UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

"DISEÑO E IMPLEMENTACIÓN DE UN PROCESO INDUSTRIAL ACCIONADO MEDIANTE GESTOS USANDO PLC, HMI Y UN ORDENADOR MONOPLACA"

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Electrónica

AUTOR: ALBUJA PAREDES JOAO MOISÉS

FALCONI CARO STEFANO ISAAC

TUTOR: VÍCTOR DAVID LARCO TORRES

Guayaquil – Ecuador

2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Joao Moises Albuja Paredes con documento de identificación N° 0704543834 y Stefano Issac Falconi Caro N° 1721353314, manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 12 de febrero del año 2025.

Atentamente,

Joao Moises Albuja Paredes
0704543834

Stefano Isaac Falconi Caro
1721353314

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITECNICA SALESIANA

Nosotros, Joao Moisés Albuja Paredes con documento de identificación N° 0704543834 y, Stefano Issac Falconi Caro con documento de identificación N° 1721353314, manifestamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "DISEÑO E IMPLEMENTACIÓN DE UN PROCESO INDUSTRIAL ACCIONADO MEDIANTE GESTOS USANDO PLC, HMI Y UN ORDENADOR MONOPLACA", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica y Automatización, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 12 de febrero del 2025

Atentamente,

Joao Moises Albuja Paredes

0704543834

Stefano Isaac Falconi Caro

1721353314

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Víctor David Larco Torres con documento de identificación N° 0923270136, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: "DISEÑO E IMPLEMENTACIÓN DE UN PROCESO INDUSTRIAL ACCIONADO MEDIANTE GESTOS USANDO PLC, HMI Y UN ORDENADOR MONOPLACA", realizado por Joao Moisés Albuja Paredes con documento de identificación N° 0704543834 y, Stefano Issac Falconí Caro con documento de identificación N° 1721353314, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 12 de febrero del 2025

Atentamente,

Ing. Víctor David Larco Torres
0923270136

Agradecimiento

Agradezco, a Dios por bendecirme al darme la fortaleza y sabiduría para superar cada desafío en este camino. A mis padres, por su constante apoyo y por ser mi inspiración en cada momento. A mi hijo Patrocleo por ser mi mayor motivo de esfuerzo y superación. A Leslie, mi compañera de vida, por su amor, paciencia y apoyo incondicional en cada paso de este proceso profesional.

Agradezco también a los docentes de la Universidad Politécnica Salesiana, quienes con su conocimiento me guiaron en mi trayecto universitario. En especial, extiendo mi gratitud a los ingenieros Emmanuel Torres y Geovanny García, cuyo apoyo y guía fueron fundamentales para la concreción de este trabajo. A mi tutor, a quien también considero en este proceso por su disposición y orientación a lo largo de la tesis.

Joao Albuja P.

Yo Stefano Falconi, estoy muy agradecido en esta etapa de formación profesional principalmente con Dios, con mis padres, mis familiares más allegados y quienes más me apoyaron durante la carrera y mis profesores. Pese a los días largos de estudio y días difíciles donde no obtenemos la nota que esperábamos por muchas veces no estudiar lo suficiente, agradezco que esto haya sido un motivo más para cogerle el amor respectivo a los estudios, y por aquellos profesores que nos hicieron sacar lo mejor de nosotros mismos con cada desafío y aliento para no perder la motivación ni la disciplina en este lindo tramo de preparación hacia nuestro grado. Quiero dar una mención de gracias especial al ingeniero Emmanuel Torres por haber acompañado todo este proceso desde el tercer semestre y darnos un gran apoyo durante nuestro proceso de tesis. De igual forma, deseo agradecer al ingeniero Geovanny García, por brindarnos un gran soporte durante nuestra elaboración de la tesis. Finalmente, agradecer a mi tutor por su guía durante este proceso y brindarnos los materiales utilizados para su implementación.

Stefano Falconi C.

Dedicatoria

Dedico este trabajo, primero, a Dios, por iluminar mi camino y darme la fortaleza para enfrentar cada reto que la vida me ha puesto. A mis padres, quienes, a pesar de la distancia, nunca dejaron de estar a mi lado. Su amor incondicional, sus palabras de aliento y sus sacrificios han sido el pilar sobre el cual he construido este logro. Cada paso que he dado ha sido gracias a su apoyo inquebrantable, y este triunfo es tan suyo como mío.

Joao Albuja P.

Dedico esta realización como estudiante principalmente a Dios, y a mis padres, quienes fueron el pilar de valores, disciplina, constancia y motivación constante para poder nunca bajar los brazos y poder llegar hasta este punto de mi vida. También se lo dedico a quien un día quería ser ingeniero y pudo lograrlo a punta de esfuerzo y disciplina.

Stefano Falconi C.

Resumen

AÑO	ALUMNOS	DIRECTOR DE PROYECTO	TEMA DE PROYECTO DE TITULACIÓN
2025	JOAO MOISÉS ALBUJA PAREDES STEFANO ISAAC FALCONI CARO	ING. VÍCTOR DAVID LARCO TORRES	"DISEÑO E IMPLEMENTACIÓN DE UN PROCESO INDUSTRIAL ACCIONADO MEDIANTE GESTOS USANDO PLC, HMI Y UN ORDENADOR MONOPLACA"

El proyecto tiene como objetivo centrarse en el diseño e implementación de un proceso industrial que utiliza el reconocimiento de gestos para accionar dispositivos mediante la integración de tecnologías como PLC, HMI y un ordenador monoplaca. El documento establece la relevancia de la inteligencia y la visión artificial en la optimización de procesos industriales, en un contexto global donde países lideres cómo Estados Unidos que contiene fuertes inversiones en automatización industrial, Alemania que es pionera en la integración de sistemas ciberfísicos, han invertido en la infraestructura y desarrollo de estas tecnologías. Además, la necesidad urgente de enfoques innovadores para interactuar con equipos industriales se enfatiza con nuestra solución, que sirve como un método alternativo de control directo para usuarios con discapacidades o limitaciones físicas.

El proyecto tiene un objetivo, el cual es desarrollar un sistema que traduzca gestos en comandos y utilice imágenes para construir un conjunto de datos para entrenar modelos de aprendizaje automático para identificar estos gestos y establecer una interfaz de comunicación entre la computadora de plaza única y el PLC. Al utilizar acceso especifico y hardware rápido, como Raspberry Pi, para este objetivo, junto con el uso de herramientas de control y programación industrial de hoy en día, se logra una integración fluida y segura en entornos de automatización.

La metodología propuesta está organizada en diferentes fases que van desde la investigación gestual y selección hasta la validación final del sistema en un escenario piloto. Primero, se completa una fase de captura ya notación de datos, seguida del desarrollo y ajuste fino de los algoritmos de procesamiento, detección y segmentación gestual. Luego, el sistema se integra mediante la configuración de protocolos de comunicación industrial, se realizan pruebas exhaustivas para maximizar la precisión y los tiempos de respuestas, y se lleva a cabo la validación en un entorno controlado, demostrando la viabilidad y el potencial de impacto de esta innovación en el campo de la automatización laboral.

Abstract

YEAR	STUDENTS	PRJ. DIRECTOR	SUBJECT
2025	JOAO MOISÉS ALBUJA PAREDES STEFANO ISAAC FALCONI CARO	ING. VÍCTOR DAVID LARCO TORRES	"Desing and Implementation of an Industrial Process Controlled by Gestures Using PLC, HMI, and a Single-Board Computer"

The project aims to focus on the design and implementation of an industrial process that uses gesture recognition to actuate devices by integrating technologies such as PLC, HMI and a single board computer. The paper establishes the relevance of artificial intelligence and vision in optimizing industrial processes, in a global context where leading countries such as the United States, which contains strong investments in industrial automation, and Germany, which is a pioneer in the integration of cyber-physical systems, have invested in the infrastructure and development of these technologies. Furthermore, the urgent need for innovative approaches to interact with industrial equipment is emphasized by our solution, which serves as an alternative method of direct control for users with disabilities or physical limitations.

The project has one goal, which is to develop a system that translates gestures into commands and uses images to build a dataset to train machine learning models to identify these gestures and establish a communication interface between the single-board computer and the PLC. By using dedicated access and fast hardware, such as Raspberry Pi, for this purpose, together with the use of today's industrial programming and control tools, a smooth and secure integration into automation environments is achieved.

The proposed methodology is organized in different phases ranging from gestural research and selection to the final validation of the system in a pilot scenario. First, a data capture and annotation phase is completed, followed by the development and fine-tuning of the gesture processing, detection and segmentation algorithms. Then, the system is integrated by configuring industrial communication protocols, exhaustive tests are performed to maximize accuracy and response times, and finally validation is carried out in a controlled environment, demonstrating the feasibility and impact potential of this innovation in the field of labor automation.

Índice de Contenido

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

Los avances en la Industria han hecho posible integrar diversas tecnologías de vanguardia en el ciclo de producción de fábricas inteligente, permitiendo automatizar procesos con Inteligencia Artificial (IA), aprendizaje automático y visión por computadora. Esto ha cambiado el panorama de la interacción humano-máquina y ha traído enormes eficiencias operativas a todas las industrias productivas, (Wilcher, 2024). Basado en esto, el estudio actual introduce el desarrollo de un sistema de control industrial operado por gestos, empleando un PLC (Controlador Lógico Programable), HMI (Interfaz Hombre-Máquina) y una computadora de placa única (Raspberry Pi).

Históricamente, los humanos habrían tenido que interactuar con dispositivos físicos como botones y pantallas táctiles para ejecutar ciertas tareas en sistemas de automatización tradicionales. Sin embargo, tales interfaces pueden sufrir limitaciones de accesibilidad y ergonomía en el sector industrial, donde tocar el equipo puede llevar a problemas directos de seguridad e higiene. Además, excluir a personas con movilidad reducida o desafíos de comunicación verbal subraya la necesidad de crear alternativas inclusivas. (Industria4punto0, 2024)

Este trabajo explica el proceso de construcción de un Modelo de Reconocimiento de Gestos, a través de visión por Computadora y Aprendizaje Automático, que permite reconocer movimientos de las manos y traducirlos en comandos procesados por un PLC para llevar a cabo tareas industriales. Para ello, se utilizará un conjunto de herramientas tecnológicas como cámaras con soportes para OpenCV, algoritmos de procesamiento de imágenes y un sistema de comunicación basado en PROFINET para la integración eficiente del modelo de IA con el PLC Siemens S7-1500 (Cerrato Nieto, s.f.).

Se concluye que, el uso de reconocimiento de gestos para sistemas de control es parte del concepto revolucionario de la automatización industrial, pero también permite ambientes de trabajo más seguros y mejores. Las secciones dentro del documento abordan la declaración del problema, delimitación, objetivos y alce del trabajo.

PROBLEMA

Antecedentes

La integración de sistemas de control industrial mediante gestos ha ganado relevancia en los últimos años, especialmente en aplicaciones que buscan mejorar la accesibilidad y la eficiencia operativa. El uso de controladores lógicos programables (PLC), interfaces hombre-máquina (HMI) y ordenadores monoplaca permite desarrollar soluciones innovadoras en este entorno. Se afirma que el monitoreo y control del equipo para operación industrial a través de los protocolos de software se denomina automatización industrial; se ha hecho posible reducir errores mejorando la eficiencia (Maisvch, 2024).

Dentro de la Universidad Politécnica Salesiana, se ha detectado la necesidad de aplicar sistemas que permitan a estudiantes y profesores interactuar con los módulos de enseñanza de automatización. En particular, aquellos con discapacidades físicas tienen dificultades al interactuar con controles tradicionales. La combinación de estas tecnologías es viable para superar estas barreras: con visión por computadora, PLC y HMI. Un PLC es un sistema informático industrial diseñado para gestionar procesos de fabricación, como líneas de producción o robots, y se caracteriza por su alta fiabilidad, facilidad de programación y capacidades de diagnóstico de fallos en los procesos lo que convierte que estos sistemas sean ideales para parámetros educativos y de capacitación, lo que convierte a estos sistemas en ideales para parámetros educativos y de capacitación (NorthWind Technical Services, 2022).

A nivel mundial, diferentes continentes como Europa y América del Norte han emprendido iniciativas que involucran la integración de tecnologías de control por gestos en el sector industrial (Erick, 2024). Por ejemplo, las empresas fabricantes alemanas están utilizando interfaces basadas en gestos para mejorar la seguridad y eficiencia en las líneas de producción. La página Phoeniz Contact que es una empresa global de tecnología y automatización con sede en Alemania dice que su interfaz hombre- maquina (HMI) y sus PC industriales han derivado máxima escalabilidad para el concepto de comando de soluciones de automatización, permitiendo una operación eficiente y monitoreo del estado de la plata (Phoenix Contact, 2023).

Los sistemas de visión por computadora en los sistemas de control de maquinaria en la industria automotriz de Japón se utilizan para emplear gestos y así evitar el contacto, minimizando así los riesgos laborables. Un modelo de trabajo en la industria para PLC, HMI, SCADA (Alex,

2023). Explica como el control automatizado de PLC en una planta de producción es útil para aplicación de estas tecnologías en un entorno industrial.

Los ingenieros de soluciones en los Estados Unidos han utilizado computadoras de placa únicas combinadas con sistemas de control industrial para ayudar a operadores con discapacidades a interactuar con controles de vehículos. Existe una máquina que muestra como un sistema de control basado en PLC y la monitorización HMI implementados para que realice cortes de bloques de esponja horizontal, utilizada en la seguridad del entorno laboral y procesos de automatización industrial, como la fabricación de colchones, pueden ser una opción interesante (Ernesto, Quezada, Vargas, & Calderon, 2020).

Mediante el aprovechamiento de mecanismo de control por gestos, los sistemas de control por gestos optimizan los sistemas industriales eliminando la necesidad de interfaces físicas convencionales, mejorando la accesibilidad y l facilidad de uso. El campo de la automatización industrial abarca industrias múltiples, como la fabricación de maquinaria, la tecnología de la información y la inteligencia artificial, y depende en gran medida de equipos como los PLC y las HMI (Maisvch, 2024).

Además, los sistemas de control industrial utilizan computadoras de placa única, como la Raspberry Pi, junto con varios dispositivos loT y sensores para crear soluciones flexibles. Proporcionan suficiente poder de procesamiento para tareas de visión por computadora y pueden integrarse bien con PLC y HMI. Existe varios estudios en dónde demuestra que los PLC pueden trabajar junto con otros equipos de automatización para crear sistemas de control completos, permitiendo que la tecnología más reciente sea instalada en establecimientos industriales (NorthWind Technical Services , 2023).

En el campo educativo, la implementación de estas tecnologías permite a las instrucciones proporcionar experiencias de aprendizaje más inclusivas e impactantes. La Universidad Politécnica Salesiana no solo respondería a las necesidades de los estudiantes con discapacidades, sino que también estaría acorde con la tendencia global de sistemas automatizados industriales gracias a este sistema controlado por gestos. Como destaca Phoenix Contact, sus HMI y PC industriales pueden montarse de manera flexible en equipos de campo (como se discute aquí) y utilizarse para la observación junto a la máquina, permitiendo integrarlos en entornos educativos (Phoenix Contact, 2023).

