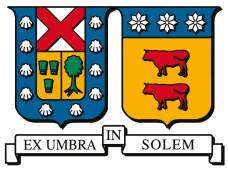
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE CONCEPCIÓN – REY BALDUINO DE BÉLGICA.



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Análisis de Soldadura por Visión Artificial Robot FANUC.

Memoria de titulación para optar al Título de Ingeniero de Ejecución en Control e Instrumentación Industrial.

Autores:
Cristóbal Antonio Díaz Fica
Abner Gabriel Rodríguez Fernández

Profesor guía:
MSc. Felipe Benavides Pantoja

Índice

Agradecimientos VI

Agradecimientos Personales VII

Resumen IX

Capítulo 1: Definición y Conceptos Generales de Soldadura Robotizada 1
1.1 Introducción . . . 1
1.2 Soldadura Convencional y Robotizada MIG/MAG . . . 2
1.3 Ventajas y Desventajas Soldadura MIG/MAG Robotizada . . . 2
Ventajas . . . 2
Desventajas . . . 2
1.4 Elementos Generales Soldadura Robotizada . . . 3
Brazo Robótico . . . 3
Controlador Brazo Robótico . . . 3
Equipo de Soldadura MIG/MAG . . . 4
Sistema de Alimentación de Alambre . . . 4
Software de Programación . . . 5
Gas protector . . . 5
Implementos de Seguridad . . . 6
I. Objetivo General . . . 7
II. Objetivos Específicos . . . 7
III. Alcances del Proyecto . . . 7
IV. Restricciones . . . 8

Capítulo 2: Comunicación y Redes 9
2.1 Diagrama Comunicación y Redes . . . 9
2.2 Brazo Robótico y Controlador . . . 10
Brazo Robótico . . . 10
Controlador R-30iB y Teach Pendant (TP) . . . 11
RI-3000 . . . 12
2.3 Equipo de Soldadura . . . 12
Equipo Selco Genesis 3200 MSE . . . 12
Selco WFR 1000 Classic . . . 13
Rollo de Soldadura . . . 13
Gas protector . . . 14
Antorcha de soldadura . . . 14
2.4 Procesamiento de imágenes . . . 15
Cámara SC130EF2 B/W . . . 15
Equipo de procesamiento de imágenes . . . 16

Capítulo 3: Integración Protocolo FTP 17
3.1 Diagrama Red FTP . . . 17

3.2 Planteamiento de Conexiones y Montaje . . . 18
Conexión al Controlador R-30iB Plus . . . 18
Montaje de la Cámara a Brazo Robótico . . . 19
3.3 Integración de los Componentes . . . 20
Realizar Conexiones en Controlador . . . 20
Asignación Direcciones IP . . . 21
Configuración Servidor FTP . . . 22
3.4 Desarrollo y Montaje de Protección Cámara de Visión . . . 24
Desarrollo de Protección y Soporte . . . 24
Montaje Cámara y Soporte . . . 25
3.5 Configuración y Calibración de Cámara . . . 26
Configuración Cámara . . . 26
Posición Robot con Cámara Montada . . . 27
Calibración User Frame en Brazo robótico . . . 28
Calibración de Cámara en iRVision de FANUC . . . 28
3.6 Captura y Transferencia de Imágenes . . . 29
Código en Brazo Robótico para captura de imágenes . . . 29
Filezilla para la visualización de imágenes y rutas . . . 31
Extracción de imágenes mediante Python . . . 31
3.7 Dificultades Técnicas . . . 33
Dificultades Técnicas Servidor FTP . . . 33

Capítulo 4: Visión Artificial **36**
4.1 Definición de Visión artificial . . . 36
4.2 Conceptos Básicos . . . 37
4.3 Aplicaciones en la Industria . . . 37
4.4 Introducción a OpenCV . . . 38
4.5 Procesamiento de Imágenes con OpenCV y Python . . . 38
4.6 Comparación entre el Filtro de Canny y Laplace . . . 39
4.7 Procesamiento de Imagen . . . 40
Análisis de Imagen con OpenCV . . . 40
Identificación de Puntos de Trayectoria . . . 42

Capítulo 5: Integración General **45**
5.1 Secuencia del Proceso . . . 45
5.2 Descripción General del Proceso . . . 46
5.3 Secuencia del Proceso Automatizado . . . 46
5.4 Coordinación de Componentes y Software . . . 47
5.5 Consideraciones Técnicas . . . 47

Capítulo 6: Resultados y Conclusiones **48**
6.1 Resultados Obtenidos . . . 48
6.2 Análisis de Resultados . . . 51

Reducción de Tiempos Muertos **51**

Precisión en la Soldadura 52
Ventajas y Desventajas del Sistema Implementado 52
6.3 Recomendaciones para Trabajos Futuros 53

Conclusiones **54**

Anexos **55**
Anexo 1: Configuración del Sistema de Visión, Calibración de Cámara y Proceso de Visión 55
Calibración de Cámara 55
Proceso de Visión 60
Anexo 2: Instalación de Software y Librerías 64
Instalación de lenguaje Python 64
Instalación y configuración de OpenCV y librerías adicionales 65
Instalación de PC Interface en controlador 66
Instalación de PCDK y Roboguide 67
Anexo 3: Código Brazo Robótico 68
Anexo 4: Código General de Python 70
Anexo 5: Manual iRVision 2D R-30iB Plus 79
Anexo 6: Manual de Operador R-30iB Funciones Ethernet 80
Anexo 7: Manual de Programación y Operación de Robots FANUC 81
Anexo 8: Manual de Operador R-30iB Plus Ethernet/IP 82
Anexo 9: Planos Soporte y Cubierta Protectora 83

Índice de figuras

Figura 1.1: Brazo Robótico . . . 3
Figura 1.2: Controlador Brazo Robótico . . . 3
Figura 1.3: Equipo soldadura MIG-MAG . . . 4
Figura 1.4: Alimentación de Alambre . . . 4
Figura 1.5: Programación Teach Pendant . . . 5
Figura 1.6: Gas protector . . . 5
Figura 1.7: Implementos de seguridad . . . 6
Figura 2.1: Diagrama General . . . 9
Figura 2.2: Brazo Robótico Arc Mate 100iD . . . 10
Figura 2.3: Controlador R-30iB Plus . . . 11
Figura 2.4: Teach Pendant . . . 11
Figura 2.5: RI-3000 . . . 12
Figura 2.6: Selco Genesis 3200 MSE . . . 12
Figura 2.7: Alimentador Selco WFR 1000 Classic . . . 13
Figura 2.8: Rollo de 15Kg soldadura ER70S-6 . . . 13
Figura 2.9: Cilindro de gas (Argón) . . . 14
Figura 2.10: Antorcha de soldadura . . . 14
Figura 2.11: Cámara SC130EF2 B/W . . . 15
Figura 2.12: Equipo Procesamiento de imágenes . . . 16
Figura 3.1: Diagrama Red FTP . . . 17
Figura 3.2: Puertos Controlador CD38-A y CD38-B . . . 18
Figura 3.3: Cámara Montada en Brazo Robótico . . . 19
Figura 3.4: Conexiones CD38-A y CD-38-B . . . 20
Figura 3.5: Conexión Cámara . . . 20
Figura 3.6: Dirección IP Puerto 1 (CD38-A) . . . 21
Figura 3.7: Dirección IP Puerto 2 (CD38-B) . . . 21
Figura 3.8: Dirección IP Estática PC . . . 21
Figura 3.9: Seleccionar Servidores . . . 22
Figura 3.10: Servidores . . . 22
Figura 3.11: Configuraciones Servidor FTP . . . 23
Figura 3.12: Impresora Anycubic Kobra S1 Combo . . . 24
Figura 3.13: Modelo 3D Cubierta Protectora . . . 25
Figura 3.14: Prototipo impreso en 3D . . . 25
Figura 3.15: Soporte y cubierta montada en brazo robótico . . . 25
Figura 3.16: Cámara de visión Artificial . . . 26
Figura 3.17: Configuración Almacenamiento de imágenes . . . 26
Figura 3.18: Imagen de cámara en posición fija . . . 27
Figura 3.19: Grid de Calibración . . . 28
Figura 3.20: Instrucción de Visión . . . 29
Figura 3.21: Instrucción RUN FIND . . . 30
Figura 3.22: Proceso de Visión . . . 30
Figura 3.23: Software Filezilla . . . 31
Figura 3.24: Establecer conexión Servidor FTP . . . 31

Figura 3.25: Identificación de último directorio 32
Figura 3.26: Filtro de imágenes 32
Figura 3.27: Descarga de imagen más reciente 32
Figura 3.28: Error servidor FTP 33
Figura 3.29: Servidores FTP sin iniciar 33
Figura 3.30: Inicializar Servidor FTP 34
Figura 3.31: Error al iniciar Servidor FTP 34
Figura 3.32: Versión antigua de firmware 35
Figura 3.33: Versión actual de firmware 35
Figura 3.34: Servidores iniciados correctamente 35
Figura 4.1: Aplicaciones en la Industria 37
Figura 4.2: Filtro canny vs Laplace 39
Figura 4.3: Filtro gausiano y filtro de mediana 40
Figura 4.4: Extracción de fondo de imagen y operación morfológica 40
Figura 4.5: Ajustes de la máscara para tonalidades de blanco 41
Figura 4.6: Instrucción para aplicar la máscara 41
Figura 4.7: Instrucción de función Canny con imagen sin fondo 41
Figura 4.8: Función transformada de Hough 42
Figura 4.9: Validación detección de líneas 42
Figura 4.10: Imprime los puntos extremos de cada línea 43
Figura 4.11: Agrupa puntos extremos 43
Figura 4.12: Cálculo promedio de puntos extremos 44
Figura 4.13: Cálculo de la pendiente 44
Figura 4.14: Resultado de puntos de soldadura 44
Figura 5.1: Diagrama de flujo del proceso 45
Figura 6.1: Resultado primera prueba 48
Figura 6.2: Resultado segunda prueba 49
Figura 6.3: Resultado tercera prueba 50
Figura 7.1: Vision Setup 55
Figura 7.2: Camera Setup 56
Figura 7.3: Calibration Setup 56
Figura 7.4: Fixture Pos. Setting 57
Figura 7.5: Resultados Fixture Pos Setting 57
Figura 7.6: Configuración de calibración 58
Figura 7.7: Seleccionar área del grid 58
Figura 7.8: Resultados de puntos de calibración 59
Figura 7.9: Puntos de calibración 59
Figura 7.10: Resultados de calibración 60
Figura 7.11: Seleccionar visión de proceso 61
Figura 7.12: Configuraciones iniciales 61
Figura 7.13: Configuración Snap tool 1 62
Figura 7.14: Delimitación objeto de referencia 62
Figura 7.15: Guardar configuración 63

Agradecimientos

Deseamos manifestar nuestro más profundo agradecimiento a todas las personas que, con su compromiso y apoyo, hicieron posible la elaboración de esta memoria.

En primer lugar, queremos reconocer a los profesores que nos acompañaron durante estos tres años de formación académica. Su dedicación, paciencia y entrega en el proceso educativo nos permitieron adquirir las competencias necesarias para nuestro desarrollo profesional.

Finalmente, extendemos un especial agradecimiento al apoyo docente de la Universidad, Matías Cid Zapata, quien, con su experiencia, disposición y profesionalismo, nos brindó un apoyo clave para llevar a cabo este trabajo de manera exitosa. Su colaboración fue fundamental para alcanzar los objetivos propuestos.

A todos ustedes, les expresamos nuestra más sincera gratitud.

Agradecimientos Personales

Autor: Cristóbal Antonio Díaz Fica

A mi familia, quiero agradecerles profundamente por haber sido mi base, mi refugio y mi fuerza en cada etapa de este camino. Gracias por creer en mí incluso antes de que yo mismo lo hiciera, por acompañarme en silencio cuando lo necesitaba y por estar presentes, una y otra vez, con palabras de aliento, gestos de amor y una confianza que nunca flaqueó. Su apoyo incondicional no solo me sostuvo, sino que me inspiró a seguir avanzando con determinación, aun en los momentos más difíciles. Cada logro, cada paso, cada obstáculo superado, lleva parte de ustedes.

A lo largo de estos años, no siempre fue fácil. Hubo cansancio, incertidumbre y dudas. Pero en todo ese proceso, mi familia fue una constante: un recordatorio de por qué vale la pena esforzarse y de lo mucho que significa tener personas que caminan a tu lado, sin pedir nada a cambio, solo deseando verte cumplir tus sueños. Su confianza en mí me dio el impulso necesario para no rendirme y para seguir adelante, con la certeza de que no estaba solo.

A mi hermano, especialmente, quiero agradecerle por abrirme su hogar en una etapa clave del proceso. Ese gesto, que puede parecer sencillo, significó muchísimo para mí. Me permitió tener un espacio de calma, un entorno lleno de cariño donde pude enfocarme, recargar energías y continuar con la concentración que requería el cierre de este trabajo. Más allá del lugar físico, fue el ambiente familiar, la conversación diaria, la calidez compartida, lo que me ayudó a mantener el equilibrio y a seguir adelante con esperanza y motivación. Su apoyo fue un regalo invaluable, que siempre llevaré conmigo.

Hoy, al terminar esta etapa, tengo muy claro que no llegué hasta aquí solo. Cada esfuerzo que hice estuvo acompañado por el cariño, la paciencia y el apoyo constante de mi familia. Todo lo que logré también es parte de ustedes. Gracias por estar, por levantarme cuando lo necesité y por nunca soltarme la mano. Este logro es tan suyo como mío.

Atentamente,
Cristóbal Antonio Díaz Fica

Autor: Abner Gabriel Rodríguez Fernández

Quiero expresar mi más profundo agradecimiento a quienes han sido un pilar fundamental en mi vida y en todo este camino académico.

A mis padres, por brindarme siempre su amor incondicional, por su esfuerzo incansable y por enseñarme con el ejemplo el valor del trabajo, la responsabilidad y la perseverancia. Gracias por creer en mí incluso cuando yo mismo dudaba, por estar presentes en cada paso y por sostenerme en los momentos más difíciles.

A mi hermano, por su apoyo constante, por ayudarme con todo lo necesario durante mis viajes y estadías de estudio, y por ofrecerme siempre un espacio para desconectarme y recargar energías en los momentos en que más lo necesitaba.

A mis abuelos, demás familiares y amigos, gracias por su cariño, consejos y compañía a lo largo de estos años. Su apoyo, ya sea con una palabra de aliento, una conversación sincera o simplemente su presencia, ha sido clave para mantenerme firme en este proceso.

