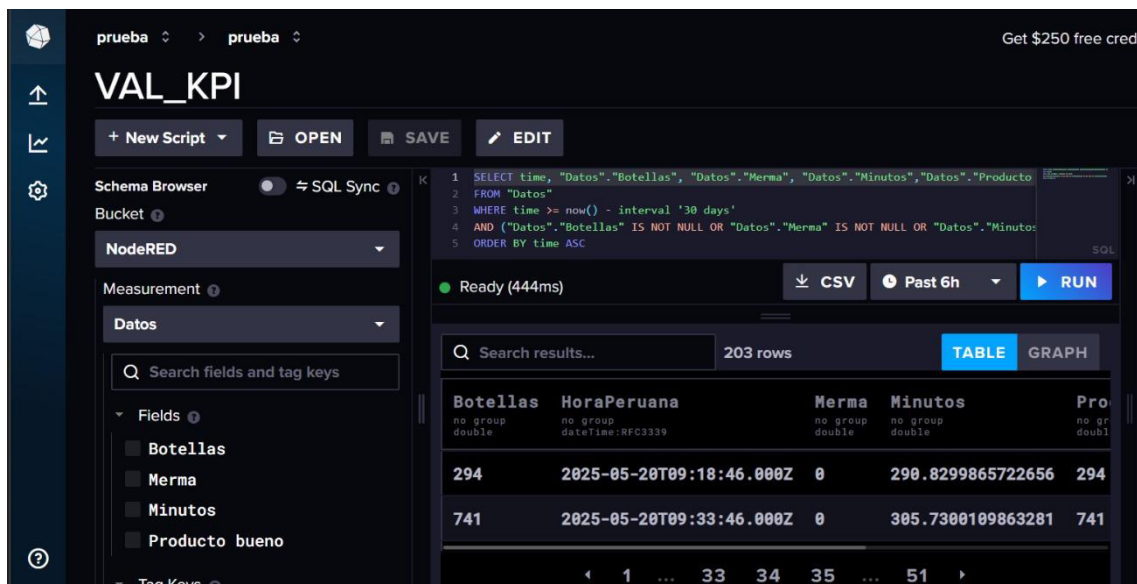


Figura 121. Recepción de datos de Node-RED – Parte 1.

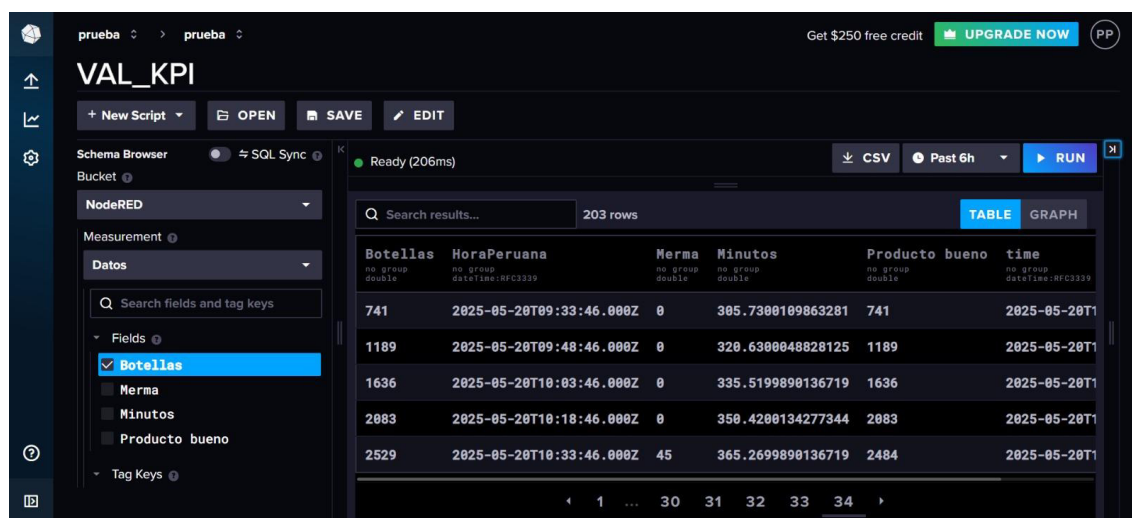


Figura 122. Recepción de datos de Node-RED – Parte 2.

Se procedió al cálculo de los KPIs según las fórmulas implementadas en los scripts de Grafana Cloud, en las cuales se consideraron los retrasos en el envío de datos. Estos cálculos permitieron verificar el correcto funcionamiento del dashboard.

Período: 10:18 hasta las 10:33 horas

Tasa de utilización de capacidad:

El cálculo se muestra en las ecuaciones 11, 12 y 13. En la Figura 123, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 13.

$$Tasa\ de\ utilización\ de\ capacidad = \frac{Producción\ total}{Total\ de\ producción\ teórica} \times 100 \quad (11)$$

$$Tasa\ de\ utilización\ de\ capacidad = \frac{2529-2083}{450} \times 100 \quad (12)$$

$$Tasa\ de\ utilización\ de\ capacidad = \frac{446}{450} \times 100 = 99.1\% \quad (13)$$

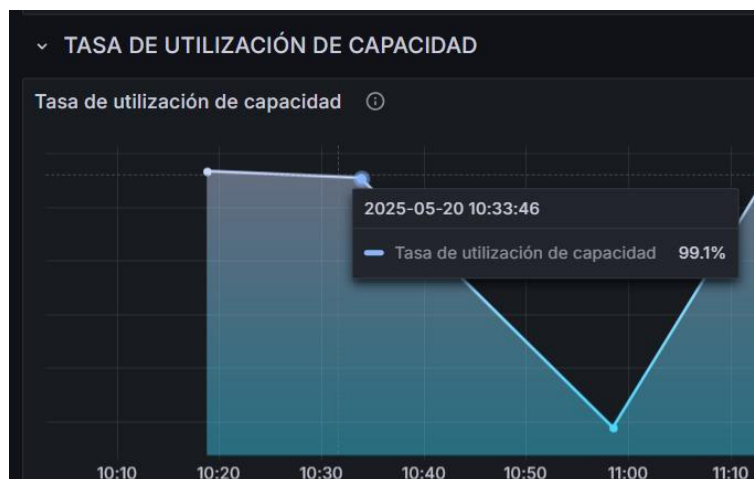


Figura 123. Tasa de utilización de capacidad – Caso 1.