Las interfaces basadas en gestos se han convertido en el foco de investigación y desarrollo en línea con la creciente necesidad de seguridad industrial y eficiencia en la fabricación. Al evitar tocar superficies de control con las manos, no solo se mejora la ergonomía y se reduce la fatiga de operador, sino que también se disminuye la posibilidad de contaminación en entornos sensibles. (Carrera, 2024).

Desde América Latina, países como México o Brasil han mostrado interés en las tecnologías de automatización basadas en la visión artificial para mejorar sus procesos industriales. En estos países, la industria fabricante ha comenzado a implementar sistemas de control sin contacto.

Con base en este contexto, esta investigación tiene como objetivo crear un sistema de control industrial basado en gestos que integran tecnologías como la visión artificial, PLC, y HMI para ofrecer una solución innovadora e inclusiva para enseñanza de la automatización en la Universidad Politécnica Salesiana. Esto ayudará a las plantas industriales con estudiantes que aprenden mediante gestos corporales en un sistema informático que monitorizará sus movimientos, para que puedan usar equipos industriales como si aprendieran naturalmente a usarlos sin tener la habilidad de moverse por dispositivos de control tradicionales. Además, la implementación de estas tecnologías conduce a la formación de profesionales capaces de operar en entornos modernos de automatización y está en línea con las tendencias de la industria 4.0, indicando una tendencia hacia un aprendizaje más accesible y efectivo. Esta investigación debería afectar el entorno académico y puede tener aplicaciones en la red industrial con el objetivo de implementar procesos de control y procesos laborales para mejorar la seguridad ocupacional.

IMPORTANCIA Y ALCANCES

Las nuevas tecnologías emergentes como la realidad aumentada/realidad virtual, los sistemas sensoriales y los sistemas robóticos aumentan la competitividad y eficiencia de las actividades industriales. El concepto clásico de interacción táctil entre operadores y maquinas puede ser reemplazado por sistemas de reconocimiento de gestos, reduciendo el contacto y aumentado la accesibilidad para realizar tareas. Este enfoque es consistente con las tendencias que hemos visto evolucionar en lo que se ha llegado a conocer como la industria 4.0 donde el objetivo es crear maquinas que procesen información visual de manera inteligente como lo haría un humano. Utilizando cámaras distribuidas especialmente que funcionan como ojos industriales virtuales, la captura de datos alimenta al modelo de inteligencia artificial. Luego, estos datos son procesados por algoritmos avanzados y modelos de aprendizaje automático que "detectan ciertas tendencias, objetos y situaciones" (EMI SUITE , s.f.).

En cuanto a la lista de tareas del proyecto, el campo de aplicación del proyecto abre un área amplia que podría generalizarse por sector de actividad industrial, lo que nos permite adaptar y desplegar los sistemas de automatización en diferentes contextos operativos. La migración de aspectos comunes como el PLC, el HMI y el ordenador de placa única allanan el camino para innovaciones personalizadas y futuras con respecto a la interacción humana con la computadora automatizada. Esto es una contribución a la transformación digital de la industria, donde esta plataforma tecnológica puede aplicarse en sectores tan variados como la manufactura, la logística, el control de calidad, etc. Los PLC y los HMI se complementan entre sí, cada uno aporta sus únicos beneficios al sistema en su conjunto, resultando en última instancia en una mejor eficiencia y control en los procesos industriales (Mais VCH, 2024)

DELIMITACIÓN DEL PROBLEMA

Temporal

El proyecto y todas las pruebas correspondientes al mismo se llevaron a cabo durante el periodo de 6 meses del proyecto.

Geográfica

A lo largo del mundo, se pueden observar patrones de implementación de IA y visión por computadora a la industria en áreas como Europa y Asia, donde la industrialización de la tecnología inteligente se ha propagado rápidamente. Cómo tal se afirma que, mientras que el uso de la IA en la manufactura ha arrasado en estas regiones, el crecimiento ha sido lento en América Latina. No obstante, países como Ecuador han visto un aumento en la automatización industrial, principalmente en el entorno académico y en laboratorios de innovación. En esta línea, en consonancia con la modernización de la región, este trabajo, desarrollado en la Universidad Politécnica Salesiana de Guayaquil, propone innovadoras automatizaciones basadas en el reconocimiento de gestos. (ecomex360, 2020)

Académica

La inteligencia artificial y el aprendizaje automático en el campo de la automatización industrial con tecnologías nuevas y emergentes. Como resultado, se han identificado varias aplicaciones de aprendizaje automático e IoT para mejorar los procesos de producción en la manufactura inteligente (Fazio, 2024). En estudios más recientes, también se ha indicado que la alta escalabilidad y la falta de infraestructura tecnológica en algunos entornos son uno de los desafíos más significativos de la IA en la industria, por lo que se requiere implementar nuevas soluciones que permitan integrar hardware especializado, como PLC y HMI con algoritmos de reconocimiento de imágenes (Ghommam, 2022). El estudio actual responde a este campo investigando un enfoque alternativo que fusiona la visión por computadora con el control industrial por gestos, permitiendo que los no especializados en el sistema accedan a estos sistemas, incluso para personas con limitaciones físicas o técnicas.

OBJETIVOS

Objetivo General

Implementar y controlar un Proceso Industrial Accionado mediante comandos gestuales usando PLC, HMI y un ordenador monoplaca.

Objetivos Específicos

- Recolectar imágenes de los gestos para elaborar el Dataset.
- Entrenar un modelo de Machine Learning para el reconocimiento y la clasificación de los gestos.
- Escribir un código que se ejecute en una computadora monoplaca y se comunique con el PLC.

REVISIÓN DE LA LITERATURA

La automatización industrial ha tenido un cambio a menudo, incorporando tecnologías que se logran promover en un acceso más amplio. Este proyecto incluye tecnologías de Hardware y Software de vanguardia para obtener un sistema de control industrial fundamentado en gestos.

Esta revisión bibliográfica demuestra los conceptos, instrumentos y las tecnologías principales que logran respaldar el avance de este sistema, logrando cubrir desde el reconocimiento de gestos hasta llegar con el PLC y HMI, ya que también se incluye las herramientas de comunicación y programaciones necesarias para la realización de este proyecto.

Reconocimiento de gestos en sistemas industriales

Algoritmos de reconocimientos de gestos.

Lo que facilita el reconocimiento de gestos es la tecnología que permite interpretar movimientos corporales mediante los algoritmos de Visión Artificial y aprendizaje automático. Este proyecto, se utilizaron las librerías Mediapipe y OpenCV, integradas en Python, para identificar gestos específicos que generan señales de control destinadas a maquinas industriales. Estos gestos son detectados en tiempo real mediante una cámara USB conectada a una Raspberry Pi, ya que lo posibilita una interacción sin contacto físico con el sistema de control.

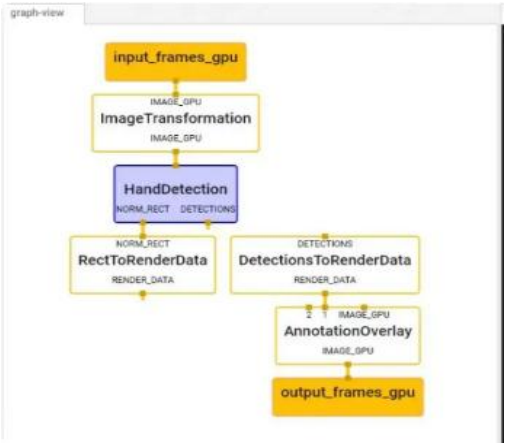
Mediapipe y Opencv como herramientas para la identificación de gestos.

Mediapipe Python se destaca como una herramienta poderosa para desarrollar soluciones de visión computacional, ya que simplifica el uso de las funcionalidades clave de Mediapipe C++ sin necesitar un conocimiento profundo del lenguaje. Este marco, conectado mediante Pybind11, permite realizar inferencias de manera eficiente, esto lo hace una elección perfecta para proyectos que se necesita el procesamiento y reconocimiento de gestos, como el implementado en este trabajo. Su uso logra impulsar el avance de la Inteligencia Artificial aplicada a procesos industriales, ofreciendo una solución práctica y económica para aquellos que buscan integrar Visión computacional en el sistema de control con PLC. (Gemini Developer API, 2024)

La figura 1 presenta un diagrama básico del flujo de visión artificial en Mediapipe. Este proceso empieza con la entrada de imágenes procesados por la GPU, seguido por la detección de manos en el módulo de detección de manos. Posteriormente, los resultados se alistan para lograr su

visualización en el componente AnnotationOverlay. Este flujo permite la identificación y visualización de gestos en tiempo real, facilitando la interacción mediante el uso de visión artificial.

Figura 1
Algoritmos de procesamiento de medios



Nota: Demostración de detección de manos de MediaPipe con la herramienta Visualizador. Fuente: (Viso.ai, 2023)

OpenCV se caracteriza por ser una librería de visión computacional creada por Intel en 1999, bajo la dirección de Gary Bradsky, con su primer lanzamiento en el año 2000. A lo largo del tiempo, ha evolucionado para incorporar una amplia gama de algoritmos de Visión Computacional y aprendizaje automático, y sigue ampliando sus capacidades. Actualmente, OpenCV es compatible con lenguajes de programación con C++, Python y Java, y estas se encuentran disponible en plataformas como Windows, Linux, OS X, Android e IOS. OpenCV-Python es la AOI de OpenCV para Python, que puede combinar las mejores características de OpenCV en C++ con las ventajas del lenguaje Python. (G., s.f.).

Controladores Lógicos Programables (PLC).

Características y ventajas del PLC S7-1500 en la Automatización industrial.

El PLC Siemens S7-1500 se presenta como una solución avanzada y eficiente para la automatización en la industria. Este controlador destaca por su alto rendimiento, con tiempos de procesamiento de órdenes que pueden llegar hasta 1 nanosegundo, lo que permite respuestas rápidas y precisas en los procesos, Su bus de fondo es de alta velocidad y el excelente desempeño en la comunicación PROFINET que logran garantizar tiempos de respuesta reducidos, un factor crucial para alcanzar altos niveles de productividad y calidad en los sistemas industriales. (SIEMENS , 2025)

Una de las principales ventajas del PLC Siemens S7-1500 es su interfaz PROFINET, que proporciona un comportamiento determinístico en el tiempo, asegurando la reproducibilidad y precisión en intervalos de microsegundos. Esto logra convertir en una excelente opción para las aplicaciones que requieren sincronización precisa y ejecución eficiente de los procesos complejos. (SIEMENS, 2025)

Además, este controlador se caracteriza por su durabilidad y adaptabilidad, lo que lo hace adecuando para una amplia gama de aplicaciones industriales. También está diseñado para poder facilitar su integración con otros sistemas de control, simplificando su implementación y mejorando la confiabilidad general del proceso de producción. (SIEMENS, 2025)

La figura 2 muestra un sistema de control automatizando, empleando el PLC Siemens S7-1500. Este controlador es especialmente adecuado para aplicaciones que demandan alta velocidad de procesamiento y una comunicación precisa con otros dispositivos, lo que lo convierte en una opción ideal para entornos industriales complejos. Su integración con la interfaz PROFINET garantiza tiempos de respuesta rápida, optimizando la eficiencia y el control en tiempo real que nos facilita en los procesos de producción. (SIEMENS, 2025)

Figura 2
PLC SIEMENES S7-1500 en entorno de control automatización.

Nota: El PLC Siemens S7-1500 proporciona tiempos de respuesta rápidos gracias a su capacidad de procesamiento de hasta 1 nanosegundo y a su interfaz PROFINET. Fuente: (SIEMENS, 2025)

Comunicación entre PLC y Python

El uso de la biblioteca Snap7 para la integración de PLC Siemens S7-1500 con Python. Snap7 es un conjunto de códigos disponibles para la comunicación multiplataforma Ethernet con los PLC Siemens S7. Es totalmente compatible con las series 300 y 400 y parcialmente compatible con las series 1200, 1500 y otros dispositivos como Simamics y Logo. Está diseñado para funcionar con arquitecturas de 32 y 64 bits y no requiere bibliotecas de terceros, lo que facilita su implementación. Admite una variedad de lenguajes de programación, incluidos Python, C/C++ y .NET. Al utilizar Snap7, es posible conectar y comunicar entre sistemas programados en Python y PLC Siemens, lo que permite un control eficaz de los procesos industriales. (PROGRAMACIÓNSIEMENS.COM, s.f.)

Protocolos de comunicación Modbus y su aplicación en la interacción entre Raspberry Pi y el PLC. El protocolo de comunicación Modbus, ampliamente utilizado en la industria para conectar dispositivos electrónicos industriales, como los PLC, se basa en la arquitectura Maestro /Esclavo o Cliente/Servidor. Ampliamente utilizado en la industria para conectar dispositivos electrónicos industriales, como PLC, se basa en la arquitectura Maestro/Esclavo o Cliente/Servidor. Su popularidad se debe a su naturaleza pública, facilidad de implementación y flexibilidad en la gestión de bloques de datos sin restricciones. Raspberry Pi puede trabajar como un maestro que exige datos o comandos al PLC, que responde como esclavo, en un entorno de control industrial, como la comunicación entre una Raspberry Pi y un PLC, donde

Modbus permite una comunicación efectiva, control y monitorización de procesos industriales ya que cada intercambio de datos sigue un modelo específico que asocia la dirección del dispositivo y un código de función. (AprendeaProgramarPLC, 2021)

Interfaz HOMBRE-MAQUINA (HMI)

La Interfaz Hombre-Máquina (H MI) resulta un componente crucial en el sector industrial, debido a que permite a los trabajadores interactuar directamente con la maquinaria y los sistemas a través de una plataforma visual. Las HMI permiten la exposición de información en tiempo real, facilitando la administración, supervisión y gestión de varios procesos industriales. A través estas interfaces, los usuarios tienen la posibilidad de hacer modificaciones en los dispositivos, tales como encender o apagar dispositivos, modificar los parámetros del sistema operativo, o supervisar el rendimiento de los equipos. (COPADATA, s.f.)

En su versión más básica, una HMI puede ser una pantalla independiente o un panel vinculado a un dispositivo. Su función primordial consiste en simplificar el acceso a los datos operativos y potenciar la interacción del usuario con las máquinas, optimizando así la eficiencia la eficiencia en el funcionamiento y la administración de procesos industriales. (COPADATA, s.f.)

La figura 3 muestra la implementación de una interfaz Hombre-Máquina (HMI) utilizada para supervisión y el control de procesos Industriales. Esta interfaz permite que los operadores puedan interactuar con la maquinaria en tiempo real, dando la visualización de información precisa del sistema y facilitando la toma de decisiones.

Figura3
Interacción con una interfaz Hombre-Maquina *(HMI)*

Nota: La HMI permite el control y supervisión de los procesos productivosFuente especificada no válida..