Cada uno de ustedes ha contribuido, desde su lugar, a que hoy pueda cumplir esta importante etapa de mi vida. A todos, gracias por estar, por creer y por caminar junto a mí.

Este logro no es solo mío, sino también de ustedes. Con profundo aprecio y gratitud, les dedico esta memoria.

Atentamente,
Abner Gabriel Rodríguez Fernández

Resumen

La soldadura robotizada ha revolucionado diversos sectores industriales al permitir una automatización precisa y eficiente de los procesos de soldadura. Sin embargo, en la industria, los brazos robóticos enfrentan una problemática: cada nueva pieza que se va a soldar requiere una programación específica para adaptarse a su forma y posición. Esto limita la flexibilidad y aumenta el tiempo de preparación de este proceso.

Este proyecto aborda esta dificultad utilizando una cámara de visión artificial, que permitirá al robot identificar y adaptar dinámicamente su proceso de soldadura, independientemente de la forma o la posición de las piezas que reciba.

El primer capítulo introduce los conceptos fundamentales sobre la soldadura robotizada, destacando sus ventajas y dificultades. En el segundo capítulo, se abordan los aspectos relacionados con la comunicación y redes, necesarios para la integración efectiva de los distintos componentes del sistema. El tercer capítulo se centra en la visión artificial, explicando el uso de herramientas como OpenCV y el filtro de Laplace para procesar imágenes y definir trayectorias de soldadura precisas. Finalmente, el cuarto capítulo describe la integración completa del sistema y presenta los resultados obtenidos, que demuestran una mejora considerable respecto a los métodos tradicionales de soldadura.

Capítulo 1: Definición y Conceptos Generales de Soldadura Robotizada

1.1 Introducción

La automatización industrial ha avanzado significativamente en las últimas décadas, motivada por la necesidad de mejorar la eficiencia y la precisión en los procesos de manufactura. Dentro de este marco, la soldadura robotizada ha emergido como una solución clave para aumentar tanto la productividad como la calidad en la fabricación de componentes metálicos. Esta memoria se enfoca en la implementación de un sistema de soldadura MIG/MAG robotizada utilizando un brazo robótico FANUC, con el objetivo de soldar piezas de manera dinámica e interactiva mediante visión artificial.

La visión artificial permite que los sistemas robóticos interpreten y comprendan su entorno a través de imágenes capturadas por cámaras. En este proyecto, se usará la biblioteca OpenCV para procesar las imágenes y aplicar un filtro para la detección de bordes. Esta técnica facilita la identificación de los puntos de trayectoria del cordón de soldadura, cuyas coordenadas se enviarán al brazo robótico para realizar la soldadura de forma autónoma.

El objetivo principal de este proyecto es reducir los tiempos de programación, los cuales actualmente generan tiempos muertos cuando se soldan piezas de diferentes geometrías con un brazo robótico. La integración de visión artificial permitirá optimizar el proceso de soldadura, mejorando la flexibilidad y adaptabilidad en la fabricación de componentes metálicos. Esto, a su vez, contribuirá a mejorar tanto la eficiencia como la calidad del proceso de soldadura, acelerando el avance de la automatización industrial.

1.2 Soldadura Convencional y Robotizada MIG/MAG

- **Soldadura Convencional**

  La soldadura es un proceso de unión de materiales mediante la aplicación de calor, presión o ambos. Los materiales se funden y se mezclan, formando una unión sólida al enfriarse. En algunos casos, se utiliza un material de aporte para reforzar la unión, y es esencial proteger el área de soldadura de contaminantes atmosféricos. La calidad de la soldadura depende del control preciso de parámetros como la temperatura, la velocidad de soldadura, la corriente y el voltaje.

- **Soldadura MIG/MAG Robotizada**

  La soldadura robotizada MIG/MAG es un proceso preciso y eficiente que utiliza un robot industrial para manipular una antorcha de soldadura. Este proceso emplea un gas de protección (inerte o activo) que crea una atmósfera controlada, evitando la contaminación del baño de soldadura y garantizando una unión de alta calidad. Gracias a su versatilidad, junto con la precisión y velocidad de los robots, esta técnica ha ganado relevancia en industrias que requieren productividad y alta calidad en productos metálicos complejos.

1.3 Ventajas y Desventajas Soldadura MIG/MAG Robotizada

Ventajas

- **Precisión y repetibilidad:** Los robots garantizan soldaduras de alta calidad y consistentes, reduciendo drásticamente los defectos.
- **Velocidad y productividad:** La automatización aumenta significativamente la velocidad de producción, lo que se traduce en menores costos y mayor eficiencia.
- **Flexibilidad:** Los robots pueden adaptarse a diferentes geometrías y posiciones de soldadura, ofreciendo una mayor versatilidad en la fabricación.

Desventajas

- **Costo inicial:** La inversión en un sistema robótico es alta, incluyendo el robot, la programación y la integración.
- **Complejidad de Programación:** La programación de robots requiere conocimientos especializados y puede ser un proceso complejo.
- **Rigidez:** Los robots son menos flexibles a cambios en el diseño o el proceso, lo que puede generar costos adicionales de reprogramación.

1.4 Elementos Generales Soldadura Robotizada

Brazo Robótico

El brazo robótico es el componente central de la celda de soldadura automatizada, encargado de ejecutar los movimientos necesarios para posicionar con precisión la antorcha en los puntos de soldadura definidos. Está diseñado para realizar desplazamientos precisos y repetitivos, asegurando uniformidad y calidad en el proceso.

Gracias a sus seis grados de libertad (articulaciones), puede moverse en múltiples ejes y orientaciones, lo que le brinda la flexibilidad necesaria para adaptarse a diversas geometrías de piezas y configuraciones del entorno, siguiendo trayectorias complejas generadas por visión artificial o mediante programación.



Figura 1.1: Brazo Robótico

Controlador Brazo Robótico

El controlador robótico funciona como el cerebro del sistema, gestionando con precisión las trayectorias, velocidades y aceleraciones del brazo robótico para garantizar movimientos eficientes y seguros. Además, integra y coordina los distintos componentes del sistema, procesando señales externas y datos de dispositivos periféricos como sensores, cámaras y sistemas de seguridad, lo que permite una operación fluida y sincronizada en la celda de trabajo.



Figura 1.2: Controlador Brazo Robótico

Equipo de Soldadura MIG/MAG

El equipo de soldadura MIG/MAG (Metal Inert Gas / Metal Active Gas) es una herramienta esencial en la industria metalúrgica para unir diferentes metales. Este proceso utiliza un arco eléctrico generado entre un electrodo continuo de metal (el hilo de soldadura) y la pieza a unir. Un gas protector, inerte o activo, envuelve el arco y el baño de fusión, protegiendo la soldadura de la contaminación atmosférica.



Figura 1.3: Equipo soldadura MIG-MAG

Sistema de Alimentación de Alambre

El Sistema de Alimentación de Alambre en la soldadura MIG/MAG tiene la función de suministrar de manera constante el alambre de soldadura hacia el arco eléctrico, donde se funde y une las piezas metálicas. Este sistema consta de una bobina de alambre, un motor de alimentación que impulsa el alambre a través de un tubo flexible, y rodillos de alimentación que controlan la tensión y dirección del alambre.



Figura 1.4: Alimentación de Alambre

Software de Programación

La programación TP (Teach Pendant) es el método principal para programar robots industriales FANUC mediante el uso del dispositivo de enseñanza manual. Esta forma de programación permite crear, editar y ejecutar rutinas directamente desde el robot, utilizando una interfaz basada en menús y bloques de instrucciones estructurados.

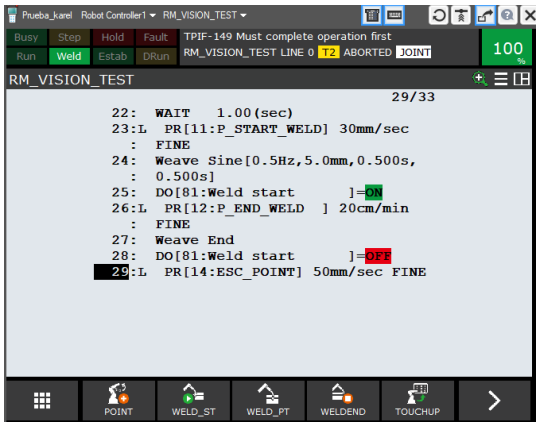


Figura 1.5: Programación Teach Pendant

Gas protector

El gas protector en la soldadura MIG/MAG se utiliza para evitar la contaminación del arco eléctrico y el baño de soldadura mediante la creación de una atmósfera controlada. Dependiendo del tipo de gas, puede ser inert (como el argón o helio) para materiales como el aluminio, o activo (como el dióxido de carbono) para aceros al carbono. Este gas protege el material de soldadura de la oxidación, mejora la estabilidad del arco y asegura una unión de alta calidad, previniendo la presencia de impurezas que puedan afectar la calidad de la soldadura.



Figura 1.6: Gas protector

Implementos de Seguridad

Para ingresar a una celda de trabajo con un brazo robótico en funcionamiento, es imprescindible el uso de **equipos de protección personal (EPP)** adecuados que garanticen la seguridad del operario frente a los riesgos inherentes a la interacción con maquinaria automatizada y los procesos de soldadura. Estos elementos de protección son fundamentales tanto para prevenir lesiones directas como para mitigar los peligros asociados con la radiación, el calor, los riesgos mecánicos y las proyecciones de partículas.

El uso de **zapatos de seguridad** con punta de acero es esencial para proteger los pies de posibles impactos por caída de piezas o herramientas, así como para proporcionar un aislamiento adecuado ante riesgos eléctricos. El **casco de seguridad** es indispensable para proteger la cabeza contra golpes o caídas de objetos en un entorno de trabajo con estructuras elevadas o en movimiento. **Guantes de soldador**, fabricados con materiales resistentes al calor, protegen las manos de quemaduras por contacto o exposición a chispas metálicas generadas durante el proceso de soldadura.

En cuanto a la **protección auditiva**, es recomendable el uso de protectores auditivos debido al elevado nivel de ruido generado por el funcionamiento del brazo robótico y los equipos asociados. Asimismo, se debe portar **ropa ignífuga de trabajo**, que cubra completamente el cuerpo del operario, minimizando el riesgo de quemaduras por radiación o contacto con metales calientes. Finalmente, en el caso de que sea necesario activar el arco de soldadura, el operario debe utilizar una **máscara de soldar** que permita proteger los ojos y la cara de la radiación intensa, además de evitar daños por la exposición al resplandor y las chispas.

Es fundamental, además, que antes de acceder a la celda de trabajo, se verifiquen los protocolos de seguridad, como el estado del robot y sus sistemas de parada de emergencia, asegurando que el entorno esté libre de riesgos innecesarios. La adecuada formación y el cumplimiento de estas medidas son esenciales para garantizar un ambiente de trabajo seguro y eficiente.

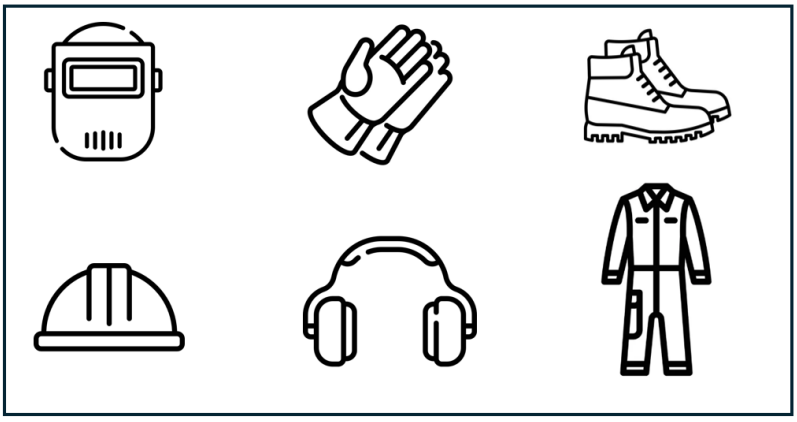


Figura 1.7: Implementos de seguridad

I. Objetivo General

Desarrollar e implementar un sistema de visión artificial integrado a un brazo robótico FANUC con el fin de automatizar el proceso de soldadura, permitiendo al robot identificar piezas de forma autónoma, planificar trayectorias de soldadura precisas y ejecutar cordones de soldadura de alta calidad.

II. Objetivos Específicos

- Desarrollar un soporte para la cámara, su función será acoplar (brazo robótico) y proteger (radiación y temperatura de arco eléctrico).
- Configurar y calibrar una cámara para capturar imágenes de las piezas a soldar.
- Desarrollar algoritmos para generar trayectorias de soldadura adaptables a las variaciones en la posición y orientación de las piezas detectadas por el sistema de visión.
- Programar brazo robótico integrando los datos obtenidos del análisis de imagen.

III. Alcances del Proyecto

- **Instalación y Protección de la cámara:** Para asegurar el correcto funcionamiento del sistema de visión artificial en el entorno de soldadura, se realizó una instalación estratégica de la cámara, considerando las condiciones adversas del ambiente, como altas temperaturas, salpicaduras de metal fundido, etc.

  Con el fin de proteger el lente y los componentes electrónicos, se diseñó una cubierta protectora fabricada con materiales resistentes al calor y a las proyecciones propias del proceso de soldadura. Esta carcasa actúa como barrera física, minimizando los riesgos de daño directo a la cámara y permitiendo su operación segura y continua. Además, su diseño incorpora un acople diseñado específicamente para este modelo de robot de la marca Fanuc.

- **Implementación del sistema de comunicación:** La cámara se conecta al ordenador mediante un cable coaxial de alta velocidad, lo que permite una transmisión rápida y estable de las imágenes. El ordenador se comunica con el controlador R-30iB Plus del brazo robótico a través de una conexión Ethernet utilizando el protocolo TCP/IP. Este controlador centraliza la comunicación entre los dispositivos, permitiendo una sincronización eficiente entre la captura de imágenes, el procesamiento de datos y la ejecución de movimientos del robot durante el proceso de soldadura.

- **Análisis de imágenes y generación de trayectoria:** Se implementó un sistema de procesamiento de imágenes mediante herramientas de visión por computador, utilizando la biblioteca OpenCV. A partir de las imágenes capturadas, se identifican características relevantes de las piezas a soldar y se generan trayectorias que permiten guiar al brazo robótico de forma automática. Este proceso posibilita una adaptación dinámica a la posición y geometría de las piezas, optimizando la precisión y eficiencia en la ejecución de la soldadura.