Tasa de disponibilidad:

El cálculo se muestra en las ecuaciones 14, 15 y 16. En la Figura 124, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 16.

$$Disponibilidad = \frac{Tiempo\ de\ ejecución}{Tiempo\ de\ producción\ planificado} \quad (14)$$

$$Disponibilidad = \frac{365.27-350.42+0.1}{15} \times 100 \quad (15)$$

$$Disponibilidad = \frac{14.95}{15} \times 100 = 99.67\% \quad (16)$$

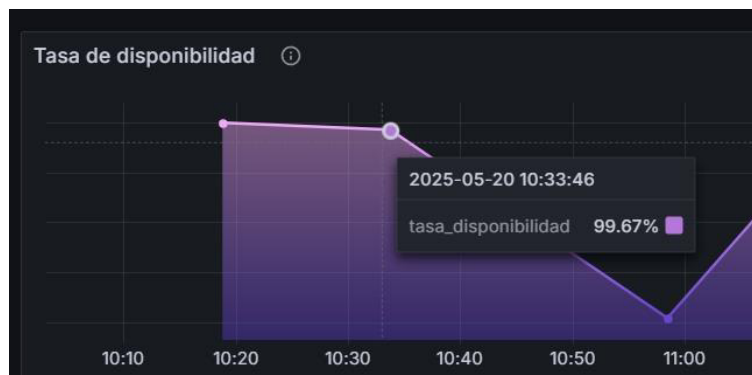


Figura 124. Tasa de disponibilidad – Caso 1.

Tasa de calidad:

El cálculo se muestra en las ecuaciones 17 y 18. En la Figura 125, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 18.

$$Calidad = \frac{\text{Unidades en buena calidad}}{\text{Unidades totales producidas}} \times 100 \quad (17)$$

$$Calidad = \frac{2484}{2529} \times 100 = 98.22\% \quad (18)$$



Figura 125. Tasa de calidad – Caso 1.

Productividad:

El cálculo se muestra en las ecuaciones 19, 20 y 21. En la Figura 126, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 21.

$$Productividad = \frac{Producción\ total - Merma}{Tiempo\ trabajado} \quad (19)$$

$$Productividad = \frac{2484 - 2083}{14.85 + 0.1} \quad (20)$$

$$Productividad = \frac{401}{14.85 + 0.1} = 26.82 \quad (21)$$



Figura 126. Productividad – Caso 1.

Se observa en la Figura 127, el periodo de 11:43 hasta las 11:58 horas.

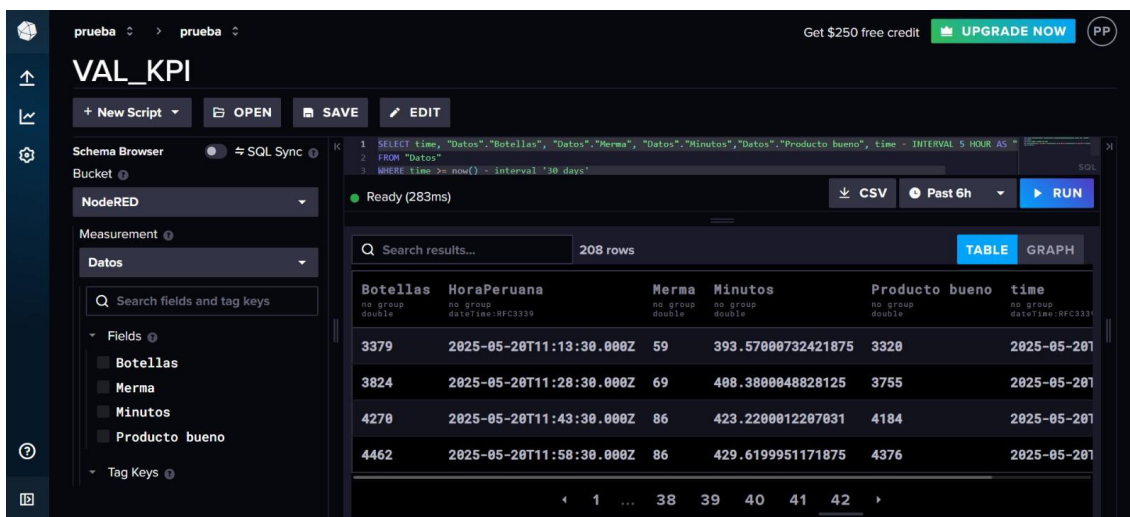


Figura 127. Recepción de datos de Node-RED – Parte 3.

Período: 11:43 hasta las 11:58 horas

Tasa de utilización de capacidad:

El cálculo se muestra en las ecuaciones 22, 23 y 24. En la Figura 128, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 24.

$$Tasa\ de\ utilización\ de\ capacidad = \frac{Producción\ total}{Total\ de\ producción\ teorica} \times 100 \quad (22)$$

$$Tasa\ de\ utilización\ de\ capacidad = \frac{4462-4270}{450} \times 100 \quad (23)$$

$$Tasa\ de\ utilización\ de\ capacidad = \frac{192}{450} \times 100 = 42.7\% \quad (24)$$



Figura 128. Tasa de utilización de capacidad – Caso 2.

Tasa de disponibilidad:

El cálculo se muestra en las ecuaciones 25, 26 y 27. En la Figura 129, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 27.

$$\text{Disponibilidad} = \frac{\text{Tiempo de ejecución}}{\text{Tiempo de producción planificado}} \quad (25)$$

$$\text{Disponibilidad} = \frac{429.62 - 423.22 + 0.1}{15} \times 100 \quad (26)$$

$$\text{Disponibilidad} = \frac{6.5}{15} \times 100 = 43.33\% \quad (27)$$



Figura 129. Tasa de disponibilidad – Caso 2.