Raspberry PI en Sistemas Industriales.

La Raspberry Pi se ha convertido en un dispositivo imprescindible en el ámbito de la automatización industrial por su tamaño reducido, bajo costo y adaptación. Este equipo de computación monoplaca se utiliza en una amplia variedad de contextos industriales, desde la producción hasta la asistencia sanitaria y el transporte. Compañías de cualquier magnitud, desde grandes hasta las más reducidas hasta las más amplias pequeñas, utilizan la Raspberry Pi para generar soluciones industriales revolucionarias. Adicionalmente, con la aparición del RP2040 y la Raspberry Pi Pico, se ha presentado la gama de productos, incluyendo microcontroladores creados para usos industriales específicas. (Raspberry Pi, 2023)

La capacidad de la Raspberry Pi para gestionar señales y manejar dispositivos lo convierte en una plataforma perfecta para sistemas de automatización industrial. Su reducido tamaño y accesibilidad facilitan a las compañías la creación y elaboración de soluciones eficaces, manteniéndose actualizadas con las innovaciones tecnológicas para satisfacer con las demandas del mercado. (Raspberry Pi, 2023).

La figura 4 presenta la aplicación de una Rapberry Pi en un entorno de automatización industrial, destacando su capacidad para gestionar señales y controlar dispositivos. Gracias a su tamaño compacto y de bajo costo, ya que este equipo monoplaca se ha consolidado como una solución versátil y eficiente en la creación de Sistemas Industriales.

Figura 4

Raspberry Pi implementada en un entorno industrial para la automatización de procesos

Nota: La imagen muestra una Raspberry Pi conectada a dispositivos industriales, destacando su uso en el control y automatización. (Raspberry Pi, 2023)

Herramientas y entornos de desarrollo.

Visual Studio Code como entorno de desarrollo.

Visual Studio Code (VS Code) es un entorno de desarrollo adaptable que facilita a los usuarios el trabajo en equipos a distancia, contenedores o que utilizan el Subsistema de Windows para Linux (WSL), impidiendo la necesidad que se ejecute l código en el equipo local. Esta característica permite distinguir el entorno de desarrollo de la configuración local, incrementando la consistencia entre los colaboradores y facilitando la utilización de diversos equipos especializados o varios sistemas operativos. (Desarrollo Remoto con VS Code, 2024)

Mediante las extensiones de desarrollo a distancia, VSCode simplifica la implementación de comandos, la depuración de aplicaciones y el uso de herramientas en el entorno ambiente remoto, preservando una experiencia de usuario parecida a la realización local. Esto resulta especialmente ventajoso para desarrollar aplicaciones en ambientes no accesibles a nivel local, o para depurar aplicaciones en la nube o en sistemas de producción. (Desarrollo Remoto con VS Code, 2024).

Python y sus ecosistemas

Dispone un amplio ecosistema de librerías que facilitan a los programadores la ampliación de las habilidades de programación, proporcionando soluciones especializadas para diversas funciones. En este proyecto emplea librerías como OpenCV, MediaPipe, numpy, scikit-learn, pickle, os y python-snap7, que desempeñan un papel crucial en el procesamiento de imágenes, la inteligencia artificial, la administración de datos y la comunicación con dispositivos industriales. (Aurora, 2023)

OpenCV se utiliza para la visión computacional, mientras que MediaPipe facilita la identificación de gestos en tiempo real. numpy se ocupa de llevar a cabo operaciones matemáticas eficaces con matrices, fundamentales para el manejo de información. Por otro lado, Scikit-learn proporciona herramientas de aprendizaje automático para la generación y el desarrollo promedio de modelos. Además, facilita el almacenamiento y recuperación de objetos en serie. Además, la librería os simplifica la comunicación con el sistema operativo, mientras que python-snap7 posibilita la interacción con PLCs, lo que resulta crucial para la incorporación de un sistema de control industrial en este proyecto. (Aurora, 2023)

TIA Portal V18.

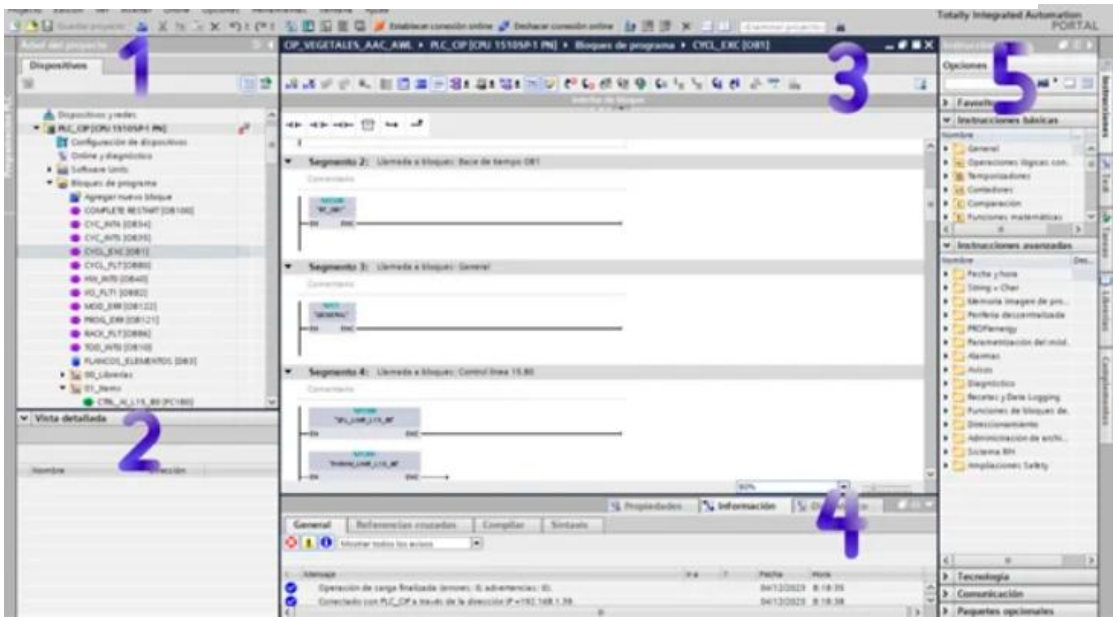
TIA Portal V18 es un software vital para una automatización industrial, ofreciendo un trabajo de entorno de crecimiento unificado para la programación, puesta en la marcha y administración de sistemas de control. Este software permite la programación de Controladores Lógicos Programables (PLCs) como los modelos S7-1200, S7-1500, entre otros, utilizando lenguajes como Ladder, Grafcet y texto organizado. Su versatilidad posibilita que los programadores optimicen procesos industriales al integrar de manera más sencilla dispositivos como servomotores, reguladores de frecuencia y módulos de entrada y salida. (WIT AUTOMATIZACIÓN, s.f.)

Además, TIA Portal V18 facilita la creación y alteración de interfaces HMI (Interfaz Humano-Maquinaria) para la supervisión y administración en tiempo real de los procesos industriales. Dentro de sus características destacan la simplificación en el diagnóstico de anomalías, la implementación de copias de seguridad de los sistemas de control y la configuración de alertas. Estas definiciones lo transforman en una plataforma eficaz que incrementa la eficiencia de los sistemas de automatización industrial. (WIT AUTOMATIZACIÓN, s.f.)TIA Portal V18 es un instrumento crucial para la automatización industrial, ofreciendo un ambiente de desarrollo unificado para la programación, puesta en marcha y administración de sistemas de control. Este programa informático facilita la programación de Controladores Lógicos Programables (PLCs) como los modelos S7-1200, S7-1500, entre otros, empleando lenguajes como Ladder, Grafcet y texto organizado. Su adaptabilidad posibilita que los programadores mejoren procesos industriales al incorporar fácilmente dispositivos como servomotores, reguladores de frecuencia y módulos de entrada y salida. (WIT AUTOMATIZACIÓN, s.f.).

La figura 5 ilustra la programación y el uso del software TIA portal V18 en un sistema de automatización industrial, destacando su capacidad para programar controladores lógicos programables (PLCs) como los modelos S7-1200 y S7-1500.

Figura 5

Interfaz de usuario de TIA PORTAL V18



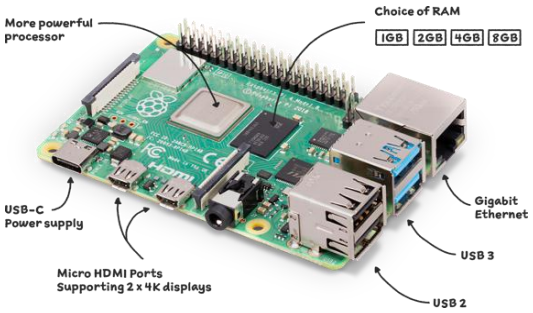
Nota: La imagen muestra la programación dentro del software TIA PORTALV18 (WIT AUTOMATIZACIÓN, s.f.)

Raspberry Pi

La figura 6 muestra la Raspberry Pi, una pequeña computadora asequible que ha ganado gran popularidad en proyectos relacionados con electrónica, robótica y desarrollo de software. Su reducido tamaño casi se asemeja al de una tarjeta de crédito. Implementa sistemas operativos basados en Linux, siendo Raspberry Pi OS el más habitual. Dependiendo del modelo, puede incorporar Wi-Fi y Bluetooth, lo que simplifica su utilización en proyectos relacionados con Internet de las Cosas (IoT). (Raspberry, 2024)

Figura 6

Raspberry Pi 4 Model B



Nota: Se describen los componentes de la placa Raspberry Pi 4 Model B. (Raspberry, 2024)

MARCO METODOLÓGICO

Descripción del sistema

El sistema de diseño e implementación de un proceso industrial accionado mediante gestos usando PLC, HMI y un ordenador monoplaca es un modelo el cual permite que diversos gestos sean captados por una cámara y sean ejecutados en un programador lógico programable.

Utiliza el protocolo de comunicación de Siemens S7 junto con tecnología ethernet para poder realizar el envío de señales generadas por el usuario. Los gestos configurados son simples e interactivos para que el usuario pueda correr el proceso designado.

Para el desarrollo de este sistema, fue necesario realizar un entrenamiento de gestos con un modelo de visión artificial, un protocolo de comunicación para la recepción de señales, el uso de un entorno de desarrollo integrado para poder ejecutar el sistema de visión artificial, un ordenador monoplaca en el cual se procesen las señales captadas, y un software de programación industrial para que controle los procesos del programador lógico programable.

La elaboración del sistema se divide en las siguientes fases de desarrollo:

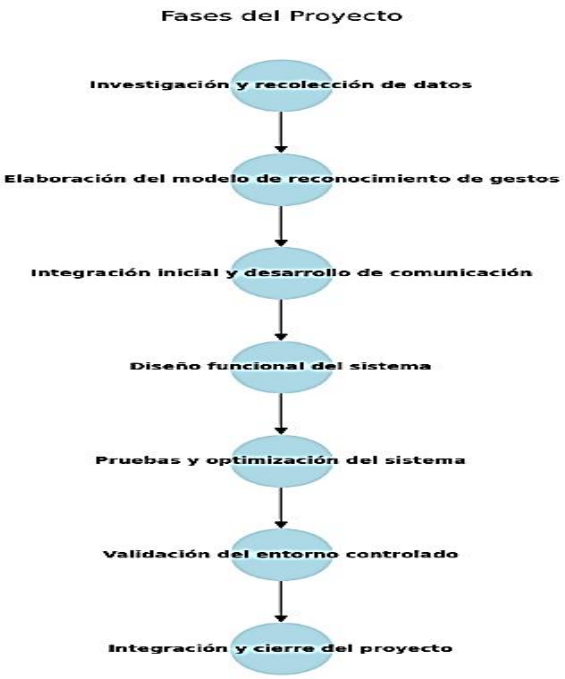
- Investigación y recolección de datos
- Elaboración del modelo de reconocimiento de gestos
- Integración inicial y desarrollo de comunicación
- Diseño funcional del sistema
- Pruebas y optimización del sistema
- Validación del entorno controlado
- Instalación en un entorno piloto

- Integración y cierre del proyecto

A continuación, se invita a visualizar un diagrama de las fases de la elaboración del sistema representado cronológicamente en la figura 7.

Figura 7

Diagrama de bloques de las etapas del diseño del proceso industrial accionado mediante gestos



Nota: Diagrama de fase del proyecto hecha en Python.

Investigación y recolección de datos

En la etapa de investigación y recolección de datos, se empezó por indagar los gestos mayormente usados por las personas. Se realizaron preguntas a profesores, familiares cercanos y amigos. Así mismo, fuentes que recopilan gestos de lenguajes de señas fueron investigadas. Con lo cual, se concluyó en utilizar 6 gestos para correr nuestro proceso industrial.

Estos gestos serían recopilados para crear el dataset y poder hacer uso de este cuando el modelo de visión artificial sea desarrollado. Los gestos que fueron utilizados son los siguientes:

Gesto 1 Marcha: El gesto de marcha fue distinguido de entre los otros gestos, para representar el inicio de accionamiento del proceso. Este gesto utiliza todos los dedos de la mano cerrados, con excepción del pulgar, el cual debe apuntar hacia arriba y con la orientación de la mano en sentido horizontal. En esta ocasión, este gesto permitirá que el motor se accione al momento de ser reconocido por el modelo de visión artificial con la cámara. Se invita al usuario a revisar la figura 8 de gesto de marcha.

Figura 8

Gesto de marcha.

Nota: El gesto consta del pulgar levantado apuntando hacia arriba, la mano cerrada, y la orientación de la mano horizontal.

Gesto 2 Paro: El gesto de paro fue distinguido de entre los otros gestos, para representar el descanso del sistema cuando está en ejecución. Este gesto utiliza todos los dedos de la mano abiertos y con la orientación de la mano en sentido vertical. En esta ocasión, este gesto permitirá que el motor detenga su ejecución al momento de ser reconocido por el modelo de visión artificial con la cámara. Se invita al usuario a revisar la figura 9 de gesto de paro.

Figura 9

Interacción con una interfaz Hombre-Maquina (HMI)

Nota: En esta imagen se visualiza que el gesto es con la mano en orientación vertical, con todos los dedos extendidos.

Gesto 3 Aumento de velocidad: El gesto de aumento de velocidad fue seleccionado con la finalidad de tener un gesto que represente el incremento en algún sentido. Este gesto utiliza todos los dedos de la mano cerrados con excepción al índice, y con la mano en posición vertical. En este caso, este gesto permitirá que el motor incremente su velocidad al ser reconocido por el modelo de visión artificial con la cámara. Se invita al usuario a revisar la figura 10 de gesto de disminución de velocidad.

Figura 10

Gesto de incremento de velocidad

Nota: Para este gesto la mano se encuentra en posición vertical, con el dedo índice extendido, y los demás dedos contraídos.