IV. Restricciones

- **Iluminación:** La iluminación juega un rol fundamental en la visión artificial, ya que afecta directamente la calidad de las imágenes capturadas. Para asegurar una detección precisa y estable, especialmente en entornos complejos como el de soldadura, es necesario mantener condiciones de iluminación constantes y controladas. Esto permite mejorar la fiabilidad del análisis de imagen y la generación de trayectorias para el brazo robótico.

- **Evaluación del software nativo Fanuc:**

  Se identificaron posibles limitaciones en el sistema de visión del software nativo de FANUC, particularmente en cuanto a la capacidad para realizar análisis de imágenes avanzados. Estas restricciones pueden afectar la detección precisa de formas o trayectorias en aplicaciones más complejas, lo que motivó la integración de herramientas externas como OpenCV para ampliar las capacidades del sistema de visión artificial.

- **Optimización de recursos:**

  Si bien los equipos y materiales necesarios para el desarrollo del proyecto se encuentran disponibles en la universidad, es fundamental realizar una gestión eficiente de estos recursos. Un uso adecuado y planificado permite evitar cuellos de botella o limitaciones que puedan afectar el avance y cumplimiento de los objetivos del proyecto.

Capítulo 2: Comunicación y Redes

Este capítulo introduce los principales sistemas que conforman una celda de soldadura robotizada, enfocándose en la comunicación entre el brazo robótico, el controlador y los periféricos. Se abordan los principios de operación del entorno de programación TP de FANUC, los equipos de soldadura MIG/MAG, y los sistemas de visión artificial utilizados para la supervisión y corrección del proceso en tiempo real, destacando la importancia de la integración y sincronización entre todos los componentes.

2.1 Diagrama Comunicación y Redes

El diagrama muestra la estructura general de un sistema de soldadura automatizado que utiliza visión artificial para guiar un brazo robótico soldador. El controlador central gestiona la comunicación entre el ordenador, encargado de la extracción y procesamiento de imágenes, el sistema de visión, el robot y la soldadora MIG/MAG. Además, se incluye una interfaz de programación y control manual a través del Teach Pendant. Un componente clave es el módulo RI 3000, responsable de traducir el protocolo CAN-BUS a Ethernet/IP, permitiendo así la comunicación entre la soldadora y el sistema robótico. Este conjunto integrado permite realizar cordones de soldadura de manera precisa, eficiente y automatizada.

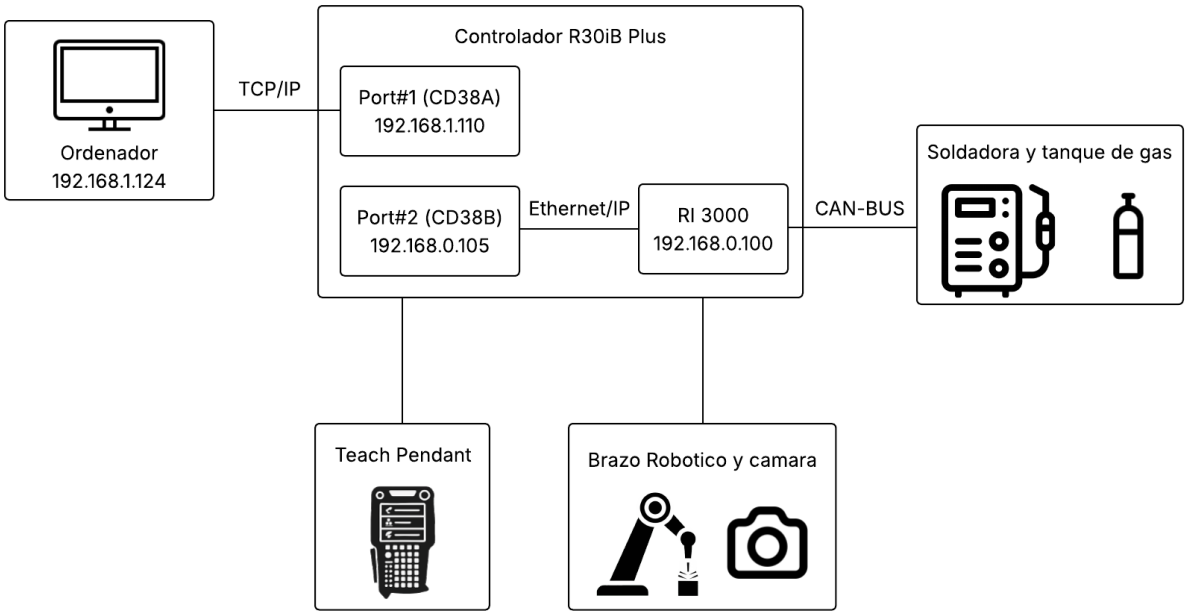


Figura 2.1: Diagrama General

2.2 Brazo Robótico y Controlador

Brazo Robótico

El brazo robótico Fanuc Arc Mate 100iD se encuentra instalado dentro de la celda de trabajo y está vinculado al controlador R-30iB Plus, el cual coordina y supervisa todas sus operaciones. Este modelo está especialmente diseñado para aplicaciones de soldadura por arco, destacándose por su diseño compacto, su capacidad de carga útil de hasta 12 kg y un alcance máximo de aproximadamente 1441 mm, lo que le permite trabajar con piezas de distintos tamaños en espacios reducidos.

Su principal función es ejecutar trayectorias de soldadura previamente programadas, posicionando la antorcha con gran precisión sobre las piezas metálicas a unir. Gracias a su alta repetibilidad (±0.02 mm), garantiza la consistencia del cordón de soldadura, incluso en procesos automatizados de alta demanda. Su estructura totalmente cerrada con clasificación IP67 en el brazo y muñeca lo hace ideal para entornos industriales exigentes, protegiendo los componentes internos del polvo, salpicaduras y humo de soldadura.

La sincronización con el controlador permite adaptar los movimientos del robot en tiempo real, integrando información proveniente de sistemas de visión o sensores, lo que asegura una operación precisa, segura y eficiente durante el proceso de soldadura. Además, su compatibilidad con tecnologías como ARC Tool y iRVision de Fanuc facilita su implementación en celdas automatizadas inteligentes.



Figura 2.2: Brazo Robótico Arc Mate 100iD

Controlador R-30iB y Teach Pendant (TP)

El controlador FANUC R-30iB Plus, instalado en un gabinete independiente, es el núcleo de procesamiento del sistema robótico. Diseñado para aplicaciones industriales exigentes, cuenta con una arquitectura modular y un procesador de alto rendimiento que mejora los tiempos de ciclo y la respuesta del sistema. Su capacidad de integración es amplia, soportando protocolos como Ethernet/IP, ProfiNet, DeviceNet y CC-Link, siendo Ethernet/IP el utilizado en nuestro caso para garantizar una comunicación rápida y estable con el brazo robótico y otros dispositivos como la cámara de visión artificial.

Este controlador ejecuta programas escritos en TPP (Teach Pendant Programming) y también admite KAREL, un lenguaje de alto nivel que permite desarrollar rutinas complejas, gestionar archivos y ejecutar múltiples tareas simultáneamente. Además, incorpora funciones de seguridad como monitoreo de colisiones, zonas de restricción y paros de emergencia.

La interacción con el sistema se realiza mediante el iPendant Touch, un dispositivo portátil con pantalla táctil a color e interfaz gráfica optimizada. Permite programar, ajustar parámetros del proceso de soldadura, visualizar variables en tiempo real y diagnosticar fallos. También habilita el movimiento manual del robot en modo jog, lo cual es fundamental para la calibración, pruebas y mantenimiento del sistema.



Figura 2.3: Controlador R-30iB Plus



Figura 2.4: Teach Pendant

RI-3000

El módulo de interfaz Selco RI 3000 permite la integración de fuentes de soldadura Selco con sistemas de automatización industrial mediante protocolos de comunicación estandarizados. Es compatible con PROFINET-IO, DeviceNet, CAN-BUS y Ethernet TCP/IP, lo que facilita su incorporación en diversas arquitecturas de control. Este dispositivo permite la transmisión bidireccional de parámetros de soldadura y alarmas.

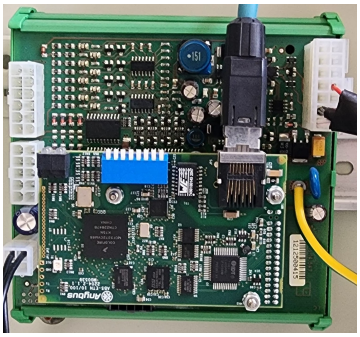


Figura 2.5: RI-3000

2.3 Equipo de Soldadura

Equipo Selco Genesis 3200 MSE

La Selco Genesis 3200 MSE es una soldadora avanzada para procesos MIG/MAG, equipada con una fuente de energía que genera el arco eléctrico, un alimentador de alambre continuo y un sistema de gas protector que evita la oxidación del cordón de soldadura. Sus características incluyen ajustes de intensidad de corriente, voltaje regulable, velocidad de alimentación del alambre y flujo de gas protector, lo que permite adaptarse a diferentes materiales y condiciones, asegurando alta precisión y calidad en aplicaciones industriales.



Figura 2.6: Selco Genesis 3200 MSE

Selco WFR 1000 Classic

Este alimentador de alambre está orientado a aplicaciones industriales que exigen alimentación continua. Incorpora un sistema de arrastre por rodillos ajustables mediante tornillos de rosca, lo que permite regular la presión según el diámetro del alambre, asegurando un avance constante y uniforme.



Figura 2.7: Alimentador Selco WFR 1000 Classic

Rollo de Soldadura

El alambre de soldadura ER70S-6 se caracteriza por su alto contenido de silicio, mejora la desoxidación del baño de fusión y permite trabajar sobre superficies con óxido o impurezas. Es ampliamente utilizado en soldadura MIG/MAG de aceros al carbono, ofreciendo buena resistencia mecánica, excelente acabado del cordón y un rendimiento confiable en aplicaciones estructurales e industriales.



Figura 2.8: Rollo de 15Kg soldadura ER70S-6

Gas protector

Se utiliza gas argón como protección, el cual evita la contaminación del cordón al aislar el arco eléctrico y el metal fundido de la atmósfera. Al ser un gas inerte, contribuye a una mayor estabilidad del arco y una buena calidad de soldadura. Su flujo debe regularse adecuadamente para asegurar una cobertura uniforme durante el proceso.Aunque no se cuenta con un regulador manual específico, se deduce que la máquina de soldadura Selco Genesis 3200 MSE controla automáticamente la apertura del flujo de gas al iniciar el proceso de soldadura, coordinándose con la activación del arco eléctrico. Esta función es probablemente gestionada internamente por la máquina, sincronizada con las señales enviadas desde el controlador del sistema.



Figura 2.9: Cilindro de gas (Argón)

Antorcha de soldadura

La antorcha de soldadura es el componente que genera el arco eléctrico y alimenta el alambre de aporte en procesos como MIG/MAG. Suministra gas protector para evitar la contaminación del cordón y consta de un cuerpo resistente al calor, una boquilla para el gas y un sistema de alimentación del alambre. La antorcha está montada en el brazo robótico, que está conectado al controlador R-30iB Plus. A través de este controlador, se gestionan diversos parámetros, como la activación del arco eléctrico, la velocidad del alambre, la velocidad de soldadura y la forma del cordón. Estas funciones son operadas desde el teach pendant, permitiendo ajustar y ejecutar el proceso de soldadura de manera precisa y sincronizada con el resto de los dispositivos de la celda de trabajo.



Figura 2.10: Antorcha de soldadura

2.4 Procesamiento de imágenes

Cámara SC130EF2 B/W

La cámara FANUC SC130EF2 B/W es un dispositivo de visión monocromática diseñado para aplicaciones industriales de alta precisión, especialmente en entornos de soldadura automatizada. Equipada con un sensor CMOS de 1/1.8 pulgadas y escaneo progresivo, esta cámara ofrece una resolución de 1280 x 1024 píxeles, lo que permite capturar imágenes detalladas y nítidas incluso en condiciones de iluminación variables.

El diseño óptico incluye una lente KOWA de 12 mm con montura C, que proporciona un control manual del enfoque y la apertura (rango de F1.8 a F12), permitiendo ajustes precisos según las necesidades específicas del proceso. La cámara también cuenta con ajustes físicos mediante ejes mecánicos que permiten regular manualmente el brillo y el enfoque, adaptándola a las condiciones específicas del entorno de soldadura. Estos ejes controlan la cantidad de luz que entra al sensor y la nitidez de la imagen, garantizando capturas precisas incluso con las variaciones de iluminación causadas por el arco eléctrico.

En términos de conectividad, la SC130EF2 B/W utiliza una interfaz de transmisión serial original de KOWA, que permite la transmisión de datos, señales de control y alimentación a través de un único cable coaxial de 75 ohmios (tipo CANARE L-SCFB), con una longitud máxima de hasta 100 metros. Esta configuración simplifica la instalación y reduce los costos del sistema al eliminar la necesidad de unidades de control adicionales.

Desde las configuraciones del controlador FANUC R-30iB Plus, es posible ajustar parámetros avanzados de la cámara, como contraste, exposición, ganancia y resolución, permitiendo una optimización detallada para aplicaciones de procesamiento de imágenes en tiempo real. La cámara se integra perfectamente con el sistema iRVision 2D, facilitando tareas como la detección de piezas, el seguimiento de trayectorias y la inspección de calidad en procesos de soldadura robotizada.



Figura 2.11: Cámara SC130EF2 B/W

Equipo de procesamiento de imágenes

El equipo de procesamiento de imágenes está compuesto por un computador industrial dedicado, cuya función es interpretar en tiempo real las imágenes capturadas por la cámara FANUC SC130EF2 B/W. Este equipo, integrado al entorno de trabajo del robot, está configurado para ejecutar tareas de análisis visual mediante la biblioteca OpenCV, aprovechando los recursos del hardware y software disponibles. Utiliza algoritmos como el filtro de Canny para detectar bordes con alta precisión, permitiendo identificar contornos de piezas, evaluar la alineación y detectar desviaciones en el proceso de soldadura. Estas funciones son especialmente críticas en entornos con alta variabilidad lumínica, por lo que el equipo ajusta automáticamente parámetros como brillo, exposición y contraste para mantener la calidad visual de los datos entrantes.