Tasa de calidad:

El cálculo se muestra en las ecuaciones 28 y 29. En la Figura 130, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 29.

$$Calidad = \frac{Unidades\ en\ buena\ calidad}{Unidades\ totales\ producidas} \times 100 \quad (28)$$

$$Calidad = \frac{4376}{4462} \times 100 = 98.1\% \quad (29)$$



Figura 130. Tasa de calidad – Caso 2.

Productividad:

El cálculo se muestra en las ecuaciones 30, 31 y 32. En la Figura 131, se observa que el valor calculado en Grafana Cloud coincide con el resultado de la ecuación 32.

$$Productividad = \frac{Producción\ total - Merma}{Tiempo\ trabajado} \quad (30)$$

$$Productividad = \frac{4376 - 4184}{6.5} \quad (31)$$

$$Productividad = \frac{192}{6.5} = 29.54 \quad (32)$$

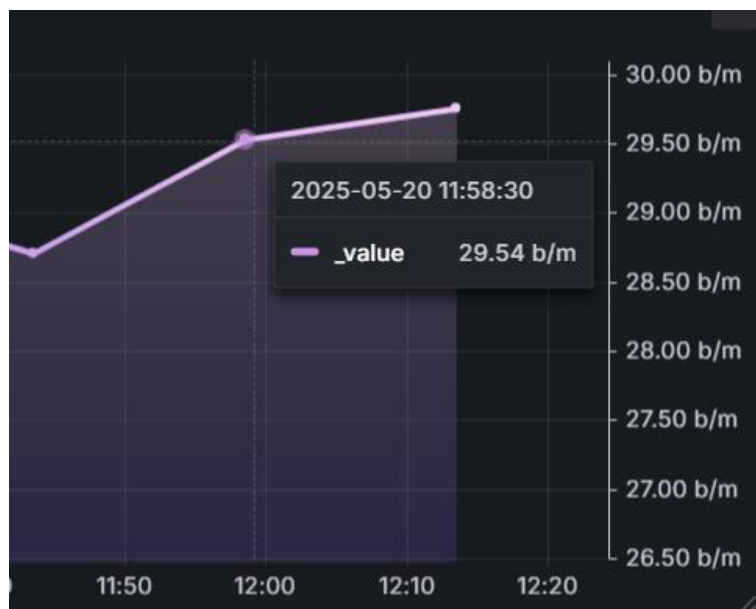


Figura 131. Productividad – Caso 2.

Se presenta la visualización final del dashboard (véanse las Figuras 132, 133, 134 y 135).

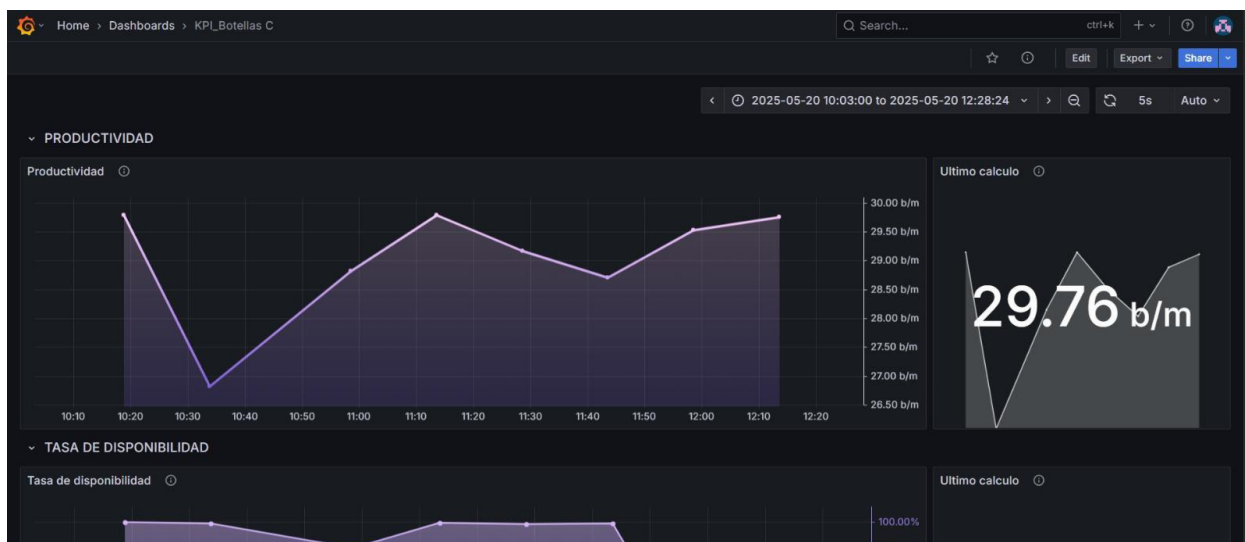


Figura 132. Dashboard realizado – Parte 1.



Figura 133. Dashboard realizado – Parte 2.