Gesto 4 Disminución de velocidad: El gesto de disminución de velocidad fue seleccionado con la finalidad de tener un gesto que represente el decremento en algún

sentido. Este gesto utiliza todos los dedos de la mano cerrados simulando un puño, pero con la mano en posición vertical. En este caso, este gesto permitirá que el motor decremente su velocidad al ser reconocido por el modelo de visión artificial con la cámara. Se invita al usuario a revisar la figura 11 de gesto de disminución de velocidad.

Figura 11
Gestos de disminución de velocidad

Nota: Este gesto consta del puño, es decir todos los dedos contraídos y la mano en posición vertical.

Gesto 5 Giro antihorario: El gesto de giro antihorario fue escogido con la finalidad de tener un gesto antagónico al gesto horario. Este gesto utiliza el dedo índice y el dedo pulgar en contacto. Mientras que los demás dedos estarán apuntando hacia arriba. Este gesto es similar al gesto comúnmente conocido como excelente. En este caso, este gesto permitirá que el motor gire en sentido antihorario de las manecillas del reloj en vista del usuario al ser reconocido por el modelo de visión artificial con la cámara. Se invita a ver la Figura 12 del gesto antihorario.

Figura 12
Gestos de giro antihorario

Nota: Para este gesto, los dedos índice y pulgar deben tocar sus dos puntas, mientras que los demás dedos de la mano deben estar extendidos con la mano en posición vertical.

Gesto 6 Giro horario: El gesto de giro horario fue escogido con la finalidad de tener un gesto amigable y que los demás usuarios no se vean en la dificultad de ejecutarlo. Este gesto utiliza el dedo índice y el dedo anular apuntando hacia a arriba, similar al gesto de paz. En este caso, este gesto permitirá que el motor gire en sentido horario de las manecillas del reloj en vista del usuario al ser reconocido por el modelo de visión artificial con la cámara. Se invita a ver la Figura 13 del gesto horario.

Figura 13
Gesto de giro horario

Nota: Este gesto consta de los dedos índice y corazón, extendidos, los demás dedos contraídos y la mano en posición vertical.

Elaboración del modelo de reconocimiento de gestos.

En la elaboración del modelo de reconocimiento de gestos, se procedió a investigar las librerías a utilizar para poder realizar un entrenamiento efectivo en cuanto a los gestos. Un modelo de reconocimiento de gestos fue utilizado para poder acoplar nuestro propio sistema de visión artificial y entrenarlo utilizando nuestros gestos. Para dicho modelo, fue útil la versión 3.8.10 de Python, ya que era una versión estable y que se acoplaba a las versiones funcionales de todas las librerías. No se utilizó la última librería de Python debido a que el modelo de visión artificial trabajado utilizaba estas versiones de Python por tema de precisión, armonía con las librerías usadas, y ejecución óptima del código con esta versión. Las librerías de Python utilizadas fueron las siguientes:

- Mediapipe
- Opencv Python
- Scikit learn

Con el script collect_images.py, se inicializó la cámara, y con la ayuda de Open Cv, se creó una carpeta data, la cual almacena las imágenes capturadas por la librería Open Cv. Con este script, el usuario es libre de capturar la cantidad de imágenes que quiera por cada clase, y así mismo, poder almacenar la cantidad de clases deseadas. Esto se comprueba en las líneas 10 y 11, las cuales son las variables que almacenarán la cantidad de imágenes y de clases que el modelo procederá a tener.

Así mismo, el código está programado para que cuando el usuario esté listo, capture sus gestos a utilizar presionando la letra Q, en la cual se puede ver en el bloque while de la línea 21. Por otro lado, la librería os, permitió manipular dichas imágenes capturadas, para poder crear las subcarpetas de cada clase, y en cada una almacenar las imágenes correspondientes.

Una vez recopiladas las imágenes, se procedió a realizar la creación del dataset. Para su posterior compresión de archivo en la extensión pickle, esto fue realizado con el archivo create_dataset.py. con este script, las imágenes capturadas son leídas, puestas en un formato que la computadora pueda leer. Esto con el fin de que el modelo sea posteriormente manipulado en el entrenamiento.

En este caso, las imágenes son capturadas en formato RGB, y los nodos de la mano sean recortados y almacenados en un archivo data con la extensión pickle, el cuál comprime todo el dataset y lo serializa en una secuencia de bytes para almacenarlo en el archivo antes mencionado.

En el script train_classifier.py, se realizó el entrenamiento con la librería scikit-learn, la cual permite manipular los datos creados en el archivo comprimido data con extensión pickle, y con la librería numpy, leer los datos como un array.

Esto con la finalidad de crear un archivo model con extensión p, la cuál es una extensión que almacena archivos de módulos Python que aún no se hallan en formato de transferencia de bytes.

Finalmente, se puede encontrar el archivo inference.py, el cual será el archivo que corra el modelo, y le muestre al usuario el modelo entrenado con las etiquetas designadas en dicho archivo.

Como se puede apreciar en las líneas de código del script, todas las librerías son utilizadas para poder correr el modelo. Sin embargo, solo se utilizará el archivo model con extensión p, el cual tiene el entrenamiento del modelo, para ejecutar el modelo de visión artificial final con los gestos que fueron recopilados.

También es visible en las imágenes que la recopilación de gestos nunca fue etiquetada en el código. No obstante, nuestro modelo es capaz de en este script, etiquetar dichos gestos para que no sean mostrados en pantalla como 0, 1, 2, etc. Sino que sean puestos un nombre deseado por el usuario. Por elección propia, se decidió a poner las acciones que realizará el motor.

Todos estos archivos pueden ser encontrados en la sección de anexos con su respectiva licencia, ya que, para este proyecto de tesis, se realizó algunas modificaciones para su posterior uso.

Integración inicial y desarrollo de comunicación

Para el desarrollo de la comunicación, se realizó varias pruebas de comunicación con diversos métodos. Entre estos métodos, se optó por la librería Snap7, la cual es una librería

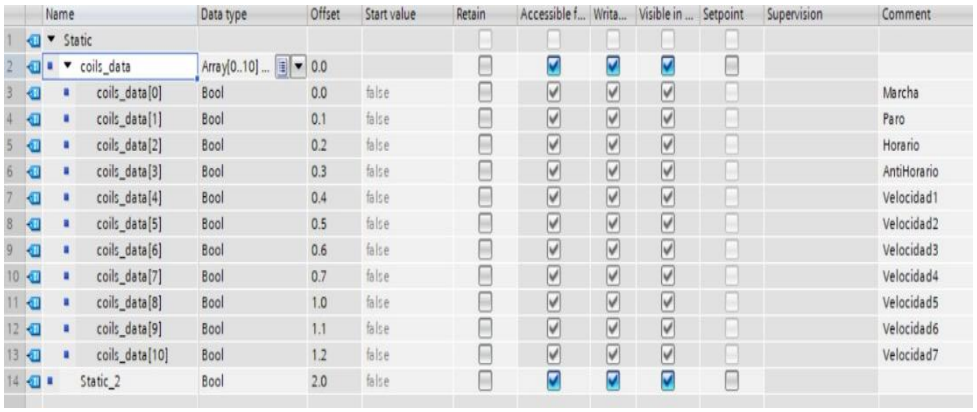
fuente abierta, ethernet multi plataforma que permite comunicar nativamente con programadores lógicos programables Siemens S7.

Esta librería permite tanto la lectura como escritura de datos en el PLC. Para esta ocasión, simplemente se escribió datos cambiando el estado de las variables booleanas designadas en un Data Block en TIA Portal para el uso de las entradas y salidas del PLC.

A continuación, se invita a visualizar la tabla que contiene las variables que ejecuta los procesos extraída de la data block de TIA Portal en la figura 14.

Figura 14

Tabla de Variables del Data Block DB_COILS_SNAP7



	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supervision	Comment
1	Static										
2	coils_data	Array[0..10] ...	0.0			☑	☑	☑			
3	coils_data[0]	Bool	0.0	false		☑	☑	☑			Marcha
4	coils_data[1]	Bool	0.1	false		☑	☑	☑			Paro
5	coils_data[2]	Bool	0.2	false		☑	☑	☑			Horario
6	coils_data[3]	Bool	0.3	false		☑	☑	☑			AntiHorario
7	coils_data[4]	Bool	0.4	false		☑	☑	☑			Velocidad1
8	coils_data[5]	Bool	0.5	false		☑	☑	☑			Velocidad2
9	coils_data[6]	Bool	0.6	false		☑	☑	☑			Velocidad3
10	coils_data[7]	Bool	0.7	false		☑	☑	☑			Velocidad4
11	coils_data[8]	Bool	1.0	false		☑	☑	☑			Velocidad5
12	coils_data[9]	Bool	1.1	false		☑	☑	☑			Velocidad6
13	coils_data[10]	Bool	1.2	false		☑	☑	☑			Velocidad7
14	Static_2	Bool	2.0	false		☑	☑	☑			

Nota: Tabla de Variables del Data Block DB_COILS_SNAP7 usadas para correr los gestos.

Al momento de realizar las pruebas de comunicación, se comprobó en tiempo real que, al enviar los datos de Python al PLC, haya sido exitoso. Esto se realizó asignado el puerto ethernet de la computadora la ip requerida para que pueda estar en la misma red LAN que el PLC. Con esto, también se asignó en el PLC la ip, el rack, y el slot correspondiente al PLC en el que se estaba trabajando. En este caso, se usó esta simulación de puesta en marcha de motor para poder realizar las pruebas:

Diseño Funcional Del Sistema

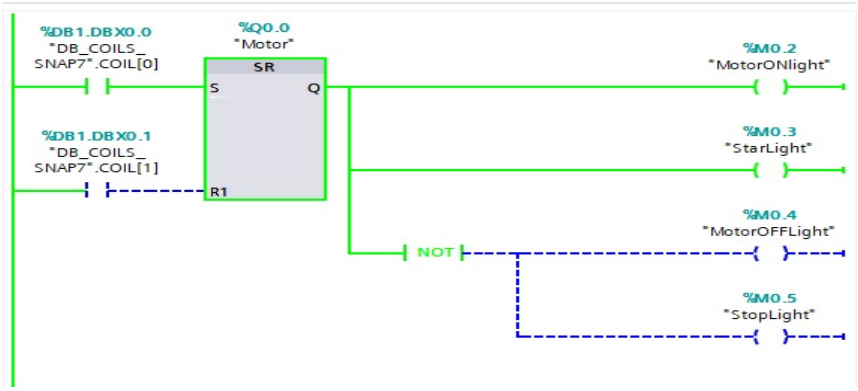
Para el diseño funcional del sistema se empezó a ajustar la tabla de variables, para que trabaje correctamente en el código de programación LADDER. Para esto, fue necesario primero configurar el PLC. Se procedió a configurar el Data Block usado, y asignar las variables booleanas que se encuentran en el Data Block para que tengan accionamiento.

Posteriormente, se pasó a la fase de pruebas, para verificar que el código elaborado corría correctamente en TIA Portal y no tenía ningún problema.

A continuación, se presenta el segmento de marcha paro ejecutado con el bloque set, reset, pulsadores y bobinas extraído del código de TIA Portal, como se muestra en la figura 15.

Figura 15

Segmento de marcha y paro extraído del código de TIA Portal.

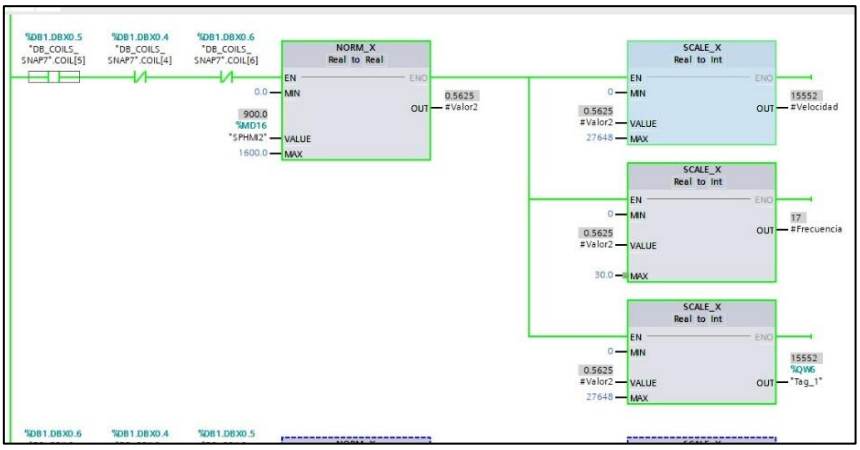


Nota: Segmento de marcha y paro extraído del código de TIA Portal, el cual acciona el motor, detiene el motor, y prende la luz de encendido.

A continuación, se presenta el segmento de escalado de velocidades usando los bloques NORM_X y SCALE_X que nos permitirán mandar la señal al VDF, del código de TIA Portal en la figura 16.

Figura 16

Segmento de escalamiento de las velocidades

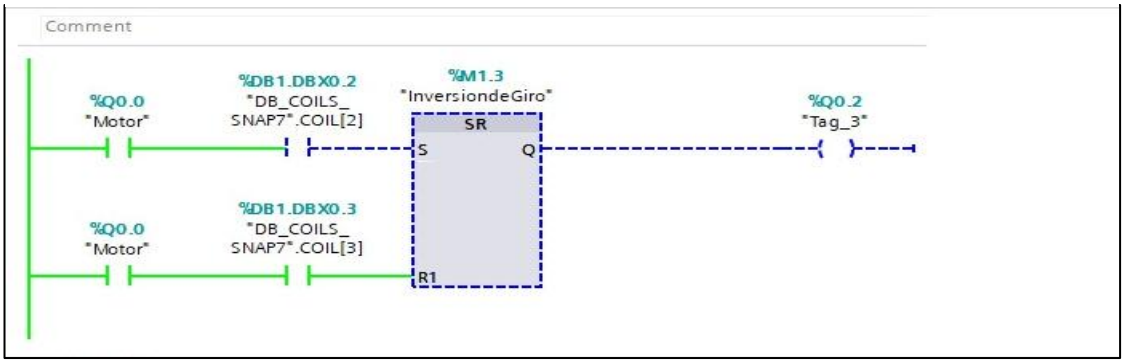


Nota: Segmento de escalamiento de las velocidades con el uso de 3 velocidades fijas.

A continuación, se presenta el segmento de inversión de giro utilizando el bloque set reset, pulsadores y bobinas del código de TIA Portal en la figura 17.

Figura 17

Segmento de inversión de giro extraído del código de TIA Portal



Nota: Segmento de inversión de giro, ya sea horario o antihorario con el bloque set reset.

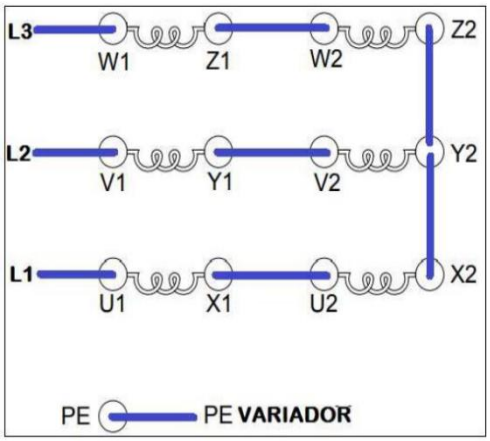
Pruebas y optimización del sistema

En las pruebas y optimización del sistema, se procedió a hacer las conexiones físicas del PLC con el motor, la alimentación, la distribución, las salidas digitales, los relés y el variador de frecuencia.