Dentro del equipo se emplean diversas herramientas de software que optimizan su funcionamiento y facilitan la comunicación con el controlador del robot. Se utiliza Python como lenguaje principal para el desarrollo de scripts de procesamiento, OpenCV para la visión artificial, FileZilla para la transferencia de archivos entre el computador y el controlador, y RoboGuide para realizar simulaciones y pruebas de funcionamiento antes de implementarlas en la celda física. Gracias a esta combinación de hardware y software, el equipo es capaz de generar y enviar información precisa al controlador R-30iB Plus, que la utiliza para ajustar la trayectoria del brazo robótico en tiempo real, garantizando una soldadura automatizada, adaptable y de alta calidad.

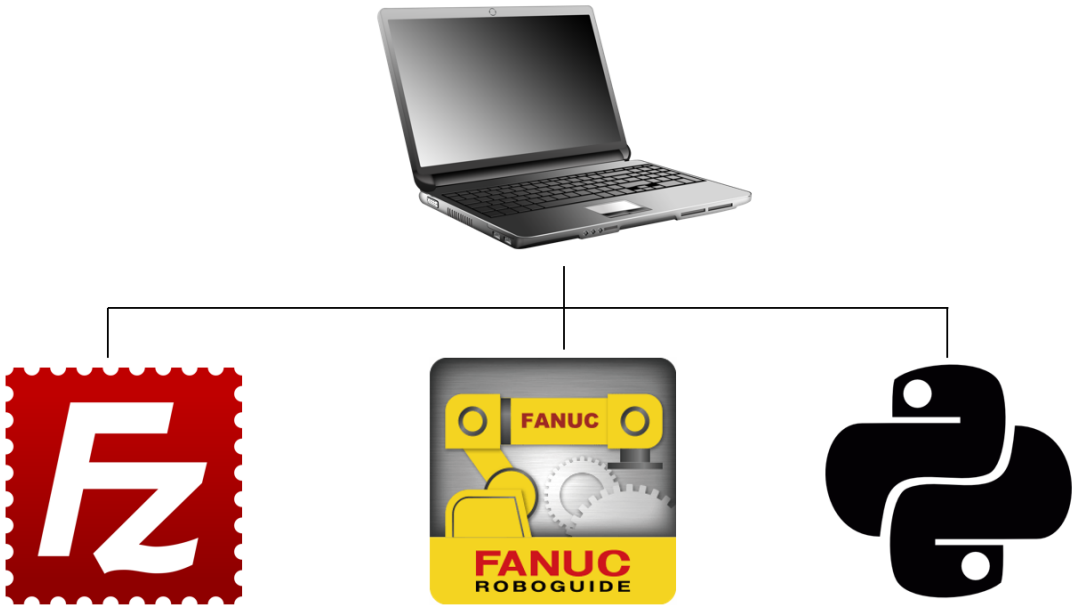


Figura 2.12: Equipo Procesamiento de imágenes

Capítulo 3: Integración Protocolo FTP

3.1 Diagrama Red FTP

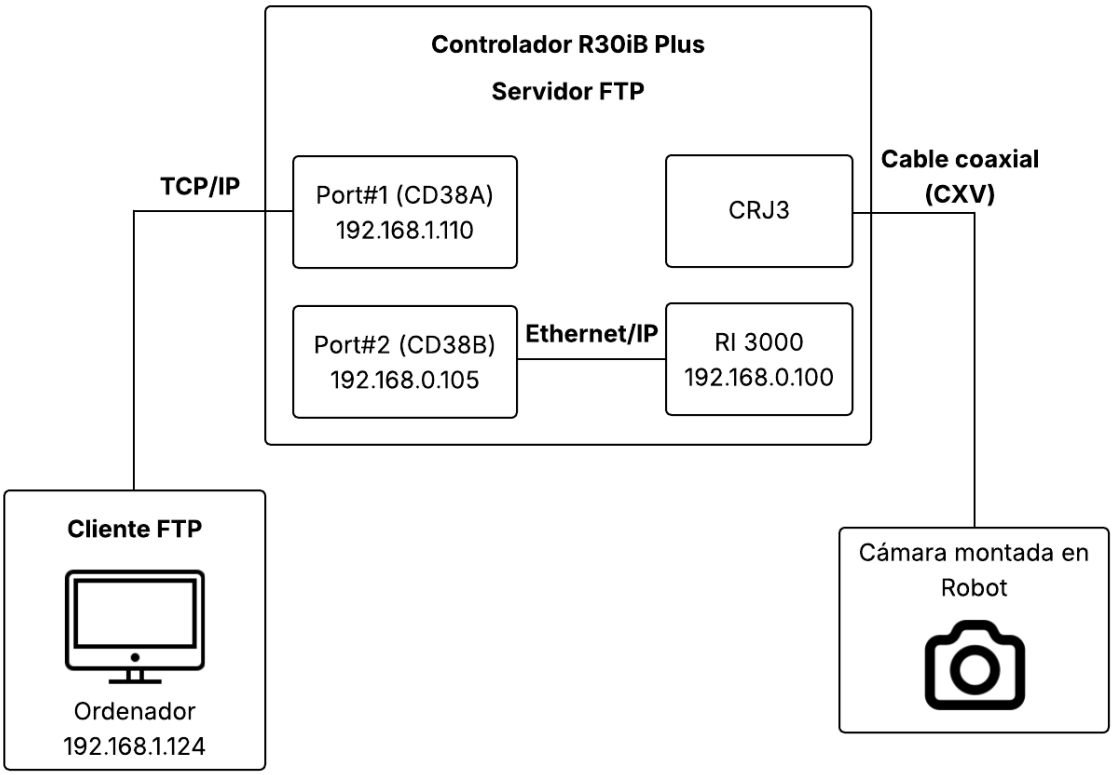


Figura 3.1: Diagrama Red FTP

3.2 Planteamiento de Conexiones y Montaje

Conexión al Controlador R-30iB Plus

El controlador FANUC R-30iB Plus permite configurar hasta dos puertos Ethernet para comunicación mediante EtherNet/IP, siempre que se ubiquen en subredes distintas para evitar conflictos. El puerto CD38B está especialmente optimizado para protocolos de E/S industriales como EtherNet/IP, mientras que el puerto CD38A se recomienda para conexión con la red del edificio, facilitando el acceso al robot mediante servicios como HTTP o FTP (Revisar **Anexo [8]**). Esta configuración garantiza una separación lógica entre las redes de automatización y las de gestión.

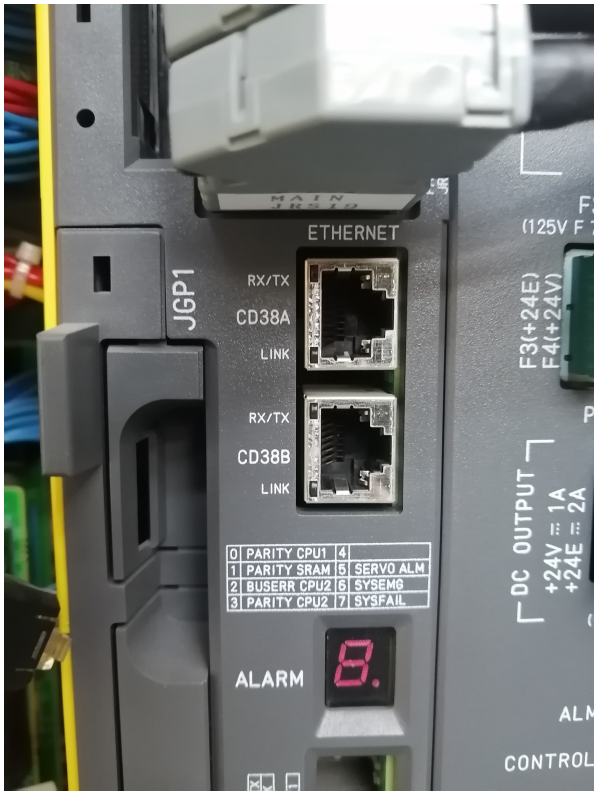


Figura 3.2: Puertos Controlador CD38-A y CD38-B

Montaje de la Cámara a Brazo Robótico

La cámara SC130EF2 B/W fue instalada sobre un soporte especialmente diseñado para su integración en la celda de trabajo. Este soporte fue modelado utilizando el software de diseño Autodesk Inventor, lo que permitió desarrollar una geometría precisa, optimizada para adaptarse a la estructura del brazo robótico FANUC, garantizando un ajuste exacto y una correcta distribución de cargas durante el funcionamiento.

Para su fabricación se utilizó tecnología de impresión 3D en una impresora de sistema cerrado, empleando filamento ASA (Acrilonitrilo Estireno Acrilato), un termoplástico técnico elegido por su excelente resistencia a la radiación UV, su estabilidad mecánica, y su tolerancia a altas temperaturas, lo que lo hace especialmente apto para entornos de soldadura. El uso de una impresora cerrada fue fundamental, dado que el ASA presenta una alta susceptibilidad a la deformación por cambios térmicos durante la impresión, lo que puede comprometer la precisión dimensional de la pieza.

Una vez completada la fabricación y verificados los ajustes del diseño, la cámara fue acoplada al soporte mediante elementos de fijación mecánicos, asegurando su correcta orientación y asegurando su correcta orientación respecto al área de trabajo y evitando movimientos o vibraciones que pudieran afectar la calidad de las imágenes. El conjunto fue montado en el extremo del brazo robótico, cuidando el alineamiento con la zona de soldadura para garantizar una captura de imágenes efectiva.

La cámara se conectó al controlador FANUC R-30iB Plus a través de un cable tipo CXV, lo que permite una transmisión de datos eficiente, con alta velocidad y baja latencia. Esta conexión asegura una comunicación estable y continua entre ambos dispositivos, condición necesaria para el análisis en tiempo real y la integración efectiva del sistema de visión artificial dentro del proceso automatizado.

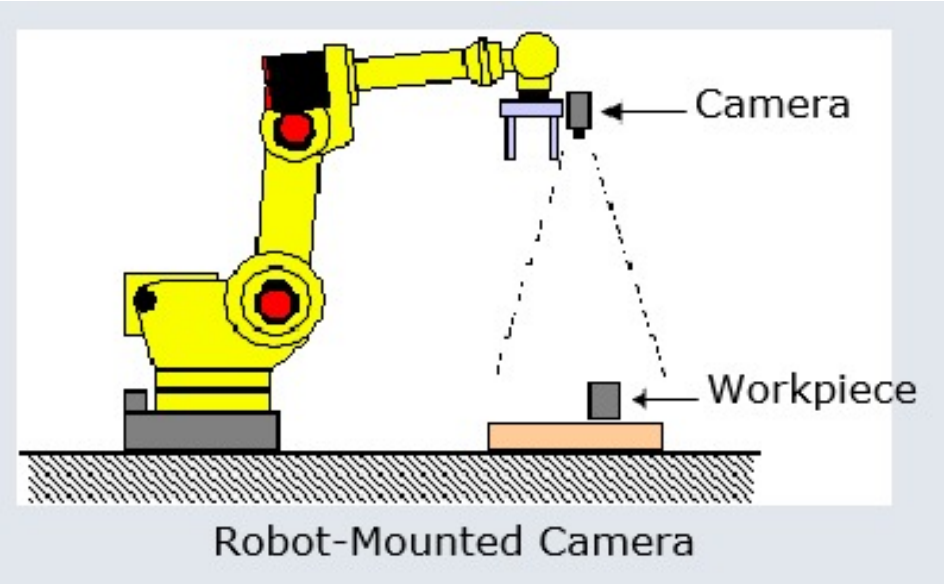


Figura 3.3: Cámara Montada en Brazo Robótico

3.3 Integración de los Componentes

Realizar Conexiones en Controlador

Para la configuración de red utilizada en este proyecto, se establecieron las conexiones físicas necesarias para garantizar una comunicación eficiente entre los distintos componentes del sistema. En primer lugar, se conectó un cable Ethernet con conector RJ45 desde el equipo de procesamiento de imágenes hasta el Puerto 1 (CD38-A) del controlador FANUC R-30iB Plus, permitiendo el intercambio bidireccional de datos necesarios para la ejecución de tareas de visión artificial en tiempo real.

Posteriormente, se integró el módulo de interfaz RI-3000, responsable de la comunicación entre el controlador y la fuente de soldadura MIG/MAG. Este módulo fue conectado al Puerto 2 (CD38-B) del controlador mediante un cable Ethernet industrial blindado, asegurando una transmisión de datos robusta y resistente a las interferencias electromagnéticas propias del entorno de soldadura.

Por último, la cámara SC130EF2 B/W fue conectada directamente al controlador mediante el protocolo CXV (Camera Extension Video), el cual permite la transmisión de señales de video y control con alta fidelidad y baja latencia. Esta conexión es fundamental para que el controlador pueda recibir imágenes en tiempo real y ejecutar algoritmos de análisis a través del sistema de procesamiento externo

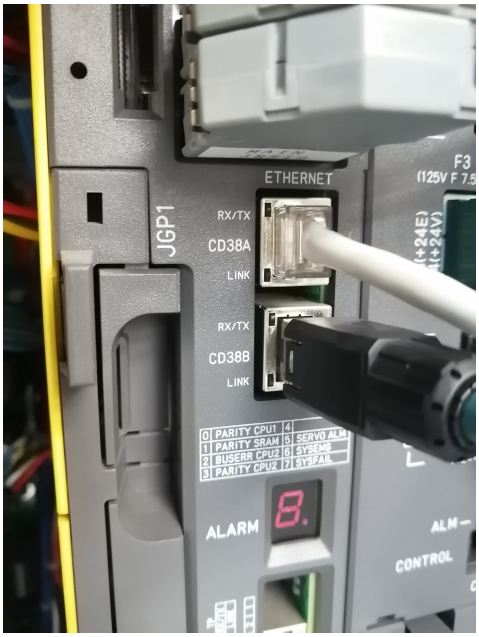


Figura 3.4: Conexiones CD38-A y CD-38-B

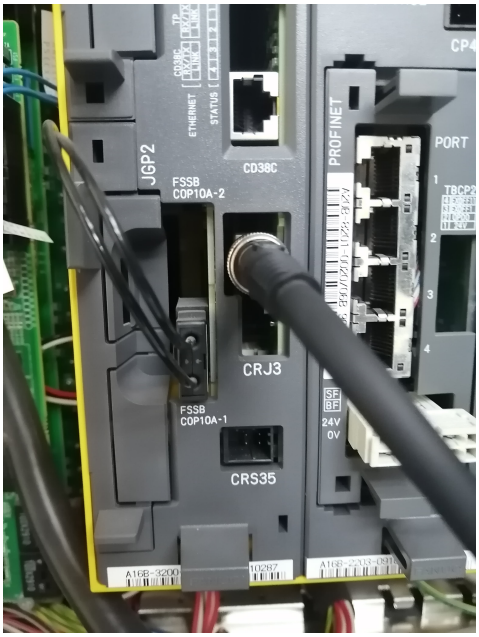


Figura 3.5: Conexión Cámara

Asignación Direcciones IP

Se llevó a cabo la asignación de direcciones IP estáticas para cada puerto, a través de la interfaz del Teach Pendant, utilizando la siguiente ruta de configuración:

- **Menú → Setup → Host Comm → TCP/IP**
- **Puerto 1 (CD38-A): 192.168.1.110**
- **Puerto 2 (CD38-B): 192.168.0.105**

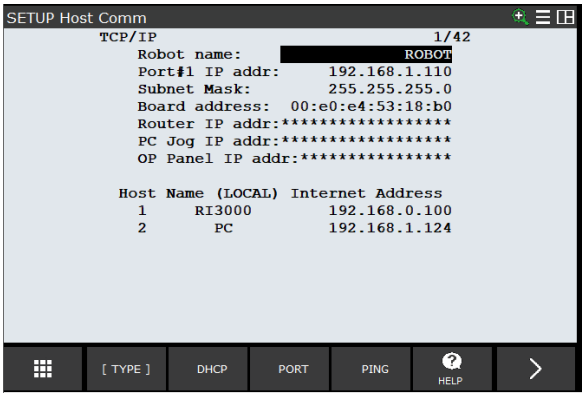


Figura 3.6: Dirección IP Puerto 1 (CD38-A)

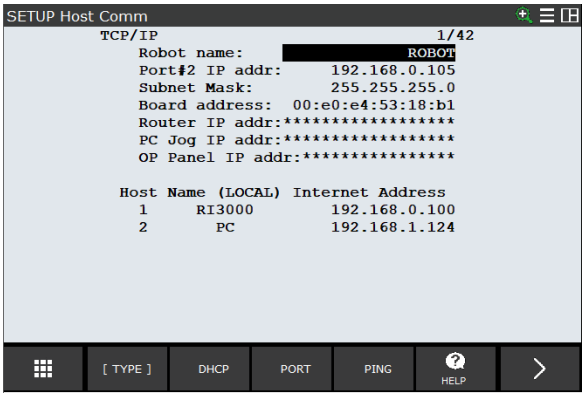


Figura 3.7: Dirección IP Puerto 2 (CD38-B)

Ahora en el apartado de **Host name (Local) Internet Address** Asignamos las direcciones IP a utilizar:

- **RI-3000 : 192.168.0.100**
- **Ordenador : 192.168.1.124**

Posteriormente, se configuró la dirección IP del ordenador, asignándole una dirección estática correspondiente a **"192.168.1.124"** para mantener una comunicación estable con el sistema.

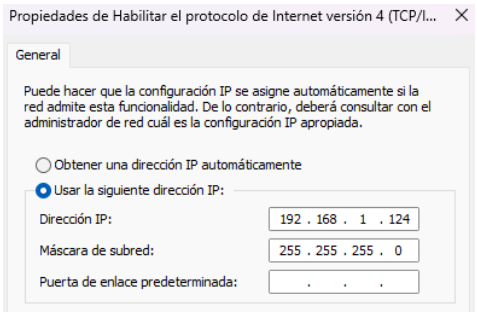


Figura 3.8: Dirección IP Estática PC

Finalmente, se realizaron pruebas de conectividad mediante el comando **Ping**, con el objetivo de verificar la correcta comunicación entre los dispositivos.

Configuración Servidor FTP

Se procedió a la activación del servidor FTP en el Teach Pendant, siguiendo la siguiente ruta de acceso:

- **Menú → Setup → Host Comm → SHOW → Servers**

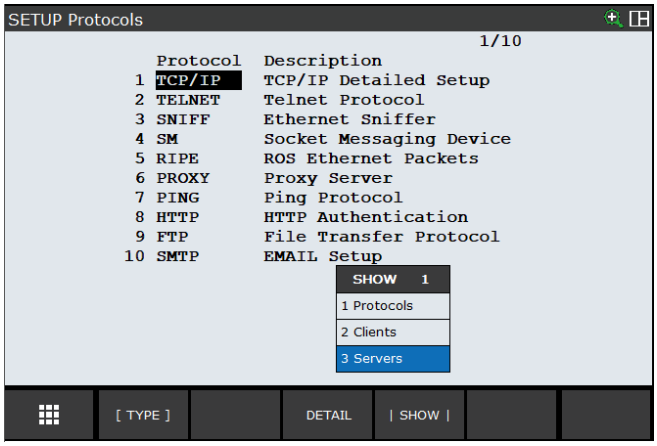


Figura 3.9: Seleccionar Servidores

Dentro del menú de servidores, se seleccionó el servidor S2 para su configuración y puesta en operación. El procedimiento realizado fue el siguiente:

- **Seleccionar S2 → Detail.**

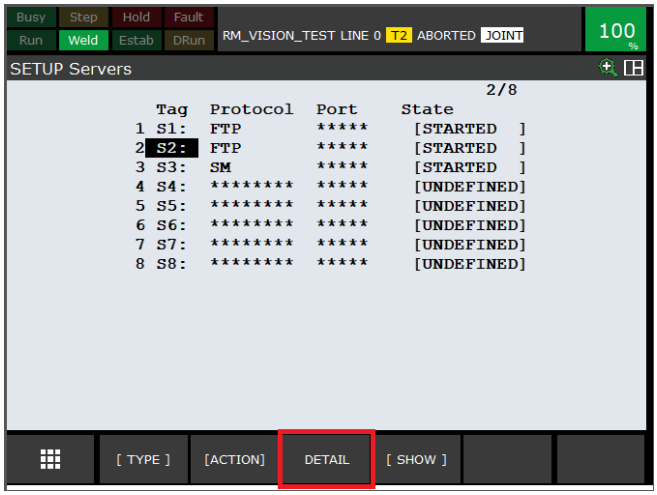


Figura 3.10: Servidores

Verificar que el campo Current State se encuentre en **UNDEFINED**, condición necesaria para permitir la modificación de los parámetros.

- **Establecer el Protocolo en FTP.**
- **Ingresar en Action y seleccionar la opción Start.**
- **Una vez finalizadas las modificaciones, se debe actualizar el campo Current State, asignándole manualmente el valor "STARTED".**

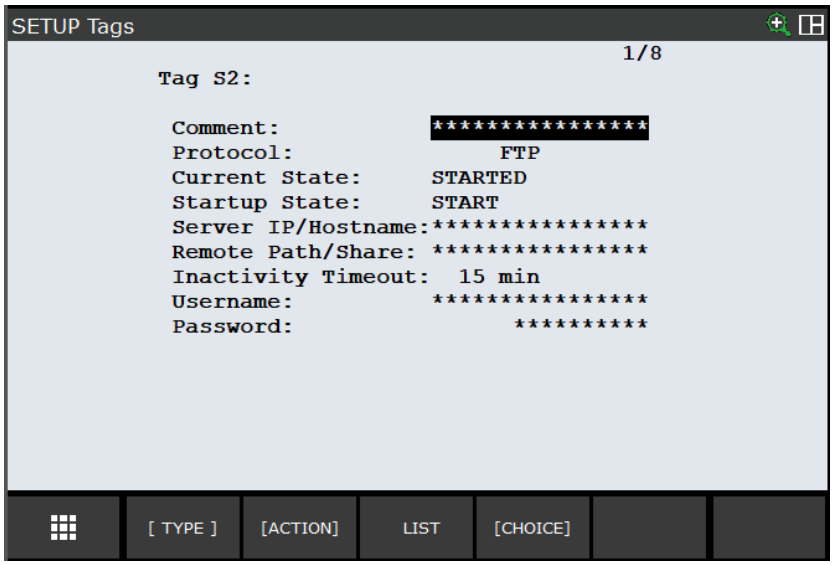


Figura 3.11: Configuraciones Servidor FTP

De este modo, se completó la activación del servidor. La correcta operación del servidor FTP se confirma al verificar que el **State** permanezca en **"STARTED"**, indicando que se encuentra activo y disponible para la transferencia de archivos.

3.4 Desarrollo y Montaje de Protección Cámara de Visión

Desarrollo de Protección y Soporte

El soporte de la cámara fue diseñado utilizando el software de modelado CAD Autodesk Inventor, tomando como base las dimensiones exactas del dispositivo de visión artificial, así como las restricciones geométricas del entorno de trabajo (Revisar planos en **Anexo [9]**). El objetivo fue desarrollar una estructura de montaje segura que, además de garantizar una fijación estable, ofreciera protección física y térmica frente a las condiciones exigentes del proceso de soldadura. Una vez finalizado el diseño digital, la pieza fue fabricada mediante tecnología de impresión 3D FDM (Fused Deposition Modeling), utilizando filamento ASA (Acrilonitrilo Estireno Acrilato), un material termoplástico técnico con propiedades avanzadas de resistencia ambiental y estabilidad dimensional.

La elección de ASA como material de impresión responde a su capacidad para mantener integridad estructural bajo exposición a altas temperaturas, así como a su resistencia frente a proyecciones de soldadura, partículas metálicas incandescentes y atmósferas corrosivas propias del entorno MIG/MAG. A diferencia de otros polímeros de uso común, el ASA no se deforma ni se degrada ante radiación UV o variaciones térmicas, lo que lo hace ideal para entornos industriales exigentes. Estas características aseguran una protección adecuada del sistema de visión, preservando su funcionalidad a lo largo del tiempo y evitando interferencias mecánicas que comprometan el desempeño del robot.

Para una correcta impresión con filamento ASA, se deben considerar parámetros técnicos como una temperatura de extrusión entre 240 °C y 260 °C, y una cama calefactada entre 90 °C y 110 °C, a fin de garantizar buena adherencia y evitar deformaciones. Se recomienda utilizar impresoras con recinto cerrado debido a la sensibilidad del material a cambios térmicos. Además, es importante emplear velocidades moderadas, una retracción adecuada y adhesivos en la cama para mejorar la estabilidad. El cumplimiento de estas condiciones permite obtener piezas resistentes y dimensionalmente estables, aptas para entornos industriales exigentes como la soldadura robotizada una de las impresoras que cumplen estos requisitos es la que se muestra en la **Figura 3.12**.



Figura 3.12: Impresora Anycubic Kobra S1 Combo

El soporte incorpora un sistema de fijación específico para el modelo FANUC Arc Mate 100iD, adaptado directamente desde la geometría del brazo robótico. Esta personalización garantiza una alineación precisa con el campo visual requerido, así como una instalación mecánicamente robusta, evitando desajustes durante movimientos dinámicos. En conjunto, el soporte constituye un componente clave para la confiabilidad operativa del sistema automatizado.

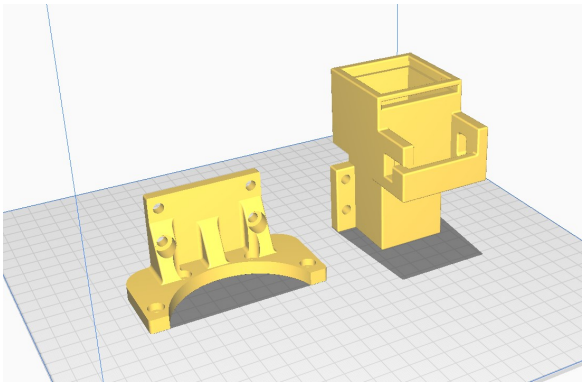


Figura 3.13: Modelo 3D Cubierta Protectora

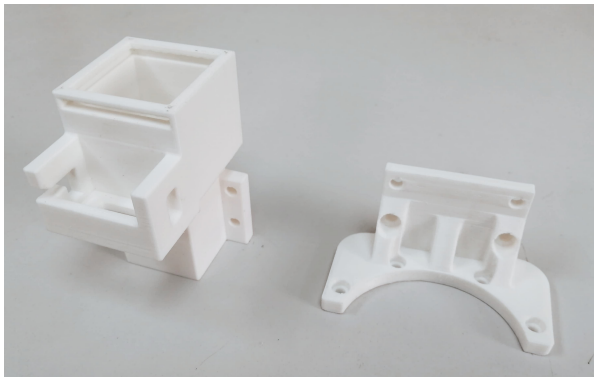


Figura 3.14: Prototipo impreso en 3D

Montaje Cámara y Soporte

Una vez fabricada la cubierta protectora mediante impresión 3D con filamento ASA, se procedió a fijar la cámara de visión artificial utilizando tornillos, asegurando su correcta sujeción. Posteriormente, se acopló la cubierta al brazo robótico a través de los puntos de anclaje diseñados, garantizando una integración estable y adecuada para su operación en el ambiente de soldadura. El ensamblaje final se muestra en la siguiente imagen.



Figura 3.15: Soporte y cubierta montada en brazo robótico

3.5 Configuración y Calibración de Cámara

Configuración Cámara

La cámara de visión artificial modelo SC130EF2 B/W, cuenta con dos perillas que permiten ajustar manualmente el brillo y el enfoque, facilitando la optimización de la imagen de acuerdo con las necesidades específicas del proceso.

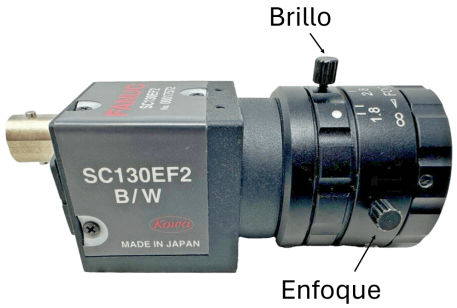


Figura 3.16: Cámara de visión Artificial

Posteriormente, dentro del Teach Pendant, se debe habilitar la captura y almacenamiento de imágenes mediante el siguiente procedimiento:

- **Menú → iRVision → Vision Config.**

En este apartado, se debe activar la opción **Enable Logging** y definir la ruta de almacenamiento para las imágenes capturadas así como se ve en la **Figura 3.17**.