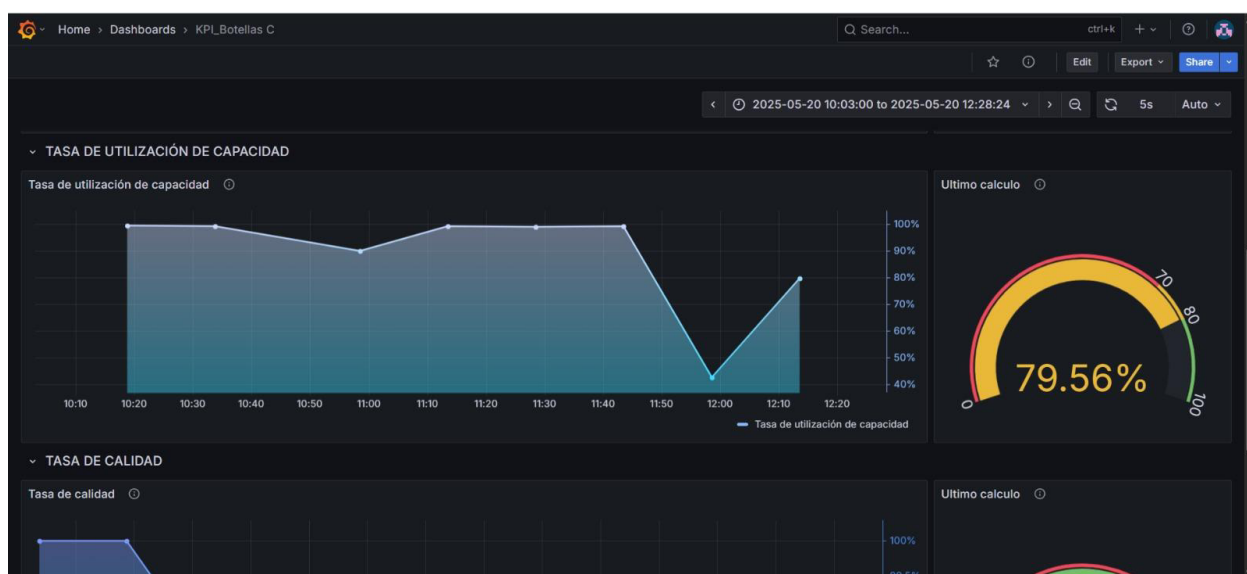


Figura 134. Dashboard realizado – Parte 3.



Figura 135. Dashboard realizado – Parte 4.

VI. DISCUSIÓN

Los hallazgos obtenidos guardan correspondencia con lo planteado por Rímac (2024), quien enfatiza que la incorporación de InfluxDB como base de datos orientada a series temporales constituye un recurso esencial para el almacenamiento continuo y estructurado de la información, permitiendo representarla de manera ordenada a lo largo del tiempo. Asimismo, se confirma la relevancia de Grafana como plataforma de monitoreo, puesto que facilita la visualización dinámica e interactiva de los registros.

De manera complementaria, los resultados también se relacionan con lo expuesto por Peña (2022), quien desarrolla una la aplicación de InfluxDB y Grafana que contribuye significativamente a la optimización de los tiempos de respuesta y la minimización de errores en la transmisión de datos, lo que garantiza que el usuario disponga de información confiable y actualizada para respaldar la toma de decisiones operativas. Estos aportes permiten sostener que el uso de dichas herramientas no solo incrementa la eficiencia en la gestión de datos industriales, sino que también refuerza la seguridad y precisión de los procesos de monitoreo continuo.

En relación con el monitoreo de procesos, esta tesis evidencia la utilidad de emplear interfaces desarrolladas por el propio fabricante Siemens, lo que se encuentra en concordancia con lo expuesto por Echevarría (2019), quien resalta el papel fundamental del HMI (Human Machine Interface) en la supervisión y control de datos. Este hallazgo resulta relevante, dado que confirma que la utilización de estas plataformas permite no solo una interacción más directa y eficiente entre el usuario y el sistema, sino también una mayor confiabilidad en la interpretación de la información procesada.

VII. CONCLUSIONES Y RECOMENDACIONES

VII.1. Conclusiones

Se logró implementar un sistema local de monitoreo de variables dinámicas de los servomotores en el proceso de llenado de botellas, utilizando la interfaz gráfica desarrollada en WinCC Professional, lo que permitió la supervisión en tiempo real del estado operativo del sistema.

El sistema de almacenamiento local de las variables dinámicas fue implementado de manera efectiva mediante la base de datos SQL Server, asegurando la persistencia y trazabilidad de los datos obtenidos durante el proceso de llenado.

Se desarrolló exitosamente un sistema remoto de visualización de KPIs mediante un panel en Grafana Cloud, lo que posibilitó el monitoreo a distancia de los indicadores clave del proceso, facilitando la toma de decisiones basada en datos.

Se comprobó la factibilidad de almacenar remotamente las variables necesarias para el cálculo de KPIs en la base de datos InfluxDB Cloud, permitiendo una gestión eficiente y segura de la información desde entornos externos.

La arquitectura del sistema resultó escalable y adaptable a procesos similares de automatización industrial, lo que abre la posibilidad de replicar este enfoque en otros contextos productivos que requieran control y análisis de datos en tiempo real.

La interfaz de usuario implementada en WinCC Professional proporcionó una visualización clara e intuitiva del estado de los servomotores, lo que facilita su uso por parte de los operarios y contribuye a una mayor eficiencia operativa.

La integración con SQL Server permitió no solo el almacenamiento de los datos, sino también su estructuración para facilitar el análisis posterior, lo cual constituye una ventaja en la mejora continua del proceso.

El panel desarrollado en Grafana Cloud demostró ser una herramienta eficaz para el seguimiento de KPIs, gracias a su capacidad para representar gráficamente grandes volúmenes de datos de manera comprensible y dinámica.

InfluxDB Cloud resultó ser una solución robusta y eficiente para el almacenamiento remoto de variables de proceso, especialmente en entornos industriales que requieren alta disponibilidad y bajo tiempo de respuesta.

VII.2. Recomendaciones

- Se recomienda designar una tabla para cada servomotor a probar, con el fin de evitar confusiones en el análisis de las variables dinámicas entre los diferentes servomotores presentes en una misma línea. En muchos casos, estos comparten algunas variables, pero cumplen funciones diferentes dentro del proceso de producción, por lo que es importante realizar un monitoreo correcto y específico.
- Se debe aprovechar la opción de eliminar tablas cuando ya se hayan realizado las pruebas y estudios necesarios, con el objetivo de optimizar el uso de almacenamiento local limitado y evitar costos elevados.
- Se recomienda utilizar gráficas lineales en Grafana Cloud, ya que permiten visualizar de manera clara y amplia los KPIs calculados durante periodos largos.
- Para mantener un orden en la elaboración de los scripts que ejecutan las tareas en la pantalla desarrollada en WinCC Professional, se sugiere incluir comentarios y observaciones, lo cual será beneficioso para futuras mejoras. De igual manera, en el caso del código presente en Grafana Cloud, donde se manejan parámetros según las metas de producción que pueden modificarse.

Referencias

Asociación de Bebidas Refrescantes (ANFABRA). (2024). ¿Cómo se fabrican los refrescos?

https://www.refrescantes.es/refrescos_como_se_hacen/

Atakok, G., Ersoy, S., & Girit, O. (2020). *Data analysis for predictive maintenance of servo motors*. *Journal of Sensors*, 2020, Article 8826802.