Para la conexión del motor fue necesario el esquema que se presenta al usuario para poder conectar en estrella el motor en la figura 18.

Figura 18

Conexión en estrella de motor



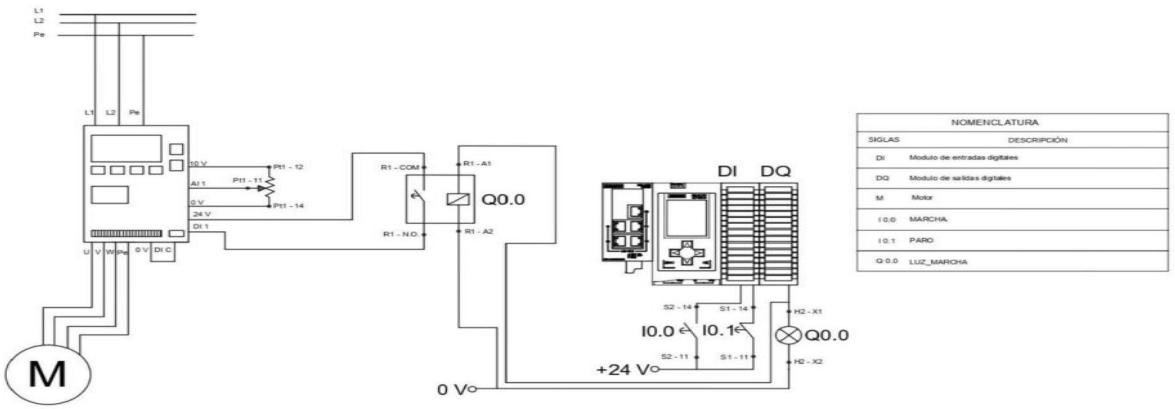
Nota: Diagrama de conexión en estrella de motor trifásico. (Guerrón, 2024)

En la conexión del motor, se realizó una conexión en estrella del motor trifásico.

Para la conexión de los elementos que van a ejecutar los diversos procesos como la inversión de giro, el marcha y paro, y el cambio de velocidades, fue necesario el esquema que en la figura 19.

Figura 19

Diagrama de fuerza y control práctica 5 de Automatización Industrial 2.



Nota: Práctica 5 de Automatización Industrial 2. (Guerrón, 2024)

Las modificaciones que se realizó en este diagrama, una vez configurado el variador en modo Cn002, fue que fue puesta una salida digital Q0.2 del PLC, con la entrada digital DI2 del variador, y el AI0- del PLC fue conectado a tierra. DI1 del variador fue conectado al relé NO R1. COM1 del mismo relé fue conectado a 24 voltios. AI2 del variador y R1 del relé va igualmente conectados a tierra. Y para finalizar, A1 R1 del relé se conectó a la luz H2 X1.

Validación del entorno controlado

Una vez realizado el código para poder hacer las pruebas, se elaboró el entorno controlado. Este entorno consistía en no enviar datos al PLC con gestos, sino con escritura por la terminal de la computadora. Para esto, no fue necesario el raspberry PI, ya que con la computadora era probada la comunicación y la ejecución de los procesos descritos.

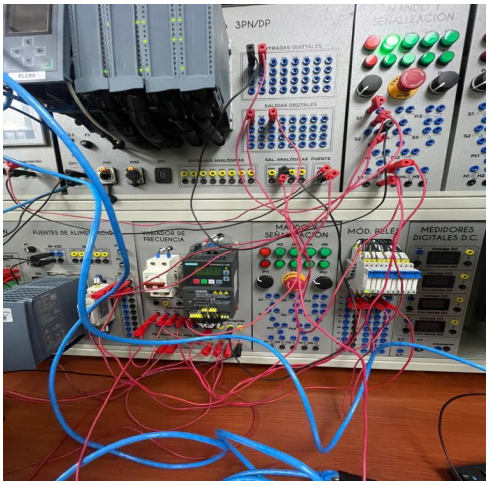
En el entorno controlado, destacó la efectiva forma en que TIA Portal recibía los datos, ya que era muy efectiva la transmisión de la misma. Casi no hubo mayor retraso en el envío

de datos. Sin embargo, tocó ajustar el código en LADDER para poder realizar todos los procesos de manera efectiva.

En la figura 20 se puede visualizar el entorno controlado e implementado para las pruebas del sistema, donde la conexión entra el PLC y los dispositivos que se llevó a cabo de manera efectiva utilizando el software TIA Portal. En esta fase del proyecto no se utilizó la Raspberry Pi, y se optó por implementar una computadora para constatar la comunicación y la ejecución de los procesos mediante el envío de datos desde la terminal.

Figura 20

Conexiones para ejecutar los procesos descritos.



Nota: Conexiones del motor trifásico con el PLC, el variador de frecuencia, el relé R1 y las luces correspondientes.

Instalación en un entorno piloto

Para la instalación en un entorno piloto, se procedió a realizar los gestos con el raspberry PI, y verificar que el mismo corra el modelo perfectamente, pero sin enviar datos del PLC.

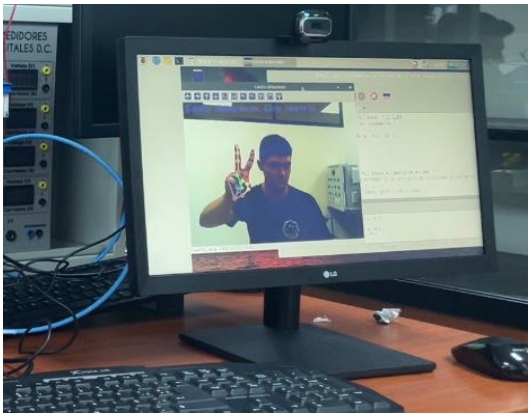
En la preparación del raspberry, se instalaron todas las librerías en un ambiente virtual, para que el modelo se ejecute correctamente. Ya que el sistema operativo instalado en la raspberry tenía Python 3.9, no hubo ningún problema con la instalación de las mismas. En

la raspberry se instaló un sistema operativo Bullseye. Para esto, se procedió a elaborar 3 modelos de prueba.

El primer modelo consistía en 1000 muestras de cada clase, el segundo, constaba de 500 muestras de cada clase, y finalmente el tercero consistía en 100 muestras de cada clase. El primer modelo tenía cierta dificultad al abrir y al ejecutar el modelo. Inclusive, muchas ocasiones no podía abrir la cámara. El segundo modelo, no tenía ningún problema con abrir la cámara al momento de ejecutar el script. Sin embargo, no era efectiva la detección de los gestos y la cámara tenía un retraso notable. Finalmente, el tercero corrió perfectamente. Presentaba un ligero retraso en la fluidez de captura de video. No obstante, su detección en gestos fue exacta y la mejor entre las 3 pruebas que se realizó. Se presenta la figura 21, la cual contienen una imagen de las pruebas de ejecución del modelo de visión artificial en la raspberry pi ejecutando gestos.

Figura 21

Prueba de ejecución del modelo de visión artificial.



Nota: Ejecución y reconocimiento de los gestos entrenado en el modelo de visión artificial.

Integración y cierre del proyecto

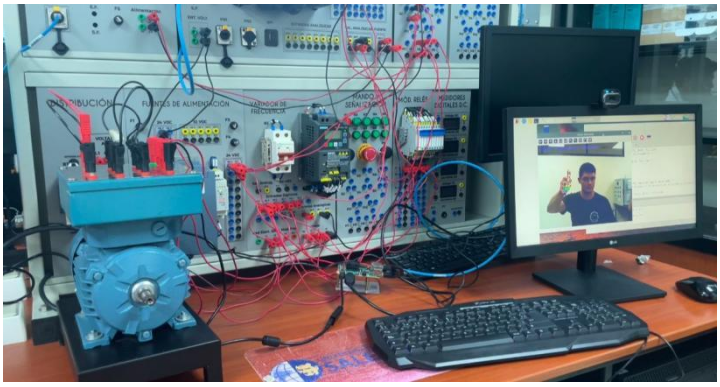
Finalmente, se tuvo que cambiar la ir del puerto ethernet de la computadora monoplaca, para que esté en la misma red que el PLC. Al haber muchos dispositivos en la misma red, se optó por configurar la raspberry con la IP 172.18.135.190. Esto permitió tener conexión con el PLC al verificar desde la terminal, y hacer ping a nuestro servidor con ip 172.18.135.61. De tal manera, que lo único que faltaba, era correr el modelo. En la ejecución del modelo, al cargar más funciones de código y requerir más recursos, se

visualizó que la cámara era mucho menos fluida que al momento de solo correr el modelo para captar gestos. No obstante, este ruido no impedía la captación de gestos y envío de datos en el tiempo impuesto en el código de Python al PLC.

Se presenta en la figura 22 el funcionamiento del motor ejecutando el modelo de visión artificial con gestos ya implementado al raspberry pi con el primer gesto de marcha.

Figura 22

Prueba del diseño e implementación del proyecto descrito

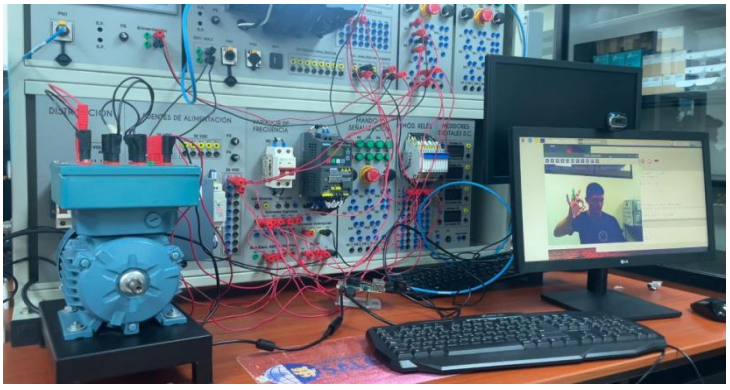


Nota: Ejecución del gesto marcha y envío de datos al PLC para el accionamiento del motor.

En la figura 23 se muestra el funcionamiento del motor ejecutando el modelo de visión artificial con gestos ya implementado al raspberry pi con el cuarto gesto de inversión de giro antihorario.

Figura 23

Prueba del diseño e implementación del proyecto descrito



Nota: Ejecución del gesto inversión del giro horario y envío de datos al PLC para el accionamiento del cambio de giro.

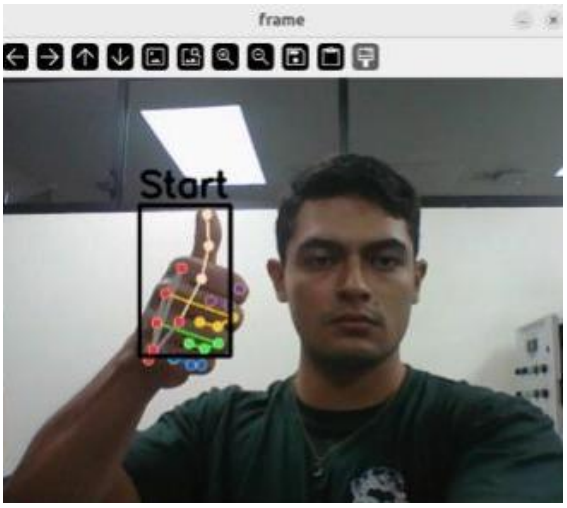
RESULTADOS

Se ejecutó correctamente el modelo de visión artificial al momento de realizar los gestos y para verificar que el modelo ejecutó, observamos como en la cámara se visualiza el nombre del gesto.

Se presenta el modelo de visión artificial en ejecución con sus respectivas etiquetas dependiendo del gesto, en este caso el gesto marcha etiquetado en inglés en la figura 24.

Figura 24

Modelo de visión artificial en ejecución.

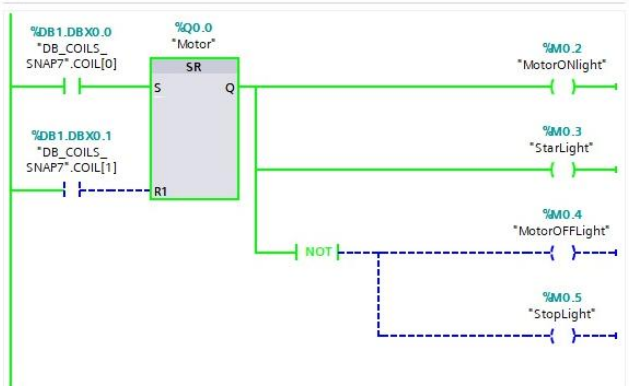


Nota: Modelo de visión artificial en ejecución detectando los gestos entrenados.

Se ejecutaron los procesos con el motor correctamente con el código de Ladder. Se muestra en la figura 25 el segmento de marcha paro en ejecución en el código de TIA Portal, mientras el motor está en marcha y enciende las salidas correspondientes.

Figura 25

Segmento de marcha y paro extraído del código de TIA Portal.



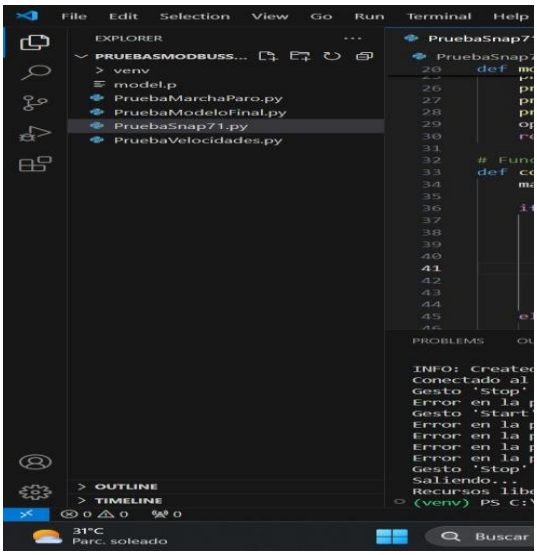
Nota: Segmento de marcha y paro extraído del código de TIA Portal, el cual acciona el motor, detiene el motor, y prende la luz de encendido.

Se comunicó efectivamente el ambiente creado en python con el PLC.

Se muestra la figura 26 donde se aprecia el directorio para la comunicación entre el dispositivo a ejecutar el código de Python y el PLC, y su prueba de conexión impresas por terminal creado.

Figura 26

Directorio de pruebas de archivos de prueba.