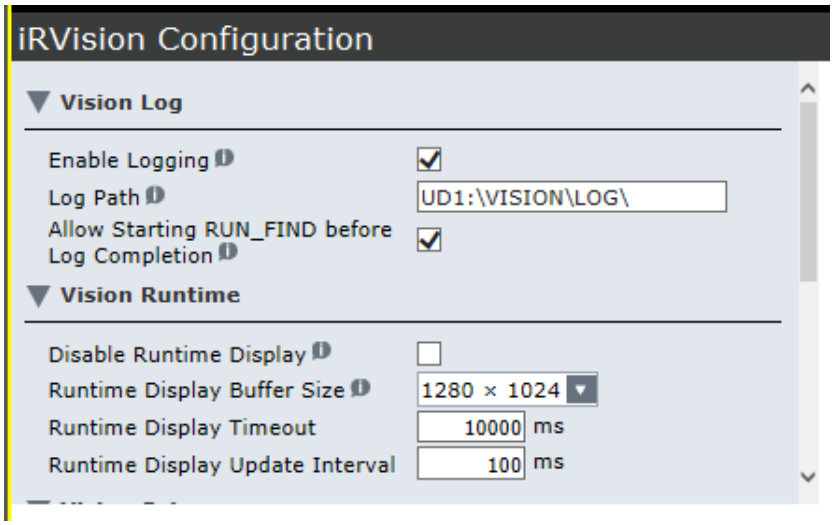


Figura 3.17: Configuración Almacenamiento de imágenes

En la opción Log Path es posible definir hasta tres directorios raíz distintos para el almacenamiento de las imágenes capturadas. Estos directorios son los siguientes:

- **UD1:** Corresponde a la unidad de almacenamiento USB, donde UD1 hace referencia a una memoria USB conectada directamente al controlador .
- **MC:** Hace referencia a una tarjeta de memoria interna (Memory Card) del controlador.
- **PC Share:** Permite el envío directo de imágenes a un ordenador mediante un protocolo de red.
- **Nota:** Para este proyecto, se seleccionó la opción **UD1:**, utilizando una memoria USB conectada directamente al controlador, con el fin de almacenar de forma local las imágenes capturadas.

Posición Robot con Cámara Montada

Una vez montada la cámara, es fundamental establecer una posición fija dentro del entorno de trabajo, la cual será utilizada por el robot de forma recurrente para la captura de imágenes. Además, esta ubicación sirve como referencia para realizar la calibración de la cámara. La posición es determinada por el operador, en función de su criterio técnico y de acuerdo con las condiciones más convenientes del proceso. Asimismo, esta posición servirá como punto de referencia para enlazar el user frame que se definirá posteriormente, permitiendo asociar correctamente las coordenadas obtenidas durante el proceso de visión artificial.



Figura 3.18: Imagen de cámara en posición fija

Una vez finalizado el procedimiento de registro de imágenes en el directorio correspondiente y de definir la posición de la cámara, se debe proceder con la calibración de la cámara. Sin embargo, antes de llevar a cabo dicho proceso, es necesario configurar adecuadamente la zona de trabajo, estableciendo el User Frame que servirá como sistema de referencia común entre el robot y la cámara. Esta configuración es fundamental para asegurar la correcta correlación espacial entre ambos sistemas y garantizar la precisión en las tareas de visión artificial.

Calibración User Frame en Brazo robótico

El User Frame en robots FANUC es un sistema de coordenadas definido por el usuario que permite programar posiciones y trayectorias en función de un entorno de trabajo específico. Su calibración es fundamental para asegurar precisión, especialmente cuando dicho entorno no coincide con el sistema base del robot.

Además de su utilidad en la programación de tareas, el User Frame también será empleado como referencia durante la calibración de la cámara de visión artificial, utilizando un patrón de calibración tipo grid, lo que garantiza la correcta alineación espacial entre el sistema de visión y el robot.

Esta calibración se realiza mediante la enseñanza de puntos físicos, utilizando métodos como el de tres o seis puntos.

- **Nota:** El procedimiento detallado para la calibración del User Frame, se encuentra en el **Anexo [5]**.

Calibración de Cámara en iRVision de FANUC

Con el fin de habilitar la funcionalidad de la cámara en la programación TP de Fanuc, es indispensable realizar el proceso de calibración utilizando la biblioteca iRVision, accesible desde el Teach Pendant. Para realizar este procedimiento se requiere de un **Grid de Calibración/Patrón de Calibración** el cual se muestra en la **Figura 3.19**. El procedimiento para la calibración se encuentra en el **Anexo [1]**.

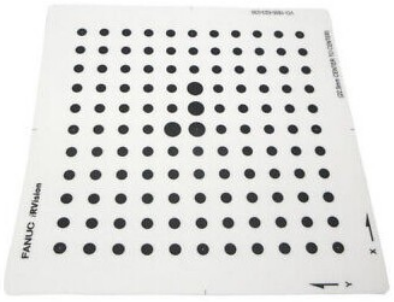


Figura 3.19: Grid de Calibración

3.6 Captura y Transferencia de Imágenes

Código en Brazo Robótico para captura de imágenes

Para comenzar se debe crear un programa en el Teach Pendant siguiendo la siguiente ruta:

- **Select → Create → Asignar nombre a nuestro programa**

Es importante asignar un nombre representativo que permita identificar fácilmente la función del programa dentro del sistema.

Una vez creado el programa, se debe ingresar la instrucción correspondiente para capturar una imagen mediante la cámara. Para ello, dentro del programa se debe seguir la siguiente ruta en el Teach Pendant:

- **Inst → Vision**

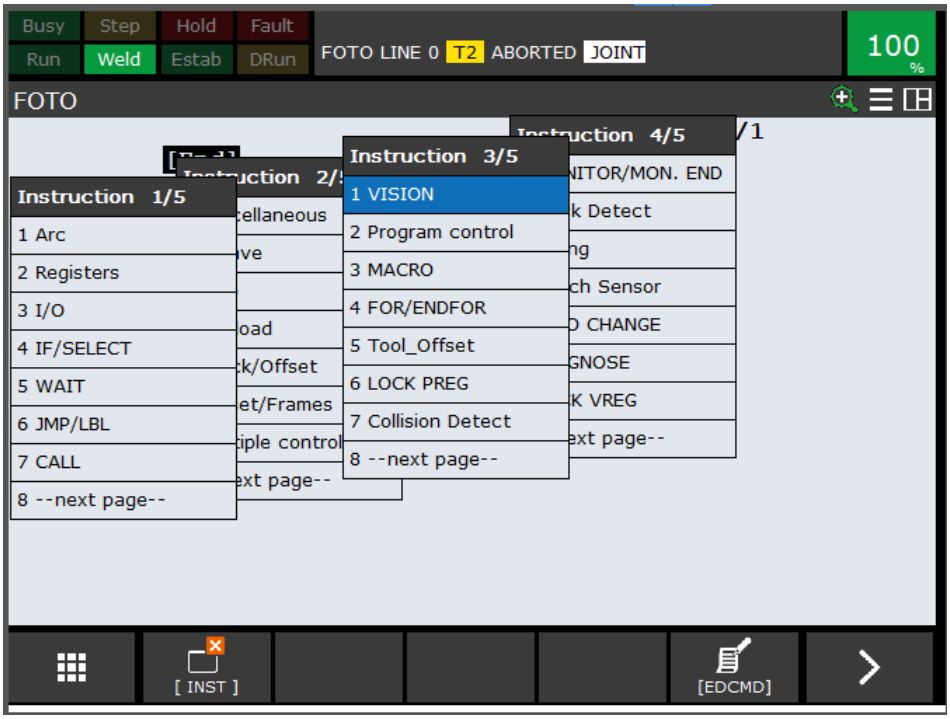


Figura 3.20: Instrucción de Visión

- **Seleccionamos en RUN FIND**

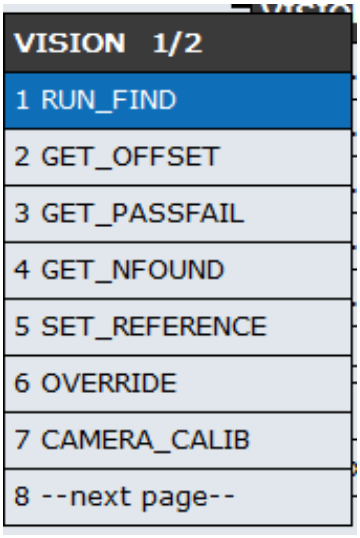


Figura 3.21: Instrucción RUN FIND

- **En este apartado seleccionamos el proceso de visión creado previamente.**
- **Nota:** Nuestro proceso de visión es **TESTPROGRAM** el cual configuramos en el **Anexo[1]** junto con la calibración de la cámara.

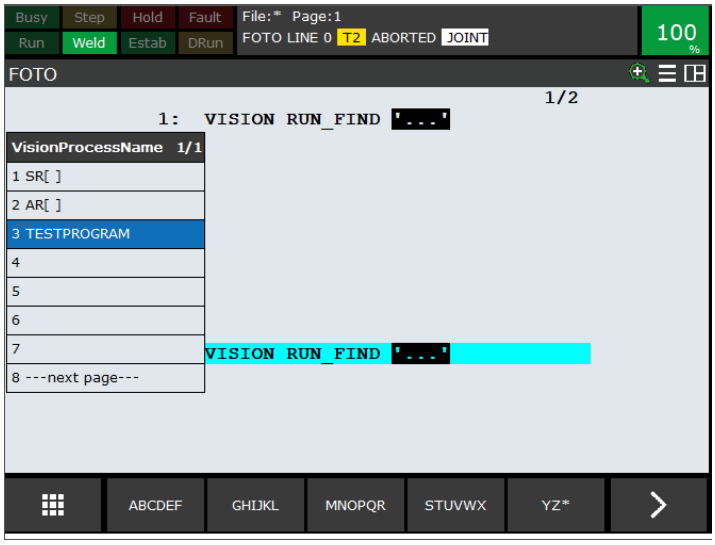


Figura 3.22: Proceso de Visión

De esta manera, la instrucción **RUN FIND** ejecutará el proceso de visión configurado, capturando la imagen correspondiente y almacenándola automáticamente en la ruta previamente seleccionada en la **Figura 3.17**.

Filezilla para la visualización de imágenes y rutas

FileZilla es un cliente FTP que se utiliza para establecer una conexión segura con el controlador y acceder a su sistema de archivos mediante protocolos como FTP o SFTP. A través de esta herramienta es posible visualizar carpetas, rutas y archivos (como imágenes), lo cual permite realizar pruebas de descarga y verificación de estructuras de directorio.

Estas pruebas iniciales permiten identificar y validar las rutas necesarias para la posterior implementación de scripts en Python, los cuales automatizarán la extracción y el análisis de dichos archivos, optimizando el flujo de trabajo y reduciendo la intervención manual.



Figura 3.23: Software Filezilla

Extracción de imágenes mediante Python

Para automatizar la obtención de imágenes capturadas por el sistema de visión del controlador FANUC, se desarrolló un script en Python utilizando el protocolo FTP (Revisar **Anexo [4]**). Este proceso permite acceder remotamente a los registros de imágenes generados durante operaciones robotizadas, identificar automáticamente el subdirectorio más reciente y extraer la imagen más actual considerada válida. Esta funcionalidad es clave para tareas como la calibración, validación de piezas y análisis visual, ya que elimina la necesidad de intervención manual en la transferencia de archivos desde el controlador.

La conexión con el controlador se establece mediante la biblioteca estándar ftplib, utilizando la IP del robot y credenciales anónimas. Una vez establecida la conexión, se accede al directorio raíz de registros de imágenes (`ud1:\vision\log`).

```
18  ftp = FTP()  # Crea una instancia del cliente FTP
19  ftp.connect(ftp_host, 21)  # Conecta al servidor FTP en el puerto 21 (puerto por defecto de FTP)
20  ftp.login(ftp_user, ftp_pass)  # Realiza el login con las credenciales proporcionadas
21  print(f'Conectado a {ftp_host}')  # Imprime mensaje de confirmación de la conexión
```

Figura 3.24: Establecer conexión Servidor FTP

Dentro de este directorio, el script emplea una expresión regular para localizar los subdirectorios que siguen un patrón temporal específico de nomenclatura: **y/AA/MMM/DD**. Este patrón consiste en:

- y: prefijo fijo.
- AA: año en dos dígitos (por ejemplo, 24 para 2024).
- MMM: abreviatura del mes en letras minúsculas (por ejemplo, apr para abril).
- DD: día del mes en dos dígitos.

Este formato facilita la identificación cronológica de los registros. El código relevante para filtrar y ordenar los directorios es el siguiente:

```
33  # Expresión regular para identificar directorios con el formato 'yAAMMMDD'
34  pattern = re.compile(r'y\d{2}[a-z]{3}\d{2}')  # Patrón para coincidir con las fechas en los directorios
35  # Listar todos los directorios y filtrar los que coinciden con el patrón
36  directorios = ftp.nlst()  # Obtiene lista de nombres de directorios en el directorio actual
37  directorios_validos = sorted(  # Filtra los directorios que coinciden con el patrón
38      [d for d in directorios if pattern.fullmatch(d)],
39      reverse=True)  # Ordena los directorios por fecha de manera descendente
```

Figura 3.25: Identificación de último directorio

Una vez identificado el directorio más reciente, se accede a una subcarpeta interna llamada **Sub00000**, donde se encuentran las imágenes. Se listan los archivos con extensión .png y se filtran aquellas imágenes "buenas", entendidas como las que no terminan con una secuencia de seis dígitos, ya que generalmente las imágenes válidas o útiles en este contexto finalizan con un número impar:

```
68  # Filtra las imágenes buenas (que no terminan con un número par)
69  imagenes_buenas = [f for f in imagenes if not re.search(r'\d{6}$', f)]
```

Figura 3.26: Filtro de imágenes

De estas imágenes, se selecciona la más reciente (última en orden alfabético) y se descarga localmente para su análisis posterior:

```
84  # Descarga la imagen buena y la guarda localmente
85  # Abre el archivo local para escritura en modo binario
86  with open(ruta_local, 'wb') as archivo:
87      # Descarga el archivo desde el servidor FTP
88      ftp.retrbinary(f'RETR {ultima_imagen_buena}', archivo.write)
```

Figura 3.27: Descarga de imagen más reciente

Este procedimiento garantiza una extracción precisa y automática de las últimas imágenes relevantes desde el sistema de visión, lo cual es esencial para tareas de validación, inspección y análisis visual dentro del entorno de producción robotizado.

3.7 Dificultades Técnicas

Dificultades Técnicas Servidor FTP

Durante el desarrollo del proyecto, se presentó una problemática relacionada con la puesta en marcha del servidor FTP, la cual se manifestaba mediante el mensaje de error "FTP Server(s) not started", como se muestra en la **Figura 3.28**. Esta condición indicaba que el servidor no lograba inicializarse correctamente. La activación de dicho servidor era un requisito explícito para poder utilizar la libreria **iRVision** la cual nos permite la implementación del sistema de visión artificial (Calibración y configuración), por lo que su correcto funcionamiento resultaba indispensable para el avance del proyecto.