<https://doi.org/10.1155/2020/8826802>

Bolton, W. (2013). *Mecatrónica: Sistemas de control electrónico en la ingeniería mecánica y eléctrica* (5.^a ed.). Alfaomega.

Buczaj, A., & Cisek, J. (2024). *Impact of angular speed calculation methods from encoder measurements on the test uncertainty of electric motor efficiency*. *Metrology*, 4(2),

164–180. <https://doi.org/10.3390/metrology4020011>

Calvo, A. (2022). *Servomotor industrial: Puesta en servicio y supervisión a través de un autómatas programable* [Trabajo fin de grado, Universidad de Sevilla]. Repositorio de la Escuela Técnica Superior de Ingeniería.

<https://biblus.us.es/bibing/proyectos/abreproy/94211/fichero/TFG-4211+CALVO+CID%2C+ANTONIO.pdf>

Chaname, L., Cruz, J. C., Hernández, C., Rojas, J., & Tisnado, M. (2022). *Elaboración de la gaseosa Big Cola: Empresa AJE Group* [Trabajo académico, Universidad Tecnológica del Perú]. <https://es.scribd.com/document/597638878/BIG-COLA-1>

Echevarría, L. (2019). *Desarrollo de una base de datos utilizando TIA Portal V15 para registro de variables físicas* [Tesis de pregrado, Escuela Politécnica Nacional]. Repositorio Digital EPN. <http://bibdigital.epn.edu.ec/handle/15000/20447>

Ghosh, S., Joshi, R., Sampalli, S., & Zaman, M. (2024). Multi-phase quantum resistant framework for secure communication in SCADA systems. *IEEE Transactions on Dependable and Secure Computing*, 21(6), 5461–5478.
<https://doi.org/10.1109/TDSC.2024.3378474>

Grafana Labs. (2025). *K8S dashboard EN 20250125* [Panel de Grafana].
<https://grafana.com/grafana/dashboards/15661-k8s-dashboard-en-20250125/>

Hernández-Sampieri, R., Fernández, C., & Baptista, P. (2014). *Metodología de la investigación* (6.^a ed.). México D. F.: McGraw-Hill.

HIWIN MECTROL. (s.f.). *Guía lineal HIWIN EG perfil largo*.
<https://hiwinmectrol.com/guia-linear-hiwin-eg-perfil-largo>

Industrias Saisac. (s.f.). *Faja cerrada 2GT 6 mm × 100 mm*.
<https://industrias.saisac.pe/producto/faja-cerrada-2gt-6mm-x-100mm/>

InfluxData. (2025). *Consulta en el Data Explorer*. InfluxDB Cloud.

<https://docs.influxdata.com/influxdb/cloud/query-data/execute-queries/data-explorer/>

InfluxData. (2025). *InfluxDB Cloud*. <https://www.influxdata.com/products/influxdb-cloud/>

Kollmorgen. (2022). *What is the difference between a servo drive and a motion controller?*

<https://www.kollmorgen.com/en-us/blogs/what-difference-between-servo-drive-and-motion-controller>

Ling, S. J., Moebis, W., & Sanny, J. (2021). *Física universitaria. Volumen I*. OpenStax.

<https://openstax.org/books/f%C3%ADsica-universitaria-volumen-1/pages/1-introduccion>

Microsoft. (2025). *Configuración y componentes de SQL Server Management Studio*.

Microsoft Learn. <https://learn.microsoft.com/es-es/ssms/tutorials/ssms-configuration>

Microsoft. (2025). *¿Qué es SQL Server?* Microsoft Learn. <https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver16>

Microsoft. (2025). *¿Qué es SQL Server Management Studio (SSMS)?* Microsoft Learn.

<https://learn.microsoft.com/es-es/sql/ssms/sql-server-management-studio-ssms>

Mitsubishi Electric. (s.f.). *MOTION module basics iQ-R RD78GH startup* [Manual en PDF].

[https://www.mitsubishielectric.com/fa/service-support/global/e-learning/pdf/spa/4-MOTION Module Basics iQ-R RD78GH Startup na spa-A.pdf](https://www.mitsubishielectric.com/fa/service-support/global/e-learning/pdf/spa/4-MOTION%20Module%20Basics%20iQ-R%20RD78GH%20Startup%20na%20spa-A.pdf)

OCME. (s.f.). *Sistemas de llenado para bebidas* [Catálogo en PDF].

<https://pdf.directindustry.es/pdf/ocme/sistemas-llenado-bebidas/57464-379005.html>

Omron Corporation. (s.f.). *Rotary encoders technical guide* [Guía técnica en PDF].

https://www.ia.omron.com/data_pdf/guide/34/rotary_tg_e_7_2_rotary_connect_cg_e_2_1.pdf

Omron Corporation. (s.f.). *Servo motor selection flow chart* [Guía técnica en PDF].

https://www.ia.omron.com/data_pdf/guide/14/servo_selection_tg_e_1_1_3-13%28further_info%29.pdf

Omron Corporation. (s.f.). *Servomotors / servo drivers* [Página web técnica].

<https://www.ia.omron.com/support/guide/14/introduction.html>

Peña, R. (2022). *Testing Lab: Servicio de pruebas de performance para sistemas basados en web* [Trabajo de suficiencia profesional para optar el título profesional de Ingeniero de Sistemas, Universidad de Lima]. Repositorio Institucional de la Universidad de Lima. <https://hdl.handle.net/20.500.12724/18782>

Rimac, B. (2024). *Diseño de una arquitectura de internet de las cosas (IoT) orientada al monitoreo de la calidad del aire en zonas de impacto ambiental de operaciones mineras a tajo abierto, empleando sensores de bajo costo y comunicación mediante el protocolo LoRaWAN* [Tesis de licenciatura, Universidad Peruana de Ciencias Aplicadas]. Repositorio Académico de la Universidad Peruana de Ciencias Aplicadas. <http://hdl.handle.net/10757/683406>

SanDoRobotics. (s.f.). *Tornillo trapezoidal 8 mm - 400 mm* [Ficha técnica en línea].
<https://sandorobotics.com.mx/producto/hr0642c/>