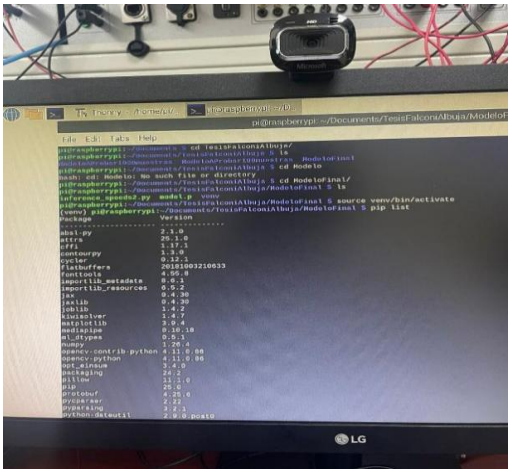


Nota: Directorio de archivos de prueba usando SNAP 7

Se realizaron los procesos enviados por computadora usando python efectivamente en el motor, se configuró perfectamente el ambiente virtual en la raspberry pi, y se asignó la IP para poder estar en la misma red que el PLC.

Se presenta al usuario la terminal de la computadora monoplaca con sus configuraciones respectivas de dirección ip y librerías de Python instaladas en la figura 27.

Figura 27
Configuración el raspberry pi.



Nota: Creación de un ambiente virtual en el raspberry y descarga de todas las librerías necesarias para la ejecución del modelo de visión artificial.

Se muestra al usuario la terminal de la computadora monoplaca con sus con la prueba de la comunicación con el PLC al ya encontrarse en la misma red en la figura 28.

Figura 28

Prueba de la comunicación del raspberry

```
wheel                   0.45.1
zipp                    3.21.0
(venv) pi@raspberrypi:~/Documents/TesisFalconiAlbuja/ModeloFinal $ ping 172.18.135.61
PING 172.18.135.61 (172.18.135.61) 56(84) bytes of data.
64 bytes from 172.18.135.61: icmp_seq=1 ttl=255 time=0.660 ms
64 bytes from 172.18.135.61: icmp_seq=2 ttl=255 time=0.355 ms
64 bytes from 172.18.135.61: icmp_seq=3 ttl=255 time=0.364 ms
64 bytes from 172.18.135.61: icmp_seq=4 ttl=255 time=0.413 ms
64 bytes from 172.18.135.61: icmp_seq=5 ttl=255 time=0.341 ms
64 bytes from 172.18.135.61: icmp_seq=6 ttl=255 time=0.304 ms
64 bytes from 172.18.135.61: icmp_seq=7 ttl=255 time=0.323 ms
64 bytes from 172.18.135.61: icmp_seq=8 ttl=255 time=0.302 ms
64 bytes from 172.18.135.61: icmp_seq=9 ttl=255 time=0.314 ms
64 bytes from 172.18.135.61: icmp_seq=10 ttl=255 time=0.325 ms
64 bytes from 172.18.135.61: icmp_seq=11 ttl=255 time=0.330 ms
^C
--- 172.18.135.61 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10208ms
rtt min/avg/max/mdev = 0.302/0.366/0.660/0.097 ms
(venv) pi@raspberrypi:~/Documents/TesisFalconiAlbuja/ModeloFinal $
```

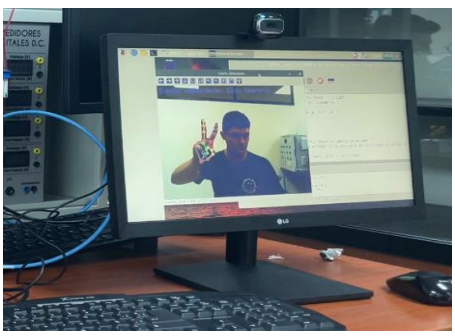
Nota: Prueba de la comunicación del raspberry pi con el PLC.

La Raspberry ejecutó el modelo de visión artificial sin ningún inconveniente. Es importante recalcar que fueron utilizadas las librerías óptimas y la versión de Python adecuada, para que el modelo no se vea afectado en ejecución por el procesador de la computadora monoplaca.

Se muestra la ejecución del modelo de visión artificial en la computadora monoplaca ya con el modelo entrando, con la cámara configurada, y leyendo los gestos en idioma español en la figura 29.

Figura 29

Prueba del modelo en la raspberry

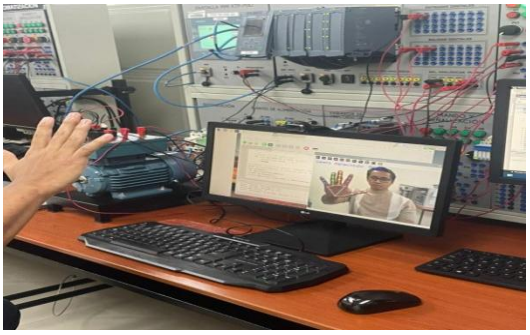


Nota: Prueba de ejecución del modelo de visión artificial en la raspberry Pi 4.

La Raspberry envío los procesos a correr con el motor y el PLC ejecutó los procesos en conjunto con el variador, en el motor.

A continuación, se presenta la imagen de la ejecución de gestos en la Raspberry Pi utilizando el modelo de visión artificial, el cual envía señales al PLC para la ejecución de acciones en el motor, como se observa en la figura 30.

Figura 30
Prueba del sistema completo



Nota: Prueba del modelo ya implementado, con uso de gestos usando la computadora monoplaca.

CRONOGRAMA DE ACTIVIDADES

Para el cronograma de actividades, se procedió a realizar una planificación para las horas que fueron invertidas en el proyecto teniendo en cuanta cada etapa y cada fase correspondientes. Para el proceso 1, el cual constaba del desarrollo del sistema de reconocimiento de gestos, se recolectaron los datos en la primera semana de noviembre al igual que el desarrollo de algoritmos. El entrenamiento del modelo y la programación de la interfaz de comunicación se trabajó y concluyó en la segunda semana de noviembre. En la tercera semana de noviembre se realizó la validación del algoritmo de reconocimiento y el mapeo de gestos y acciones. Para la última semana de noviembre, se probó y ajustó el sistema y se adicionó un ajuste de parámetros adicionales necesarios que se encontraron durante las pruebas.

En la primera semana de diciembre se elaboraron diversas pruebas en diferentes condiciones tomando las limitaciones del ambiente a utilizar, y se valida el entorno. La instalación piloto y la implementación en este entorno fue realizado durante la segunda semana de diciembre. Para verificar el sistema, se realizó una revisión en la efectividad y en la precisión durante la tercera semana de diciembre. Para la última semana de diciembre, se concluyó el proceso 2, el cual era la integración con el sistema industrial y el ajuste de parámetro adicionales durante este proceso.

Empezando la primera semana de enero, se realizaron las pruebas y ajustes finales, y para la segunda, tercera y cuarta semana de enero se finalizó el proyecto con la implementación completa, adición de parámetro para que el sistema sea amigable para el usuario y se cerró el proyecto.

A continuación, se procederá a presentar la Tabla 1 de cronograma de actividades, la cual fue indispensable para realizar las actividades designadas en el lapso disponible.

Tabla 1

Cronograma de Actividades

Meses	Noviembre				Diciembre				Enero			
Semanas	1	2	3	4	1	2	3	4	1	2	3	4
Proceso 1: Desarrollo del Sistema de reconocimiento gestos	X											
Recolección de gestos	X											
Desarrollo de algoritmos	X											
Entrenamiento del modelo		X										
Programación de la interfaz de comunicación		X										
Validación del Algoritmo de reconocimiento			X									
Mapeo de gestos y acciones			X									
Pruebas y ajustes				X								
Ajustes de parámetros adicionales				X								
Pruebas en diferentes condiciones					X							
Validación en entorno de prueba					X							
Implementación Piloto						X						
Instalación en entorno piloto						X						
Revisión de la efectividad							X					
Revisión de la precisión							X					
Proceso2: integración con el sistema industrial								X				
Ajustes de parámetros adicionales								X				
Pruebas y ajustes finales									X			
Implementación completa y cierre del proyecto										X	X	X

PRESUPUESTO

En el presupuesto se invirtió solamente horas de ingeniería presupuesto fue elaborada teniendo en cuenta el salario básico determinado por el Ministerio del Trabajo del Ecuador impuesto en el Acuerdo Ministerial NRO. MDT-2024-300 el 1 de enero de 2025. Esto gracias a que los elementos como el PLC, el motor, los pulsadores, las luces piloto, los módulos de entradas y salidas, el SCALANCE del PLC y los cables para conectar fueron encontrados en el laboratorio de automatización industrial 2. Por otro lado, el tutor puso a disposición de los estudiantes la cámara web y la computadora monoplaca.

A continuación, se presenta la Tabla 2 de presupuesto la cual fue diseñada según las horas invertidas en este proyecto de titulación.

Tabla 2
Presupuesto

Cantidad	Descripción	Costo unitario	Costo total
160	HORA DE INGENIERÍA	$2.93	$470
		TOTAL	$470

CONCLUSIONES

Se recolectó correctamente las imágenes de gestos, y se elaboró sin ninguna complicación el dataset requerido para la ejecución del modelo de visión artificial.

El modelo de visión artificial fue entrenado, y puesto en ejecución con el uso de Python y algunas librerías, destacando “Mediapipe Hands”, para el reconocimiento de gestos añadidos al dataset y que posteriormente fueron entrenados.

Se escribió un código utilizando la librería Python-snap7, para comunicar correctamente la computadora monoplaca y el PLC, para la activación y desactivación de variables booleanas y ejecución de procesos con TIA Portal.

RECOMENDACIONES

Se recomienda limpiar la cámara y tener la luz encendida para una mejor captura de imágenes debido a que esto pudiese dificultar la captura de gestos con la cámara.

Es recomendable ubicar correctamente la cámara, para asegurar una posición óptima para una captura clara de los gestos sin obstrucciones en una posición óptima para visualizar los nodos de la mano, y que no se visualicen más de una mano en la captura de gestos para no ejecutar dos funciones a la vez y confundir el modelo.

Se recomienda añadir confirmación de gestos, e implementar un mecanismo que evite activaciones involuntarias, como un tiempo mínimo de detección o una confirmación adicional.

Es indispensable garantizar una conexión estable, y utilizar una red confiable para la comunicación entre la computadora monoplaca y el PLC, priorizando conexiones cableadas y siguiendo el protocolo S7 de Siemens para este tipo de conexiones, así como una correcta asignación de direcciones ip en la red en que se desea ejecutar dicha comunicación y sus configuraciones.

Realizar pruebas previas para verificar el funcionamiento del sistema en un entorno de prueba es muy recomendable antes de su implementación en producción, ajustando parámetros si es necesario, siguiendo primero con la implementación del modelo de visión artificial, luego con la ejecución de las acciones del motor usando TIA Portal, probando la comunicación entre el dispositivo a ejecutar el script de Python y el PLC, ejecutar el modelo de prueba a ejecutar por consola las acciones con el script de Python y PLC, y finalmente, la implementación del modelo final.

Por último y primordialmente, se sugiere instalar las versiones indicadas tanto de Python, como las librerías a utilizar para que las dependencias y las versiones no dañen la ejecución de funciones en el modelo.

REFERENCIAS BIBLIOGRÁFICAS

Agrobot. (4 de Agosto de 2020). *AGROBOT*. Obtenido de AGROBOT: https://www.agrobot.com/

Alex. (22 de Noviembre de 2023). *YouTube* . Obtenido de Electronic Board : https://www.youtube.com/watch?v=0TcVDOjR0po

AprendeaProgramarPLC. (2021). Obtenido de AprendeaProgramarPLC: https://carlosabneryt.com/comunicar-2-raspberrys-via-modbus-tcp/

Aurora. (18 de Julio de 2023). *ID boot camps*. Obtenido de ID boot camps: https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/#:~:text=Las%20librer%C3%ADas%20de%20Python%20desempe%C3%B1an,datos%20de%20manera%20m%C3%A1s%20eficiente.

Blynk. (16 de 9 de 2024). *Blynk*. Obtenido de Blynk: https://blynk.io/

Carrera, R. (Octubre de 2024). *YouTube, INTESLA*. Obtenido de CLASE 01 - SISTEMAS HMI Y SCADA: https://www.youtube.com/watch?v=CECdIFNiDAI

Computervisioneng. (2023). *Sign-language-detector-python*. Obtenido de GIthub: https://github.com/computervisioneng/sign-language-detector-python

COPADATA. (s.f.). Obtenido de COPADATA: https://www.copadata.com/es/productos/zenon-software-platform/que-significa-hmi-interfaz-humano-maquina-copa-data/#:~:text=HMI%20son%20las%20siglas%20de,para%20las%20de%20entornos%20industriales.

COPADATA. (s.f.). Obtenido de COPADATA: https://www.copadata.com/es/productos/zenon-software-platform/que-significa-hmi-interfaz-humano-maquina-copa-data/#:~:text=HMI%20son%20las%20siglas%20de,para%20las%20de%20entornos%20industriales.

Desarrollo Remoto con VS Code. (11 de 12 de 2024). Obtenido de Desarrollo Remoto con VS Code: https://code-visualstudio-com.translate.goog/docs/remote/remote-

overview?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge#:~:text=Desarrollo%20remoto%20con%20VS%20Code,cliente%20o%20en%20la%20nube.

Dinatek. (2024). *Dinatek*. Obtenido de https://dinatek.ec/producto/drones-de-fumigacion/

EIC GROUP. (16 de 9 de 2024). *EIC GROUP*. Obtenido de EIC GROUP: https://eiccontrols.com/es/inicio/480-sensores-de-nitrogeno-fosforo-y-potasio-del-suelo-salida-4-20ma.html

EMI SUITE . (s.f.). *EMI Suite 4.0: Software* . Obtenido de https://emisuite.es/vision-artificial-en-la-industria-4-0/#

Ernesto, Quezada, J., Vargas, L., & Calderon, R. (Julio de 2020). *ResearchGate*. Obtenido de Sistema de control basado en PLC y monitoreo mediante HMI para la automatización de una máquina de corte horizontal de bloques de esponja en el proceso industrial de fabricación de colchones: 342706129_Sistema_de_control_basado_en_PLC_y_monitoreo_mediante_HMI_para_la_automatizacion_de_una_maquina_de_corte_horizontal_de_bloques_de_esponja_en_el_proceso_industrial_de_fabricacion_de_colchones

Fazio, D. C. (16 de Noviembre de 2024). *MDPI*. Obtenido de https://www.mdpi.com/1999-5903/16/11/394

G., C. J. (s.f.). *OpenCV-Python*. Obtenido de OpenCV-Python: https://cjjouanne.github.io/OpenCV-Python/#:~:text=OpenCV%20es%20una%20librer%C3%ADa%20de,est%C3%A1%20expandiendo%20d%C3%ADa%20a%20d%C3%ADa.

Gemini Developer API. (24 de Abril de 2024). Obtenido de https://ai.google.dev/edge/mediapipe/framework/getting_started/python_framework?hl=es-419

Ghommam. (11 de Junio de 2022). *MDPI*. Obtenido de https://www.mdpi.com/2218-6581/11/6/139

Guerrón, A. (2024). *Diagrama de fuerza y control.* Diagrama de fuerza y control: Universidad Politécnica Salesiana.