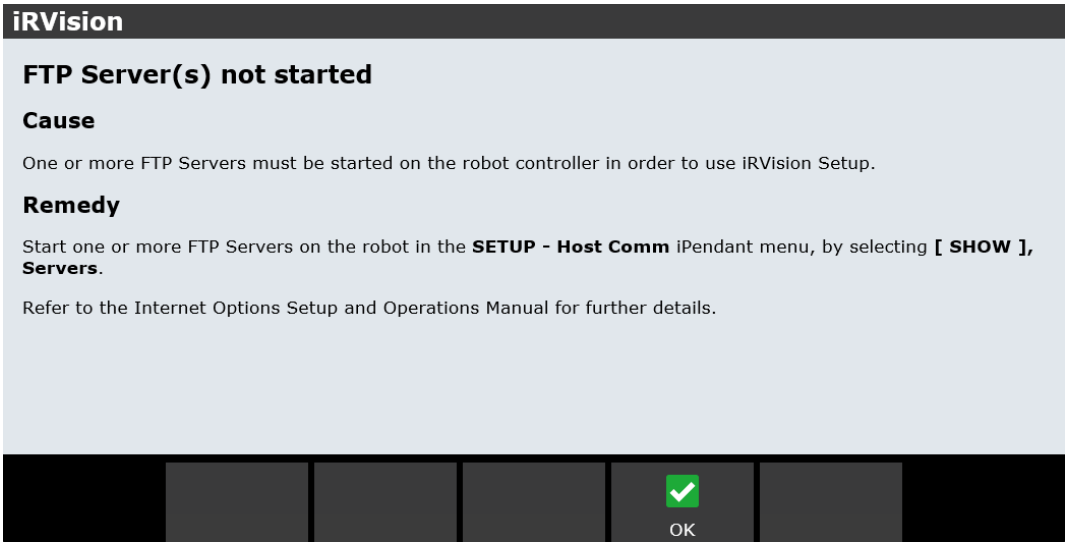


Figura 3.28: Error servidor FTP

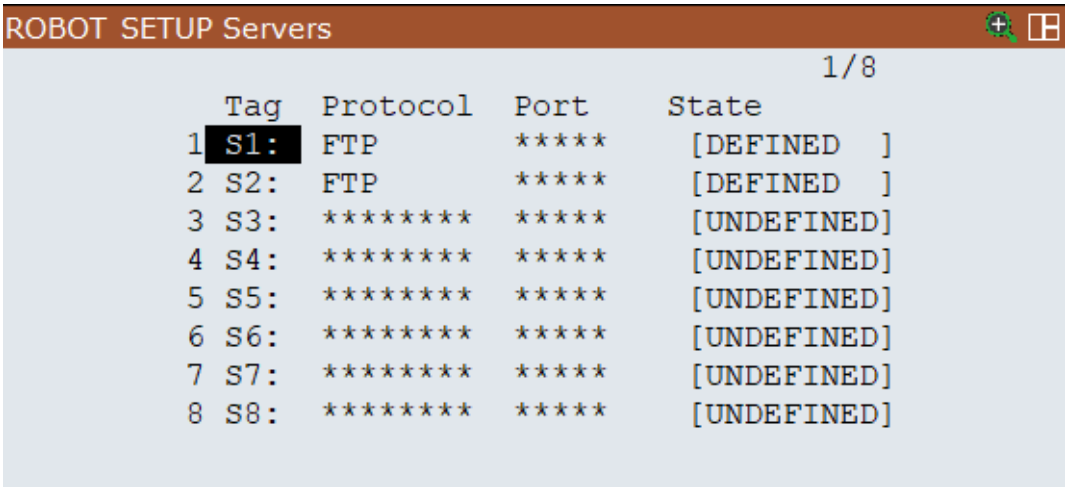


Figura 3.29: Servidores FTP sin iniciar

Una vez identificado que los servidores FTP del controlador FANUC no se encontraban en ejecución, se procedió a iniciar manualmente el servicio, tal como se ilustra en la **Figura 3.30**. Sin embargo, esta acción resultó en un mensaje de error que indicaba el fallo de la operación como se indica en la **Figura 3.31**. Ante esta situación, se optó por configurar un nuevo servidor FTP adicional, siguiendo el procedimiento oficial provisto por FANUC para el modelo de controlador específico utilizado en este proyecto (Revisar **Anexo [6]**). A pesar de realizar correctamente la configuración y el proceso de inicio, el intento de activación del nuevo servidor produjo el mismo error reportado anteriormente, lo que evidenció una posible limitación o conflicto a nivel del sistema del controlador.

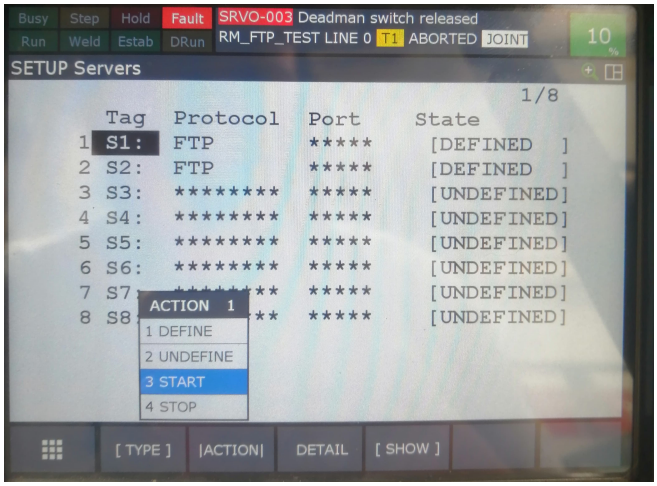


Figura 3.30: Inicializar Servidor FTP

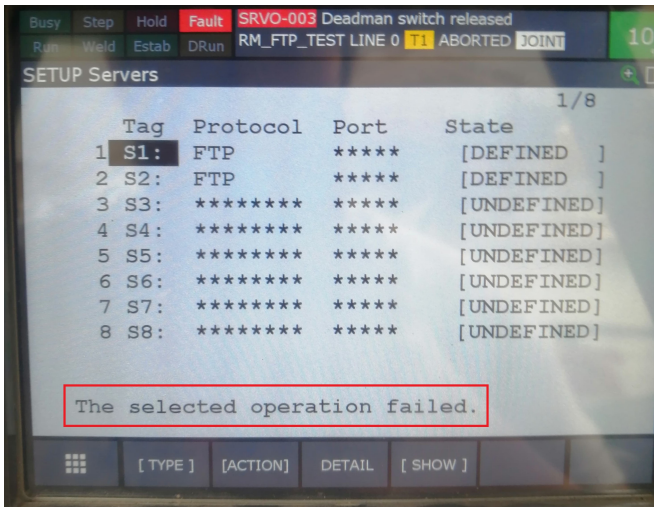


Figura 3.31: Error al iniciar Servidor FTP

A pesar de realizar una investigación exhaustiva sobre la causa de este inconveniente, se encontró escasa documentación técnica disponible al respecto. Ante esta situación, se optó por establecer contacto directo con el soporte técnico de la empresa FANUC. En conjunto con el personal especializado, se llevaron a cabo diversas pruebas sugeridas por la compañía con el objetivo de resolver el inconveniente; sin embargo, tras realizar las configuraciones necesarias para la correcta activación del servidor, este permanecía en estado "Defined", lo cual indicaba que el protocolo FTP y sus parámetros estaban correctamente establecidos, pero por alguna razón no se producía la transición al estado "Started", que representa la inicialización exitosa del servidor.

Finalmente, se planteó la hipótesis de que el origen del problema podría estar asociado al firmware del controlador, ya sea por una versión desactualizada o por la posible presencia de archivos corruptos, la actualización del firmware fue proporcionada por FANUC junto las instrucciones correspondientes para su correcta instalación. Una vez realizada la actualización del firmware del brazo robótico, el servidor FTP logró inicializarse correctamente, permitiendo así continuar con el desarrollo del sistema de visión artificial.

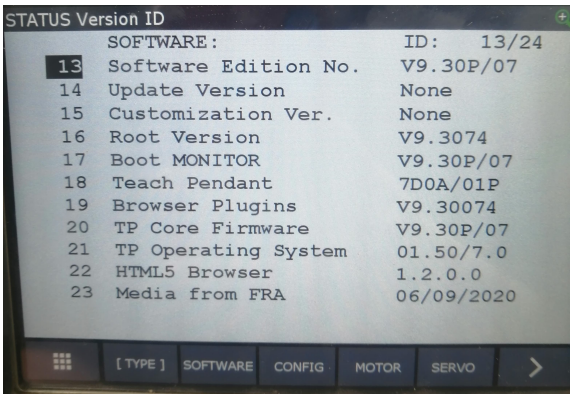


Figura 3.32: Versión antigua de firmware

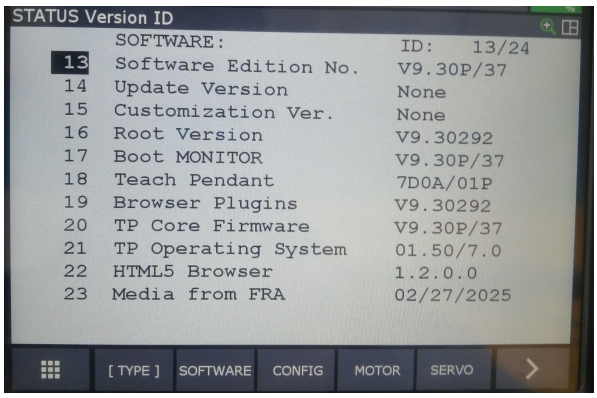


Figura 3.33: Versión actual de firmware

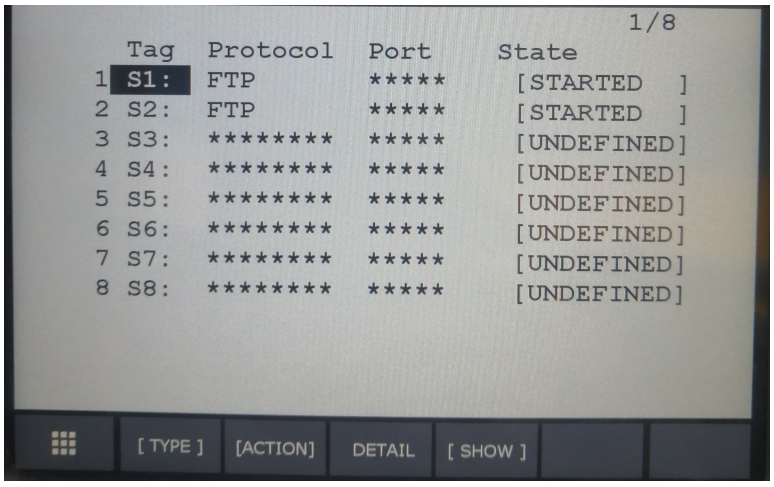


Figura 3.34: Servidores iniciados correctamente

Capítulo 4: Visión Artificial

4.1 Definición de Visión artificial

La visión artificial es una rama especializada dentro de la inteligencia artificial que tiene como objetivo dotar a las máquinas de la capacidad de percibir, interpretar y comprender información visual del entorno, de manera similar a cómo lo hace el ser humano a través de la vista. Esta disciplina combina principios de óptica, procesamiento de imágenes, aprendizaje automático y computación para desarrollar sistemas capaces de analizar imágenes o secuencias de video capturadas por cámaras o sensores especializados.

El proceso comienza con la adquisición de datos visuales, que se realiza mediante cámaras digitales, cámaras térmicas, sensores de profundidad u otros dispositivos ópticos. A continuación, las imágenes capturadas son sometidas a un conjunto de algoritmos de procesamiento digital, que pueden incluir técnicas como mejora de contraste, filtrado de ruido, detección de bordes (por ejemplo, mediante el algoritmo de Canny), segmentación de regiones y extracción de características. Estas operaciones permiten resaltar elementos clave de la imagen, como formas geométricas, texturas, colores o patrones específicos.

En etapas más avanzadas, la visión artificial se apoya en técnicas de aprendizaje automático y redes neuronales convolucionales (CNN) para realizar tareas complejas como clasificación de objetos, reconocimiento facial, detección de anomalías, o seguimiento en tiempo real. Esto permite a los sistemas tomar decisiones basadas en la interpretación visual, adaptándose incluso a variaciones en el entorno.

Esta tecnología tiene una amplia gama de aplicaciones en sectores industriales y tecnológicos. En la industria manufacturera, se utiliza para inspección de calidad automática, detección de defectos en productos, medición sin contacto, verificación de ensamblaje y control de procesos. En la robótica, permite el guiado de robots, identificación de piezas y navegación autónoma en entornos dinámicos. También es esencial en áreas como la conducción autónoma, la agricultura de precisión, la medicina (por ejemplo, en el análisis de imágenes médicas), y la seguridad, mediante sistemas de vigilancia inteligentes.

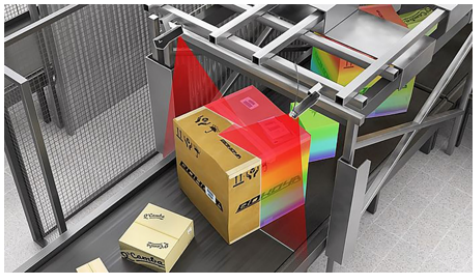
La visión artificial no solo mejora la eficiencia y la precisión de los procesos automatizados, sino que también reduce la necesidad de intervención humana en tareas repetitivas, peligrosas o que requieren alto nivel de detalle visual. A medida que avanzan las capacidades de hardware y los algoritmos de IA, esta tecnología continúa expandiendo su impacto, posicionándose como una herramienta clave en la transformación digital de múltiples industrias.