Siemens AG. (2015). *Instrucciones de servicio de SINAMICS V90 Pulse Train, USS/Modbus, SIMOTICS S-1FL6* (Versión 07/2015) [Manual técnico, ID de artículo 109479012].
<https://support.industry.siemens.com/cs/mdm/109479012?c=65418338955&lc=es-EC>

Siemens AG. (2016). *SINAMICS V90 PN 200V/400V OPI: Instrucciones de servicio* (Versión 04/2016, A5E37208872) [Manual técnico].
https://cache.industry.siemens.com/dl/files/880/109737880/att_884437/v1/SINAMICS_V90_PN_200V_400V_OPI_es-ES_es-ES.pdf

Siemens AG. (2017). *SINAMICS V90 PN con motor SIMOTICS S-1FL6: Puesta en marcha rápida* [Manual técnico].

https://cache.industry.siemens.com/dl/files/476/109748476/att_923336/v1/V90_PN_1FL6_getting_started_0417_es-ES.pdf

Siemens AG. (2018). *SINAMICS V90: Ayuda online de SINAMICS V-ASSISTANT* [Manual técnico].

https://cache.industry.siemens.com/dl/files/408/109762408/att_968148/v1/OnlineHelp_V90_es_ES.pdf

Siemens AG. (2020). *Speed control and position control between V90 PN and S7-1200 via HSP* [Manual técnico].

<https://support.industry.siemens.com/cs/document/109780752/speed-control-and-position-control-between-v90-pn-and-s7-1200-via-hsp?dti=0&dl=en&lc=es-PY>

Siemens AG. (2021). *SIMATIC S7-1500 CPU 1511C-1 PN: Manual de producto* (A5E40898593-AB) [Manual técnico]. Siemens AG, Digital Industries.

https://cache.industry.siemens.com/dl/files/843/109752843/att_936132/v1/s71500_cpu1511c_1_pn_dtc_manual_es-ES_es-ES.pdf

Siemens AG. (2021). *STEP 7 / WinCC V17: Manual de funciones* [Manual técnico].

https://cache.industry.siemens.com/dl/files/671/109798671/att_1070873/v1/STEP_7_WinCC_V17_esES_es-ES.pdf

Siemens AG. (2021). *S7-1500/S7-1500T: Sinopsis de Motion Control V6.0, STEP 7 V17 o superior* [Manual de funciones].

https://cache.industry.siemens.com/dl/files/848/109781848/att_1067091/v1/s71500s71500t_motion_control_overview_function_manual_es-ES_es-ES.pdf

Siemens AG. (2022). *PROFINET con STEP 7: Manual de funciones* (Ed. 11/2022, A5E03444489-AM). Siemens AG, Digital Industries.

https://cache.industry.siemens.com/dl/files/856/49948856/att_898975/v1/profinet_step7_v18_function_manual_es-ES_es-ES.pdf

Siemens AG. (s.f.). *Instrucciones de servicio SINAMICS V90: Interfaz PROFINET (PN)*.

Recuperado de <https://www.manualslib.es/manual/311895/Siemens-Sinamics-V90.html?page=203#manual>

Siemens AG. (s.f.). *Power monitoring & control*.

<https://www.siemens.com/us/en/products/energy/ruselectric/products/power-monitoring-control.html>

Siemens AG. (s.f.). *SINAMICS V90: Sistema de accionamiento servo optimizado en rendimiento y fácil de operar*.

<https://www.siemens.com/global/en/products/drives/sinamics/low-voltage-converters/servo-converter/sinamics-v90.html>

SYNHO SAC. (2024). *Demo transportador de botellas*.

Ultatek. (2020). *Cómo lograr la integración de indicadores clave de rendimiento en la industria de alimentos y bebidas*. <https://ultatek.com/como-lograr-la-integracion-de-indicadores-clave-de-rendimiento-en-la-industria-de-alimentos-y-bebidas/>

Anexo A Matriz de consistencia

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA LOCAL Y REMOTO PARA EL MONITOREO Y ALMACENAMIENTO DE VARIABLES RELACIONADAS A SERVOMOTORES Y KPIs EN EL PROCESO DE LLENADO DE BOTELLAS					
PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLE	INDICADORES	METODOLOGÍA
PROBLEMA GENERAL	OBJETIVOS GENERAL	HIPÓTESIS GENERAL	V. INDEPENDIENTE Sistema local y remoto.	1. Implementar un sistema local de monitoreo mediante una interfaz desarrollada en WinCC Professional.	ENFOQUE Cuantitativo
¿Cómo diseñar e implementar un sistema local y remoto que permita el monitoreo y almacenamiento de variables asociadas a servomotores y KPIs en el proceso de llenado de botellas?	Diseñar e implementar un sistema local y remoto para el monitoreo y almacenamiento de variables asociadas a servomotores y KPIs en el proceso de llenado de botellas.	Es posible la implementación de un sistema local y remoto para el monitoreo y almacenamiento de variables relacionadas a servomotores y KPIs en el proceso de llenado de botellas.		2. Implementar un sistema local de almacenamiento utilizando SQL Server.	
				3. Implementar un sistema remoto de monitoreo mediante un panel visual (dashboard) en Grafana Cloud.	
				4. Implementar un sistema remoto de almacenamiento en la nube mediante InfluxDB Cloud.	
problema específicos	objetivos específicos	hipótesis específicas	V. DEPENDIENTE Variables dinámicas de los servomotores	1.1 Velocidad (lineal)	TIPO Aplicada ALCANCE Descriptivo DISEÑO: Experimental POBLACIÓN Y MUESTRA POBLACIÓN: Equipos de monitoreo y almacenamiento de variables de procesos industriales. MUESTRA: El prototipo de llenado de botellas, desarrollado por el grupo de investigación TAS de la Facultad de Ingeniería Electrónica y Eléctrica (FIEE) de la Universidad Nacional Mavor de San Marcos.
a. ¿cómo implementar un sistema local de monitoreo de las variables dinámicas de los servomotores, a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas?	a. implementar un sistema local de monitoreo de las variables dinámicas de los servomotores a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas.	a. es posible implementar un sistema local de monitoreo de las variables dinámicas de los servomotores, a través de una interfaz desarrollada en WinCC Professional, en el proceso de llenado de botellas.		1.2 Posición (lineal)	
b. ¿cómo implementar un sistema local de almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas?	b. implementar un sistema local para el almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas.	b. es posible implementar un sistema local de almacenamiento de las variables dinámicas de los servomotores en SQL Server, en el proceso de llenado de botellas.		1.3 Torque (momento de fuerza)	
c. ¿cómo implementar un sistema remoto de monitoreo de KPIs, mediante un panel visual desarrollado en Grafana Cloud, en el proceso de llenado de botellas?	c. implementar un sistema remoto de monitoreo de KPIs, mediante un panel visual desarrollado en Grafana Cloud, en el proceso de llenado de botellas.	c. es posible implementar un sistema remoto de monitoreo de KPIs, mediante un dashboard en Grafana Cloud, en el proceso de llenado de botellas.	KPI's (Indicadores Claves de Rendimiento)	2.1 Tasa de utilización de capacidad.	
				2.2 Tasa de calidad.	
				2.3 Tasa de disponibilidad.	
				2.4 Productividad.	
d. ¿cómo implementar un sistema remoto de almacenamiento de las variables necesarias para el cálculo de KPIs en InfluxDB Cloud, en el proceso de llenado de botellas?	d. implementar un sistema remoto para el almacenamiento de las variables necesarias para el cálculo de KPIs, utilizando InfluxDB Cloud, en el proceso de llenado de botellas.	d. es posible implementar un sistema remoto de almacenamiento de las variables necesarias para el cálculo de KPIs en InfluxDB Cloud, en el proceso de llenado de botellas.			