Mais VCH. (12 de Noviembre de 2024). *Maisvch*. Obtenido de https://www.maisvch.com/es/blog/industrial-automation-plc-and-hmi/

Maisvch. (12 de Noviembre de 2024). *Comprender la automatización industrial: el papel de los PLC y las HMI.* Obtenido de https://www.maisvch.com/es/blog/industrial-automation-plc-and-hmi

NC State University . (26 de Julio de 2016). *Scientific Computing with Python and Raspberry Pi*. Obtenido de Scientific Computing with Python and Raspberry Pi: https://www.lib.ncsu.edu/events/scientific-computing-python-and-raspberry-pi-0

NorthWind Technical Services . (2023). *PROGRAMMING*. Obtenido de https://www.northwindts.com/industrial-automation/control-system-programming/

NorthWind Technical Services. (16 de junio de 2022). Obtenido de Control System Programming.

Phoenix Contact. (2023). Obtenido de https://www.phoenixcontact.com/es-pc/productos/hmi-y-pc-industriales

Pickdata. (11 de Mayo de 2020). *PickData*. Obtenido de Node-Red, La herramienta de programación visual el internet of things: https://www.pickdata.net/es/noticias/node-red-programacion-visual-iot

PROGRAMACIÓNSIEMENS.COM. (s.f.). Obtenido de PROGRAMACIÓNSIEMENS.COM: https://programacionsiemens.com/snap7-como-crear-un-hmi-en-un-pc-sin-opc/

Raspberry. (2024). *Raspberry*. Obtenido de Rasberry: https://www.raspberrypi.org/

Raspberry Pi. (22 de Noviembre de 2023). Obtenido de Raspberyy Pi: https://www.raspberrypi.com/news/raspberry-pi-for-industrial-applications/

Rowe, D. G. (11 de Julio de 2007). *Robotic Farmer*. Obtenido de Robotic Farmer: https://www.technologyreview.com/2007/07/11/224665/robotic-farmer/

SIEMENS . (2025). Obtenido de SIEMENS: https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/controladores-simatic/simatic-s7-1500.html

SIEMENS. (2025). Obtenido de SIEMENS: https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/controladores-simatic/simatic-s7-1500.html

SIEMENS. (2025). Obtenido de SIEMENS: https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/controladores-simatic/simatic-s7-1500.html

SIEMENS. (2025). Obtenido de SIEMENES : https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/controladores-simatic/simatic-s7-1500.html

Viso.ai. (30 de Octubre de 2023). Obtenido de Viso.ai: https://viso.ai/computer-vision/mediapipe/

WIT AUTOMATIZACIÓN. (s.f.). Obtenido de WIT AUTOMATIZACIÓN: https://witautomatizacion.es/que-es-tia-portal-y-para-que-sirve/#:~:text=Totally%20Integrated%20Automatioin%20Portal%20o,frecuencia%2C%20Servomotores%2C%20entre%20otros.

ANEXOS

Script collect_images.py: Es el archivo de código Python con extensión py para la recolección de imágenes para el dataset. Este archivo inicializa la cámara con la librería OpenCv Python. Posteriormente, procede a capturar las imágenes y almacenarlas en los directorios como fue mencionado en la elaboración del modelo de reconocimiento de gestos del marco metodológico. Se puede apreciar las líneas de código siguiente del archivo mencionado:

```
import os

import cv2


DATA_DIR = './data'
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)

number_of_classes = 6
dataset_size = 100

cap = cv2.VideoCapture(0)
for j in range(number_of_classes):
    if not os.path.exists(os.path.join(DATA_DIR, str(j))):
        os.makedirs(os.path.join(DATA_DIR, str(j)))

    print('Collecting data for class {}'.format(j))

    done = False
    while True:
        ret, frame = cap.read()
        cv2.putText(frame, 'Ready? Press "Q" ! :)', (100, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,
                    cv2.LINE_AA)
        cv2.imshow('frame', frame)
        if cv2.waitKey(25) == ord('q'):
            break

    counter = 0
    while counter < dataset_size:
        ret, frame = cap.read()
        cv2.imshow('frame', frame)
        cv2.waitKey(25)
        cv2.imwrite(os.path.join(DATA_DIR, str(j),
'{}.jpg'.format(counter)), frame)
```

```
        counter += 1

cap.release()
cv2.destroyAllWindows()
```

#Código collect_images.py:

(Computervisioneng, 2023)

Script create_dataset.py: Las líneas de código de este archivo Python con extensión py, es útil para la creación del dataset. Este archivo procede a comprimir los directorios creados de las muestras como fue mencionado en la elaboración del modelo de reconocimiento de gestos del marco metodológico**.** Se presenta las líneas de código de dicho archivo:

```
import os
import pickle

import mediapipe as mp
import cv2
import matplotlib.pyplot as plt


mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True,
min_detection_confidence=0.3)

DATA_DIR = './data'

data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []

        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)

                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))

            data.append(data_aux)
            labels.append(dir_)

f = open('data.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

Script train_classifier.py: Lo que presenta este archivo Python con extensión py, es el entrenamiento del modelo, para poder obtener un archivo que tenga el dataset ya entrenado, y comprimido. La elaboración del modelo de reconocimiento de gestos del marco metodológico describe la funcionalidad de este archivo y el archivo que genera con su ejecución. Se logra visualizar las líneas de código presentes en el archivo descrito:

```
import pickle

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np


data_dict = pickle.load(open('./data.pickle', 'rb'))

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])
```

```
x_train, x_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, shuffle=True, stratify=labels)

model = RandomForestClassifier()

model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)

print('{}% of samples were classified correctly !'.format(score *
100))

f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()
```

Script inference_classifier.py: El archivo de Python con extensión py, es aquel archivo que permite el reconocimiento de gestos y la ejecución del modelo generado después del entramiento. Este archivo dependerá del archivo model con extensión p para su ejecución como se presentó en la elaboración del modelo de reconocimiento de gestos del marco metodológico. En las líneas siguientes se presente las líneas de código del archivo mencionado:

```
import pickle
import cv2
import mediapipe as mp
import numpy as np

# Load the trained model
model_dict = pickle.load(open('./model.p', 'rb'))
model = model_dict['model']

# Initialize the video capture (0 = default webcam)
cap = cv2.VideoCapture(0)

# Initialize Mediapipe Hands
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
```

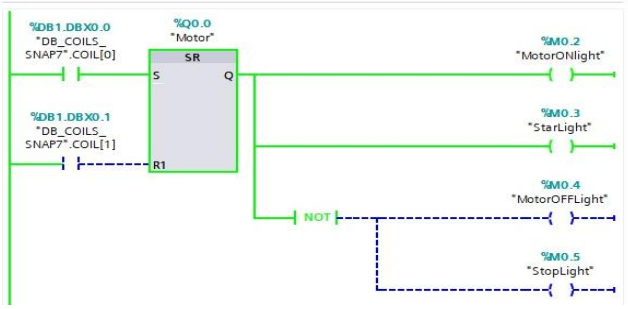
```
hands = mp_hands.Hands(static_image_mode=True,
min_detection_confidence=0.3)

# Label dictionary for predictions
labels_dict = {0: 'Marcha', 1: 'Paro', 2:'Giro Horario', 3:'Giro
Antihorario', 4:'Aumentar Velocidad', 5: 'Disminuir Velocidad'}

while True:
    data_aux = []
    x_ = []
    y_ = []

    ret, frame = cap.read()

    if not ret:
        print("Failed to capture frame. Exiting...")
        break

    H, W, _ = frame.shape

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(frame_rgb)
    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(
                frame,  # image to draw
                hand_landmarks,  # model output
                mp_hands.HAND_CONNECTIONS,  # hand connections
                mp_drawing_styles.get_default_hand_landmarks_style()
,
                mp_drawing_styles.get_default_hand_connections_style
()
            )

        for hand_landmarks in results.multi_hand_landmarks:
            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y

                x_.append(x)
                y_.append(y)

            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                data_aux.append(x - min(x_))
```

```
                data_aux.append(y - min(y_))

        x1 = int(min(x_) * W) - 10
        y1 = int(min(y_) * H) - 10

        x2 = int(max(x_) * W) - 10
        y2 = int(max(y_) * H) - 10

        prediction = model.predict([np.asarray(data_aux)])

        predicted_character = labels_dict[int(prediction[0])]

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
        cv2.putText(frame, predicted_character, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3,
                    cv2.LINE_AA)

    cv2.imshow('frame', frame)

    # Check if the ESC key is pressed or the window is closed
    if cv2.waitKey(1) & 0xFF == 27:  # ESC key (ASCII 27)
        print("ESC key pressed. Exiting...")
        break

    # If the window is closed manually, break the loop
    if cv2.getWindowProperty('frame', cv2.WND_PROP_VISIBLE) < 1:
        print("Window closed manually. Exiting...")
        break

# Release the camera and close OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

Para la ejecución y puesta en marcha del motor, se elaboró un código en lenguaje de programación LADDER el cual activaba las entradas y salidas correspondientes en el PLC, como así fue mencionado en el diseño funcional del sistema del marco metodológico. En la figura 31 se puede observar el segmento generado en TIA Portal para la puesta en marcha del motor.

Figura 31

Segmento de marcha y paro extraído de TIA Portal.

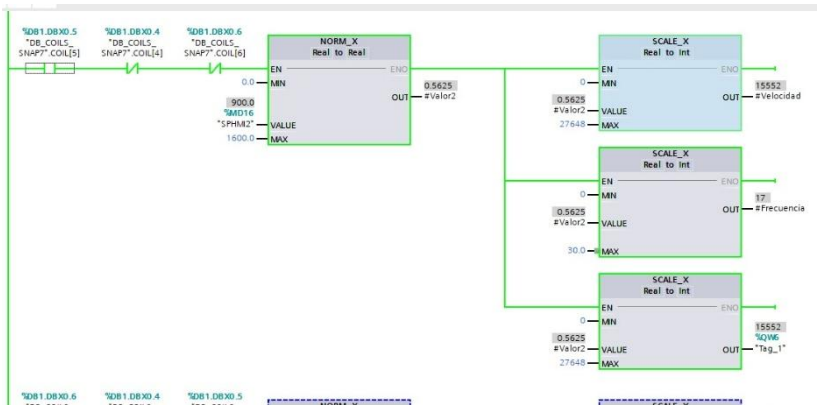


Nota: Segmento de marcha y paro extraído del código de TIA Portal, el cual acciona el motor, detiene el motor, y prende la luz de encendido.

El escalamiento de velocidades para el cambio de frecuencia en el motor fue elaborado con el segmento de programación LADDER utilizando los bloques NORM_X y SCALE_X detallado en el Diseño funcional del sistema del Marco metodológico. Se observa en la figura 32 el segmento mencionado.

Figura 32

Segmento de velocidades extraído de TIA Portal.



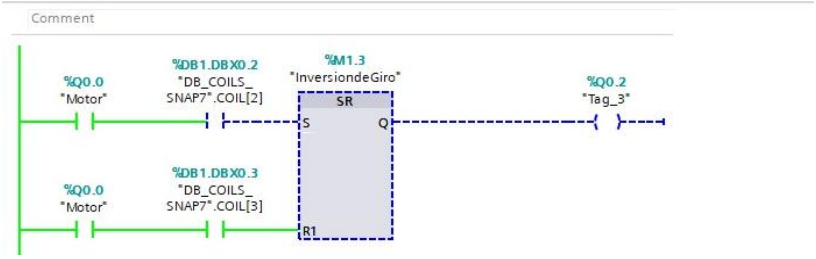
Nota: Segmento de escalamiento de velocidades extraído del código de TIA Portal, el cual permite que el motor cambie sus velocidades giro a horario o a antihorario respectivamente.

La inversión de giro para la rotación del motor tuvo su elaboración con el segmento de programación LADDER utilizando el bloque SR, detallado en el Diseño funcional del sistema del Marco metodológico. Dicho segmento permite activar y desactivar las entradas

y salidas correspondientes para ejecutar el giro en la orientación indicada. La figura 33 ilustra el segmento de inversión de giro.

Figura 33

Segmento de inversión de giro del código LADDER de TIA Portal.



Nota: Segmento de escalamiento de velocidades extraído del código de TIA Portal, el cual permite que el motor cambie su giro a horario o a antihorario respectivamente.