Anexo B Líneas de código del Script *CrearDB*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
Provider=MSDASQL; DSN=" & DSN & ";	Cadena de conexión ODBC. Usa un DSN (Data Source Name) para definir a qué base de datos se conectará.
SELECT COUNT(*) FROM sys.databases WHERE name = 'NombreDB'	Consulta SQL que verifica si ya existe una base de datos con el nombre especificado.
CREATE DATABASE NombreDB	Comando SQL para crear una nueva base de datos con el nombre indicado.
connect.Close	Cierra la conexión a la base de datos una vez finalizadas las operaciones.
Set connect= Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo C Líneas de código del Script *CrearTabla*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
Provider=MSDASQL;Initial Catalog=master;DSN=" & DSN & "	Abre la conexión a la base de datos master usando una cadena de conexión (DSN) específica. Esto permite interactuar con el servidor de bases de datos.
SELECT COUNT(*) FROM sys.databases WHERE name = 'NombreDB'	Consulta SQL que verifica si ya existe una base de datos con el nombre especificado.
connect.Open "Provider=MSDASQL;Initial Catalog=NombreDB;DSN=..."	Abre una nueva conexión, esta vez a la base de datos NombreDB, usando la cadena de conexión adecuada.
SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'NombreTabla'	Ejecuta una consulta SQL para verificar si una tabla llamada NombreTabla ya existe en la base de datos.
CREATE TABLE NombreTabla (Fecha_hora DATETIME, ColumnaX FLOAT, ...)	Construye y ejecuta una sentencia SQL para crear una nueva tabla llamada NombreTabla con las columnas definidas.
Set connect= Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo D Líneas de código del Script *BorrarDB*

Línea de código	Descripción
Set conn = CreateObject("ADODB.Connection")	Crea una conexión a una base de datos usando ADO y la asigna a la variable conn.
Set rst = CreateObject("ADODB.Recordset")	Crea un objeto Recordset que permite almacenar y manipular el resultado de una consulta SQL.
conn.Open "Provider=MSDASQL;DSN=..."	Establece la conexión con la base de datos utilizando un origen de datos definido (DSN) a través del proveedor ODBC.
DROP DATABASE NombreDB10	Comando SQL que elimina por completo la base de datos con el nombre indicado. Es irreversible y borra todo su contenido.
Set conn = Nothing	Asigna Nothing al objeto conn, lo que indica que ya no se necesita el objeto de conexión y libera su referencia.
Set rs = Nothing	Asigna Nothing al objeto rs, que representa el recordset. Esto también libera su referencia para que el sistema pueda manejar la memoria de manera eficiente.

Anexo E Líneas de código del Script *BorrarTabla*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
connect.Open "Provider=MSDASQL;Initial Catalog=master;DSN=" & DSN & ";"	Se establece la conexión al servidor usando el DSN proporcionado, con la base de datos 'master'.
SELECT COUNT(*) FROM sys.databases WHERE name = 'NombreDB10'	Se ejecuta la consulta para verificar si la base de datos con el nombre 'NombreDB10' existe.
connect.Open "Provider=MSDASQL;Initial Catalog=" & SmartTags("NombreDB10") & ";DSN=" & DSN & ";"	Se conecta a la base de datos 'NombreDB10' usando el DSN proporcionado.
SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'NombreTabla10'	Se ejecuta una consulta para verificar si la tabla 'NombreTabla10' existe en la base de datos.
SQLDropTable = "DROP TABLE " & SmartTags("NombreTabla10")	Se prepara la sentencia SQL para eliminar la tabla 'NombreTabla10'.
connect.Execute(SQLDropTable)	Se ejecuta la sentencia SQL para eliminar la tabla 'NombreTabla10' de la base de datos.
Set connect= Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo F Líneas de código del Script *InsertarDataMAN*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
connect.Open "Provider=MSDASQL;Initial Catalog=" & DBName & ";DSN=" & DSN & ";"	Establece una conexión a la base de datos especificada por DBName, usando un origen de datos (DSN) preconfigurado. Es esencial para ejecutar consultas sobre esa base de datos.
SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = " & TableName & "	Consulta la estructura de la tabla, es decir, obtiene los nombres de las columnas de la tabla indicada por TableName. Se usa para construir dinámicamente la sentencia de inserción.
SQLinsert = "INSERT INTO " & TableName & " (" & columnList & ") VALUES (" & valueList & ")"	Construye la sentencia SQL de inserción de datos. columnList contiene los nombres de las columnas donde se insertarán datos, y valueList contiene los valores que se van a insertar.
Set connect= Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo G Líneas de código del Script *InsertarDataAUTO*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
connect.Open "Provider=MSDASQL;Initial Catalog=" & DBName & ";DSN=" & DSN & ";"	Establece una conexión a la base de datos especificada por DBName, usando un origen de datos (DSN) preconfigurado. Es esencial para ejecutar consultas sobre esa base de datos.
SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = "" & TableName & ""	Consulta la estructura de la tabla, es decir, obtiene los nombres de las columnas de la tabla indicada por TableName. Se usa para construir dinámicamente la sentencia de inserción.
SQLinsert = "INSERT INTO " & TableName & " (" & columnList & ") VALUES (" & valueList & ")"	Construye la sentencia SQL de inserción de datos. columnList contiene los nombres de las columnas donde se insertarán datos, y valueList contiene los valores que se van a insertar.
Set connect= Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo H Líneas de código del Script *MostrarTabla*

Línea de código	Descripción
Set conn = CreateObject("ADODB.Connection")	Crea una conexión a una base de datos usando ADO y la asigna a la variable conn.
conn.Open "Provider=SQLOLEDB;Data Source=LAPTOP-SJQG2EJP;Initial Catalog=" & DBName & ";Integrated Security=SSPI;"	Establece una conexión a la base de datos utilizando el proveedor SQLOLEDB y parámetros como el nombre del servidor, base de datos y autenticación integrada de Windows.
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = "" & tableName & ""	Consulta para verificar si la tabla especificada (tableName) existe en la base de datos.
SELECT MIN(fecha_hora) AS FechaInicio, MAX(fecha_hora) AS FechaFin FROM " & tableName	Recupera la fecha más antigua (FechaInicio) y más reciente (FechaFin) de la columna fecha_hora en la tabla especificada.
SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = "" & tableName & "" AND COLUMN_NAME <> 'fecha_hora' ORDER BY ORDINAL_POSITION	Obtiene los nombres de todas las columnas de la tabla, excluyendo fecha_hora, y las ordena según su posición en la tabla.
SELECT * FROM " & tableName & " ORDER BY [fecha_hora] ASC	Recupera todos los registros de la tabla, ordenados por la columna fecha_hora en orden ascendente.
conn.Close	Cierra la conexión con la base de datos, liberando los recursos asociados a esa conexión.
Set conn = Nothing	Asigna Nothing al objeto conn, lo que indica que ya no se necesita el objeto de conexión y libera su referencia.
Set rs = Nothing	Asigna Nothing al objeto rs, que representa el recordset. Esto también libera su referencia para que el sistema pueda manejar la memoria de manera eficiente.

Anexo I Líneas de código del Script *DBrefres*

Línea de código	Descripción
Set connect = CreateObject("ADODB.Connection")	Crea un objeto de conexión ADO para establecer la comunicación con la base de datos.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
connect.Open "Provider=MSDASQL;DSN=" & DSN & ";"	Abre una conexión con el origen de datos (DSN) usando ODBC mediante el proveedor MSDASQL.
"SELECT name FROM sys.databases WHERE state_desc = 'ONLINE' AND name NOT IN ('master', 'tempdb', 'model', 'msdb');"	Consulta SQL que obtiene los nombres de bases de datos en línea.
connect.Close	Cierra la conexión a la base de datos abierta.
record.Close	Cierra el conjunto de registros (Recordset) abierto.
Set connect = Nothing	Libera el objeto de conexión de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo J Líneas de código del Script *TablaporDBrefresh*

Línea de código	Descripción
Set conn = CreateObject("ADODB.Connection")	Crea una conexión a una base de datos usando ADO y la asigna a la variable conn.
Set record = CreateObject("ADODB.Recordset")	Crea un objeto Recordset ADO para ejecutar consultas y almacenar los resultados.
conn.Open "Provider=MSDASQL;DSN=" & DSN & ";"	Abre una conexión con el origen de datos (DSN) usando ODBC mediante el proveedor MSDASQL.
"SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE';"	Devuelve los nombres de todas las tablas físicas (no vistas) de la base de datos actual.
Set record = conn.Execute(SQLTablas)	Ejecuta la consulta SQL (SQLTablas) sobre la conexión conn y guarda el resultado en el objeto record.
conn.Close	Cierra la conexión a la base de datos para liberar recursos.
Set conn = Nothing	Libera el objeto de conexión (conn) de la memoria.
Set record = Nothing	Libera el objeto Recordset de la memoria.

Anexo K Líneas de código del Script *ColumnaRefreshDBdinamico*

Línea de código	Descripción
Set conn = CreateObject("ADODB.Connection")	Crea una conexión a una base de datos usando ADO y la asigna a la variable conn.
Set rs = CreateObject("ADODB.Recordset")	Crea un objeto Recordset que se usará para almacenar los resultados de una consulta SQL.
conn.Open "Provider=MSDASQL;DSN=" & DSN & ";"	Abre una conexión ODBC al origen de datos especificado por el DSN usando el proveedor MSDASQL.
"SELECT name FROM sys.databases WHERE state_desc = 'ONLINE' AND name NOT IN ('master', 'tempdb', 'model', 'msdb');" & "	Consulta SQL que obtiene los nombres de bases de datos en línea, excluyendo las del sistema.
conn.Open "Provider=SQLOLEDB;Data Source=LAPTOP-SJQG2EJP;Initial Catalog=" & dbName & ";Integrated Security=SSPI;"	Abre una conexión SQL Server utilizando el proveedor SQLOLEDB y se conecta al servidor LAPTOP-SJQG2EJP con base de datos especificada por dbName. Usa la seguridad integrada de Windows.
"SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = " & tableName & "" & ""	Consulta SQL que obtiene los nombres de las columnas de la tabla especificada por tableName dentro de la base de datos actual, accediendo al esquema de la base de datos.
Set rs = conn.Execute(sqlQuery)	Ejecuta la consulta SQL y almacena los resultados en el objeto Recordset rs.
Conn.Close	Cierra la conexión con la base de datos.
Set conn = Nothing	Libera el objeto de conexión de la memoria.
Set rs = Nothing	Libera el objeto Recordset de la memoria.