El siguiente código ejecuta el envío de señales con Python utilizando la terminal de la computadora, al PLC con SNAP7. La ejecución de acciones con el motor es puestas en marcha con este script sin la utilización del software TIA Portal directamente. Para ejecutar este script es necesario tener el código en formato LADDER ya subido al PLC previamente. Preferiblemente se recomienda ejecutar primero el archivo pruebaComunicacionConPLC.py para probar la comunicación, y luego si, ejecutar este script.

```
# Ejecutar primero este código preferiblemente para verificar la
comunicación.
import snap7

# Configuración del PLC
plc = snap7.client.Client()
plc.connect('192.168.1.10', 0, 1)  # Cambia la IP, Rack y Slot según
tu PLC

if plc.get_connected():
    print("Conexión exitosa con el PLC.")

    # Leer 1 byte del DB1, dirección inicial 0
    db_number = 1
    start_address = 0
    size = 1  # Leer 1 byte
```

```
    data = plc.db_read(db_number, start_address, size)
    print(f"Datos leídos desde DB{db_number}: {data}")

    plc.disconnect()
else:
    print("No se pudo conectar al PLC.")
```

```
# Preferiblemente ejecutar primero el archivo
pruebaComunicacionConPLC.py para probar la comunicación.

import snap7
from snap7.util import set_bool, set_int
from snap7.types import Areas

class ControlDeMotor:
    def _init_(self, plc):
        self.estado = 'enParo'
        self.direccion = None
        self.velocidad = 1  # Velocidad por defecto
        self.plc = plc  # Conexión al PLC

    def marcha(self):
        if self.estado == 'enParo':
            self.estado = 'enMarcha'
            self.direccion = 'horario'  # Dirección por defecto
            self.escribir_booleano(0, True)  # Activar posición
DBX0.0 para marcha
            print("Motor iniciado.")
        else:
            print("El motor ya está en marcha.")

    def paro(self):
        if self.estado == 'enMarcha':
            self.estado = 'enParo'
            self.escribir_booleano(1, True)  # Activar posición
DBX0.1 para paro
            print("Motor detenido.")
        else:
            print("El motor ya está detenido.")

    def giroHorario(self):
        if self.estado == 'enMarcha':
            if self.direccion == 'horario':
                print("El motor ya está girando en sentido
horario.")
```

```
            else:
                self.direccion = 'horario'
                self.escribir_booleano(2, True)  # Activar posición
DBX0.2 para giro horario
                print("Motor girando en sentido horario.")
        else:
            print("No se puede girar el motor, está detenido.")

    def giroAntihorario(self):
        if self.estado == 'enMarcha':
            if self.direccion == 'antihorario':
                print("El motor ya está girando en sentido
antihorario.")
            else:
                self.direccion = 'antihorario'
                self.escribir_booleano(3, True)  # Activar posición
DBX0.3 para giro antihorario
                print("Motor girando en sentido antihorario.")
        else:
            print("No se puede girar el motor, está detenido.")

    def establecerVelocidad(self):
        # Desactivar todas las posiciones de velocidad antes de
establecer una nueva
        self.escribir_booleano(4, False)
        self.escribir_booleano(5, False)
        self.escribir_booleano(6, False)

        if self.velocidad == 1:
            self.escribir_booleano(4, True)  # Activar posición
DBX0.4 para velocidad 1
        elif self.velocidad == 2:
            self.escribir_booleano(5, True)  # Activar posición
DBX0.5 para velocidad 2
        elif self.velocidad == 3:
            self.escribir_booleano(6, True)  # Activar posición
DBX0.6 para velocidad 3

        print(f"Velocidad establecida a nivel {self.velocidad}.")

    def aumentoVelocidad(self):
        if self.estado == 'enMarcha':
            if self.velocidad < 3:
                self.velocidad += 1
                self.establecerVelocidad()
            else:
```

```
                print("La velocidad ya está en el nivel máximo
(3).")
        else:
            print("No se puede aumentar la velocidad, el motor está
detenido.")

    def reducirVelocidad(self):
        if self.estado == 'enMarcha':
            if self.velocidad > 1:
                self.velocidad -= 1
                self.establecerVelocidad()
            else:
                print("La velocidad ya está en el nivel más bajo
(1).")
        else:
            print("No se puede reducir la velocidad, el motor está
detenido.")

    def escribir_booleano(self, posicion, valor):
        # Leer datos del DB1
        data = self.plc.read_area(Areas.DB, 1, 0, 1)  # Leer 1 byte
del DB1
        set_bool(data, 0, posicion, valor)  # Modificar el bit
específico
        self.plc.write_area(Areas.DB, 1, 0, data)  # Escribir de
nuevo en el DB1

def main():
    # Configurar el cliente Snap7 para conectar al PLC
    plc = snap7.client.Client()
    plc.connect('172.18.135.61', 0, 1)  # IP del PLC, Rack 0, Slot 1

    if plc.get_connected():
        print("Conexión establecida con el PLC.")

        motor = ControlDeMotor(plc)

        while True:
            print("\nMenú de control del motor:")
            if motor.estado == 'enParo':
                print("1. Iniciar motor")
            else:
                print("2. Detener motor")
                print("3. Girar sentido horario")
                print("4. Girar sentido antihorario")
                print("5. Aumentar velocidad")
                print("6. Disminuir velocidad")
```

```
            print("7. Salir")

            opcion = input("Seleccione una opción (1-7): ")

            if opcion == '1' and motor.estado == 'enParo':
                motor.marcha()
            elif opcion == '2' and motor.estado == 'enMarcha':
                motor.paro()
            elif opcion == '3' and motor.estado == 'enMarcha':
                motor.giroHorario()
            elif opcion == '4' and motor.estado == 'enMarcha':
                motor.giroAntihorario()
            elif opcion == '5' and motor.estado == 'enMarcha':
                motor.aumentoVelocidad()
            elif opcion == '6' and motor.estado == 'enMarcha':
                motor.reducirVelocidad()
            elif opcion == '7':
                print("Saliendo del programa.")
                break
            else:
                print("Opción no válida o el motor está detenido.
Inténtelo de nuevo.")

        plc.disconnect()
    else:
        print("No se pudo establecer conexión con el PLC.")

if __name__ == "__main__":
    main()
```

El siguiente código ejecuta el envío de gestos con el modelo de visión artificial al PLC con SNAP7 para la ejecución de acciones con el motor y representación en el HMI. Este script generado con extensión Python, es decir py, no solo será el que corra el modelo de visión artificial, sino también, será el que le envíe las señales al PLC para cambiar los valores booleanos, según corresponda, y ejecutar los procesos descritos en el marco metodológico en el motor, el HMI y el mando y señalización.

```
import cv2
import mediapipe as mp
import numpy as np
import pickle
import snap7
import time
```

```
# Configuración del PLC.
PLC_IP = '172.18.135.61'
PLC_RACK = 0
PLC_SLOT = 1 # Slot del PLC utilizado para la red, en este caso X1.
DB_NUMBER = 1  # Número de DB donde están los coils.

# Configuración de los gestos y modelo.
GESTURE_DELAY = 5  # Tiempo de espera entre comandos en segundos.
Modificable.
labels_dict = { #Etiquetas del modelo.
    0: 'Marcha',
    1: 'Paro',
    2: 'Giro Antihorario',
    3: 'Giro Horario',
    4: 'Aumentar Velocidad',
    5: 'Disminuir Velocidad'
}
last_gesture = None  # Variable que almacena el ultimo gesto
procesado.
last_sent_time = time.time()  # Registro del último envío al PLC
current_speed = 1  # Velocidad inicial

# Cargar el modelo entrenado
try:
    model_dict = pickle.load(open('./model.p', 'rb'))
    model = model_dict['model']
except Exception as e:
    print(f"Error al cargar el modelo: {e}")
    exit()

# Inicializar cámara
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: No se pudo abrir la cámara")
    exit()

# Inicializar MediaPipe Hands
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands = mp_hands.Hands(static_image_mode=False,
min_detection_confidence=0.5)

# Conectar al PLC
client = snap7.client.Client()
try:
```

```
    client.connect(PLC_IP, PLC_RACK, PLC_SLOT)
    print("Conectado al PLC exitosamente.")
except Exception as e:
    print(f"Error conectando al PLC: {e}")
    cap.release()
    cv2.destroyAllWindows()
    exit()

def send_to_plc(gesture):
    """Envía el comando al PLC según el gesto detectado."""
    global current_speed  # Acceder a la variable de velocidad.

    # Gestos traducidos a bits para el envio de datos.
    if gesture == 'Marcha':
        data = bytearray([0b00000001])  # Bit 0 = True (Marcha), Bit
1 = False (Paro).
    elif gesture == 'Paro':
        data = bytearray([0b00000010])  # Bit 0 = False (Marcha),
Bit 1 = True (Paro).
    elif gesture == 'Giro Antihorario':
        data = bytearray([0b00000100])  # Activar giro antihorario.
    elif gesture == 'Giro Horario':
        data = bytearray([0b00001000])  # Activar giro horario.
    elif gesture == 'Aumentar Velocidad':
        if current_speed < 3:
            current_speed += 1
        # Enviar comando de velocidad.
        data = bytearray([0b00000000 | (1 << (current_speed +
3))])  # Bit 4-6 para velocidades.
        print(f"Velocidad actual: {current_speed}")  # Imprimir
velocidad al ejecutar el gesto.
    elif gesture == 'Disminuir Velocidad':
        if current_speed > 1:
            current_speed -= 1
        # Enviar comando de velocidad.
        data = bytearray([0b00000000 | (1 << (current_speed +
3))])  # Bit 4-6 para velocidades.
        print(f"Velocidad actual: {current_speed}")  # Imprimir
velocidad al ejecutar el gesto.
    else:
        return

    try:
        client.db_write(DB_NUMBER, 0, data)  # Escribir en el DB1,
posición 0.
        print(f"Gesto '{gesture}' enviado al PLC.")
    except Exception as e:
```

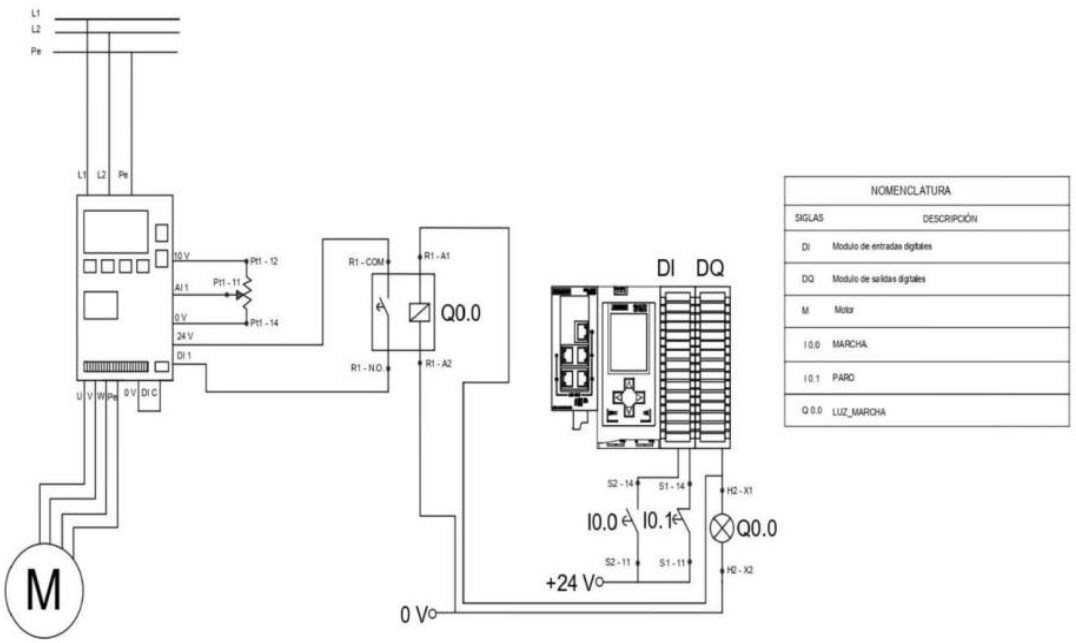
```
        print(f"Error enviando al PLC: {e}")

# Ciclo principal para detección de gestos.
while True:
    ret, frame = cap.read()
    if not ret:
        print("Error al capturar la cámara. Saliendo...")
        break

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            # Dibujar los puntos de la mano en el frame.
            mp_drawing.draw_landmarks(
                frame, hand_landmarks, mp_hands.HAND_CONNECTIONS,
                mp_drawing_styles.get_default_hand_landmarks_style()
,
                mp_drawing_styles.get_default_hand_connections_style
()
            )

            # Extraer coordenadas de la mano para predicción.
            x_, y_, data_aux = [], [], []
            for landmark in hand_landmarks.landmark:
                x_.append(landmark.x)
                y_.append(landmark.y)

            for landmark in hand_landmarks.landmark:
                data_aux.append(landmark.x - min(x_))
                data_aux.append(landmark.y - min(y_))

            # Hacer la predicción con el modelo.
            try:
                prediction = model.predict([np.asarray(data_aux)])
                predicted_class = int(prediction[0])
                predicted_label = labels_dict[predicted_class]
            except Exception as e:
                print(f"Error en la predicción del gesto: {e}")
                continue

            # Evitar enviar el mismo gesto repetidamente y con el
delay de 5 segundos.
            current_time = time.time()
            if (current_time - last_sent_time) > GESTURE_DELAY:
                send_to_plc(predicted_label)
```

```
                last_sent_time = current_time
                last_gesture = predicted_label

            # Mostrar el gesto detectado en el frame.
            cv2.putText(frame, f"Gesto detectado:
{predicted_label}",
                        (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
0, 0), 2, cv2.LINE_AA)

    # Mostrar el frame en una ventana.
    cv2.imshow('Gesto detectado', frame)

    # Salir con la tecla ESC.
    if cv2.waitKey(1) & 0xFF == 27:
        print("Saliendo...")
        break

# Liberar recursos y enviar comando "Paro" al PLC para resetear las
coils.
send_to_plc('Paro')  # Asegura que el motor se detenga antes de
salir.
cap.release()
cv2.destroyAllWindows()
client.disconnect()
print("Recursos liberados y desconexión del PLC completada.")
```

En la figura 34 se muestra el esquema de control de un motor trifásico mediante un variador de frecuencia y un PLC. El sistema permite realizar las funciones de marcha, paro e inversión de giro. Para ello, se utilizan entradas y salidas digitales del PLC que controlan tanto el estado del motor como las señales hacia el variador. Además, el esquema incluye un relé encargado de ejecutar la inversión de giro. Este control eficiente permite la automatización del proceso, optimizando la gestión del motor. Para una mejor comprensión de las conexiones y su funcionalidad, se invita a revisar la figura 34, donde se detalla el flujo de señales y los componentes involucrados en el sistema.

Figura 34

Diagrama de fuerza y control práctica 5 de Automatización Industrial 2.

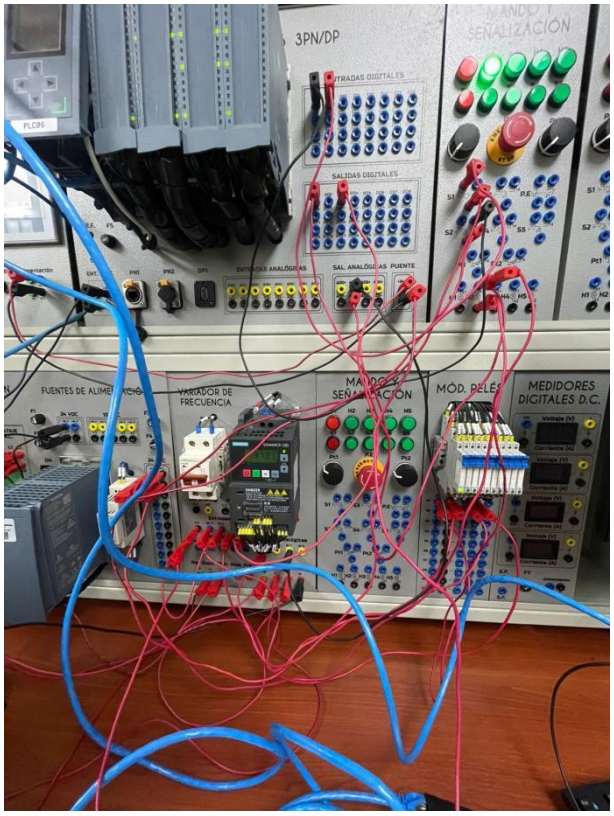


Nota: Esquema de conexiones del diagrama de fuerza y control utilizado para realizar tanto la puesta en marcha, como la inversión de giro del motor.

A continuación, se presenta la configuración de conexiones utilizadas en el desarrollo del presente proyecto, las cuales integran diversos componentes esenciales como fuentes de alimentación, variador de frecuencia y entradas digitales del PLC. La fuente de 24VDC suministra energía para mando y señalización, mientras que las conexiones de los módulos de relés y entradas analógicas aseguran la correcta operación del sistema. Para una mejor comprensión de estas conexiones, se invita al lector a revisar la figura 35, donde se muestra el diagrama detallado del circuito implementado y su interacción entre los distintos dispositivos.

Figura 35

Conexiones para ejecutar los procesos descritos.



Nota: Conexiones realizadas para ejecutar los procesos con el motor adicionadas las velocidades y conexión tanto con el relé utilizado y el variador de frecuencia.

La siguiente licencia corresponde al modelo de visión artificial utilizado para posteriormente su modificación y elaboración del diseño de implementación de un proceso industrial accionado mediante gestos usando PLC, HMI, y un ordenador monoplaca. Se recalca que todos sus archivos usados están citados en este libro y pueden ser encontrados como la licencia APACHE 2.0 lo rige.

MIT License

Copyright (c) 2023 Computer vision developer

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

(Computervisioneng, 2023)