4.2 Conceptos Básicos

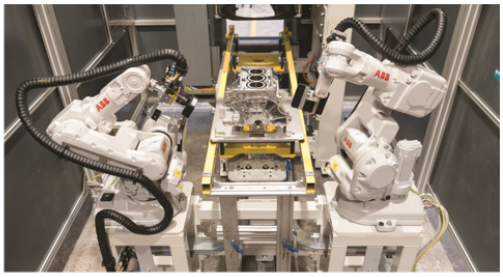
- **Captura de imágenes:** Obtención de imágenes mediante cámaras o sensores.
- **Procesamiento de imágenes:** Mejora y manipulación de imágenes utilizando técnicas como el filtro de Canny y el uso de bibliotecas como OpenCV.
- **Características visuales:** Identificación de elementos clave como bordes, colores y formas.
- **Reconocimiento de patrones:** Detectar y clasificar objetos o patrones dentro de las imágenes.
- **Aprendizaje automático:** Utilizar modelos de inteligencia artificial para mejorar la precisión de las decisiones basadas en datos visuales

4.3 Aplicaciones en la Industria

- **Inspección de calidad:** Detectar defectos en productos durante la producción.
- **Montaje automatizado:** Ensamblar piezas con precisión usando visión para guiar el brazo robótico.
- **Manipulación de objetos:** Identificar y mover objetos en almacenes o líneas de empaquetado



Inspección de Calidad



Montaje automatizado



Manipulación de objetos

Figura 4.1: Aplicaciones en la Industria

4.4 Introducción a OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto diseñada para el procesamiento de imágenes y visión por computadora. Su objetivo es proporcionar herramientas eficientes para crear aplicaciones capaces de interpretar y analizar imágenes y videos en tiempo real. Ofrece una amplia variedad de algoritmos para tareas como detección de objetos, reconocimiento facial, seguimiento de movimiento y análisis visual. Es compatible con lenguajes como Python, C++ y Java, lo que facilita su integración en diversas plataformas. Su versatilidad ha impulsado su uso en campos como la robótica, inteligencia artificial, seguridad, medicina y realidad aumentada, donde se requiere análisis visual avanzado para la toma de decisiones y automatización.

4.5 Procesamiento de Imágenes con OpenCV y Python

El procesamiento de imágenes con OpenCV representa una herramienta fundamental en el campo de la visión por computadora, permitiendo analizar y transformar imágenes digitales para extraer información relevante. Sus capacidades abarcan desde el preprocesamiento para reducir ruido y mejorar propiedades visuales, hasta la detección de bordes, contornos y la realización de transformaciones geométricas, útiles en aplicaciones como alineación de piezas o corrección de perspectiva.

OpenCV se implementa de forma eficiente a través del lenguaje Python, gracias a su sintaxis accesible, amplia comunidad de soporte y facilidad de integración con bibliotecas complementarias como NumPy, TensorFlow y scikit-learn. Esto facilita el desarrollo ágil de soluciones complejas de análisis visual sin sacrificar rendimiento.

Asimismo, OpenCV permite la segmentación de imágenes y la detección de objetos y patrones, lo cual es esencial en tareas de inspección, reconocimiento facial, seguridad y automatización. Su integración con algoritmos de aprendizaje automático y redes neuronales profundas amplía sus aplicaciones hacia sistemas inteligentes con procesamiento en tiempo real, siendo clave en áreas como la robótica, los vehículos autónomos y la industria 4.0.

En conjunto, su eficiencia, versatilidad y compatibilidad con tecnologías emergentes convierten a OpenCV, junto con Python, en una herramienta estratégica para el desarrollo de soluciones avanzadas de visión artificial las cuales utilizaremos en este proyecto (Revisar **Anexo [2]** para instalación de Python y librerías necesarias).

Una de las herramientas más empleadas en el procesamiento de imágenes es el uso de filtros, los cuales permiten modificar características espaciales o frecuenciales de la imagen para resaltar, suavizar o eliminar información específica. Estos filtros son fundamentales para tareas como la reducción de ruido, la detección de bordes o la mejora del contraste, y constituyen una etapa crucial en los sistemas de visión artificial.

4.6 Comparación entre el Filtro de Canny y Laplace

Aunque existen múltiples técnicas para la detección de bordes, el filtro de Laplace y el filtro de Canny son dos de los más utilizados debido a su simplicidad y eficacia. El filtro de Laplace, basado en derivadas de segundo orden, detecta bordes identificando regiones donde la intensidad cambia abruptamente, es decir, donde la segunda derivada se anula. Sin embargo, su sensibilidad al ruido es considerable, ya que amplifica las variaciones pequeñas en la imagen, lo que puede generar múltiples falsas detecciones si no se aplica un preprocesamiento adecuado.

En contraste, el filtro de Canny realiza un análisis más robusto al combinar suavizado Gaussiano, cálculo de gradientes, supresión de no-máximos y umbralización por histéresis. Esta secuencia de etapas permite detectar bordes con mayor precisión y continuidad, minimizando la influencia del ruido. Además, la técnica de histéresis permite preservar bordes significativos mientras se descartan los débiles o irrelevantes.

Justificación en Elección del Filtro de Canny

Dado que el presente proyecto requiere una detección de bordes confiable, especialmente en condiciones donde puede haber presencia de ruido o variaciones sutiles en las imágenes, se optó por el uso del filtro de Canny. Su capacidad para generar contornos bien definidos, junto con una alta precisión en la localización de bordes relevantes, lo convierte en una opción más adecuada que el filtro de Laplace, el cual, si bien es computacionalmente más simple, resulta menos efectivo en entornos donde la calidad y continuidad del borde son determinantes para el análisis posterior.

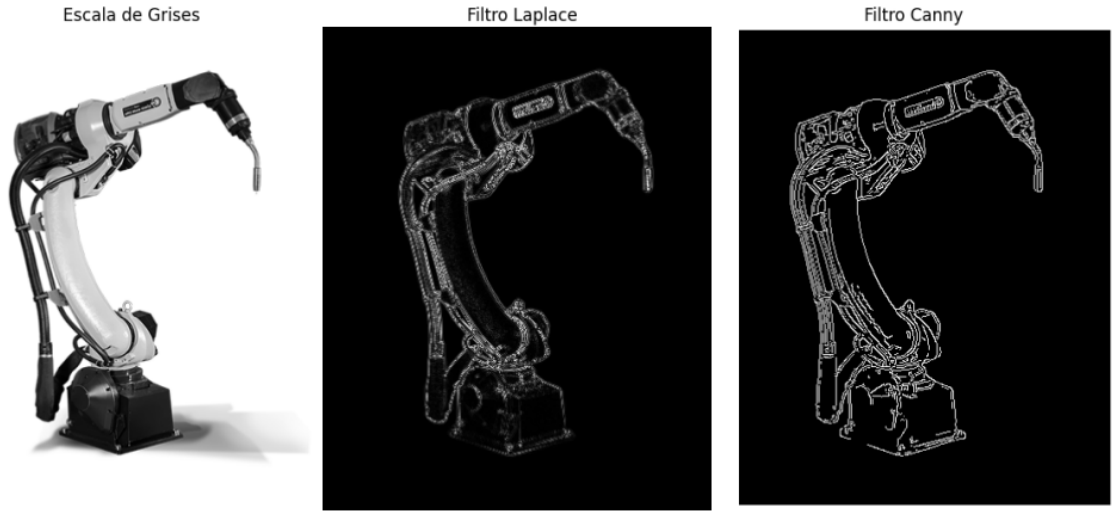


Figura 4.2: Filtro canny vs Laplace

4.7 Procesamiento de Imagen

Análisis de Imagen con OpenCV

En el contexto del procesamiento de imágenes, la función **detectar_linea_de_union()** tiene por objetivo preprocesar una imagen capturada para identificar de manera precisa los bordes asociados a una posible línea de unión entre piezas. Este procedimiento es esencial para sistemas de visión artificial que requieren un alto grado de precisión en el análisis estructural de las imágenes.

Inicialmente, se convierte la imagen a escala de grises mediante la función **cv2.cvtColor**, lo cual permite reducir la complejidad computacional al eliminar la información de color, conservando únicamente la intensidad lumínica. A continuación, se aplica un filtrado combinado: primero un suavizado gaussiano seguido de un filtro de mediana, con el fin de reducir el ruido y preservar los bordes relevantes.

```
# Transforma la imagen a escala de grises
gris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
# Filtros de suavizado
gris_suavizado = cv2.GaussianBlur(gris, (7, 7), 0)
gris_suavizado = cv2.medianBlur(gris_suavizado, 7)
```

Figura 4.3: Filtro gausiano y filtro de mediana

Posteriormente, se emplea una función de eliminación de fondo **remove()**, complementada por una operación morfológica de tipo cierre (**cv2.morphologyEx**) para rellenar regiones internas y mejorar la continuidad de las estructuras en la imagen. Este paso es fundamental para destacar los elementos de interés eliminando información irrelevante del fondo.

```
#Remover fondo de la imagen (Se conserva solo los objetos a trabajar)
fondo_removido = remove(gris_suavizado)
# rellenar regiones internas y mejorar la continuidad de las estructuras en la imagen.
fondo_removido = cv2.morphologyEx(fondo_removido, cv2.MORPH_CLOSE, kernel)
```

Figura 4.4: Extracción de fondo de imagen y operación morfológica

Una vez preprocesada, la imagen es normalizada y convertida a tipo uint8 con un único canal, asegurando su compatibilidad con las funciones de detección de bordes. En esta etapa, se realiza una filtración por nivel de intensidad, en la que se conserva únicamente el rango de grises que corresponde a tonalidades próximas al color blanco (valores altos de intensidad). Esto se implementa mediante una operación de umbral binario, generando así una máscara que aísla las regiones de alto brillo, comúnmente asociadas a áreas de interés en aplicaciones de inspección visual.

```python
# Asegurarse de que fondo_removido sea tipo uint8 y de 1 canal
if len(fondo_removido.shape) == 3:
    fondo_removido = cv2.cvtColor(fondo_removido, cv2.COLOR_BGR2GRAY)
fondo_removido = cv2.normalize(fondo_removido, None, 0, 255, cv2.NORM_MINMAX)
fondo_removido = fondo_removido.astype(np.uint8)
# Filtro para conservar solo píxeles blancos
_, mascara_blanco = cv2.threshold(fondo_removido, 140, 255, cv2.THRESH_BINARY)
# Asegurar tamaños iguales
if fondo_removido.shape != mascara_blanco.shape:
    mascara_blanco = cv2.resize(mascara_blanco, (fondo_removido.shape[1], fondo_removido.shape[0]))
```

Figura 4.5: Ajustes de la máscara para tonalidades de blanco

Esta máscara se aplica directamente sobre la imagen mediante una operación de enmascaramiento por bits (**cv2.bitwise_and**), lo que permite enfocar el análisis exclusivamente en las zonas con alta reflectancia o contraste, optimizando el resultado para la fase final.

```python
# Aplicar la máscara
fondo_removido = cv2.bitwise_and(fondo_removido, fondo_removido, mask=mascara_blanco)
```

Figura 4.6: Instrucción para aplicar la máscara

Finalmente, se ejecuta el operador Canny para la detección de bordes, herramienta ampliamente utilizada por su robustez en la localización de transiciones bruscas de intensidad. El resultado es una imagen binaria que representa los contornos predominantes, útil para la detección de líneas de unión o delimitación entre componentes.

```python
bordes = cv2.Canny(fondo_removido, 50, 150)
```

Figura 4.7: Instrucción de función Canny con imagen sin fondo

Identificación de Puntos de Trayectoria

Esta línea aplica la transformada de Hough probabilística (**cv2.HoughLinesP**) sobre la imagen de bordes previamente calculada. Esta técnica permite detectar segmentos de línea en imágenes binarizadas. Los parámetros especifican:

- **1:** resolución en píxeles del parámetro.
- **np.pi / 180:** resolución angular en radianes (1 grado)
- **threshold=80:** número mínimo de votos para que un segmento sea considerado línea.
- **minLineLength=30:** longitud mínima (en píxeles) para que se reconozca como línea.
- **maxLineGap=70**: separación máxima entre puntos alineados que pueden considerarse un mismo segmento.

```
lineas = cv2.HoughLinesP(bordes, 1, np.pi / 180,
                         threshold=80, minLineLength=30, maxLineGap=70)
```

Figura 4.8: Función transformada de Hough

Se valida si se detectaron líneas. Si no se detecta ninguna, el proceso se interrumpe. Luego se simplifica la estructura de datos del array resultante para facilitar el acceso directo a los puntos extremos de cada línea.

```
if lineas is None:
    print("No se detectaron líneas.")
    return
extremos = []
lineas = lineas[:, 0, :]  # Simplificar estructura
```

Figura 4.9: Validación detección de líneas

Se recorren todas las líneas detectadas para:

- Guardar sus extremos en una lista.
- Dibujar círculos de color sobre los extremos, permitiendo una visualización directa en la imagen de salida (**imagen_resultado).**

```
# print("Puntos extremos de las líneas detectadas:")
for x1, y1, x2, y2 in lineas:
    extremos.append(((x1, y1), (x2, y2)))
    # print(f"({x1}, {y1}) -> ({x2}, {y2})")
    cv2.circle(imagen_resultado, (x1, y1), 3, (0, 255, 0), -1)
    cv2.circle(imagen_resultado, (x2, y2), 3, (0, 165, 255), -1)
```

Figura 4.10: Imprime los puntos extremos de cada línea

Cada línea se clasifica en función del eje X, separando el extremo izquierdo del derecho. Esto es útil para calcular una línea promedio respetando la orientación de las líneas detectadas.

Posteriormente, se agrupan los puntos similares con una función externa agrupar_puntos(...), que presumiblemente elimina ruido agrupando coordenadas cercanas dentro de un umbral (**dist_umbral=4**).

```
# Se agrupan los extremos
for (x1, y1), (x2, y2) in extremos:
    if x1 < x2:
        puntos_izq.append((x1, y1))
        puntos_der.append((x2, y2))
    else:
        puntos_izq.append((x2, y2))
        puntos_der.append((x1, y1))
```

Figura 4.11: Agrupa puntos extremos

Se calculan puntos promedio para el extremo izquierdo y derecho a partir de los valores agrupados. Estos puntos representan una especie de línea central derivada de la distribución de líneas detectadas.

```
x_prom1 = int((min(x_izq_vals) + max(x_izq_vals)) / 2)
y_prom1 = int((min(y_izq_vals) + max(y_izq_vals)) / 2)
x_prom2 = int((min(x_der_vals) + max(x_der_vals)) / 2)
y_prom2 = int((min(y_der_vals) + max(y_der_vals)) / 2)
```

Figura 4.12: Cálculo promedio de puntos extremos

Se calcula la pendiente m = dy/dx. Si dx = 0, se trata de una línea vertical. En ambos casos, se generan dos nuevos puntos (punto1 y punto2) que extienden la línea promedio desde el mínimo al máximo en X, para visualizarla de forma completa.

```
# Calcular pendiente de la línea promedio
dx = x_prom2 - x_prom1
dy = y_prom2 - y_prom1
```

Figura 4.13: Cálculo de la pendiente

Se dibuja la línea promedio extendida sobre la imagen y se marcan sus extremos con colores específicos. Finalmente, se muestra el resultado en una ventana o interfaz utilizando una función personalizada mostrar_imagen.

```
# Dibujar la línea extendida y puntos
cv2.line(imagen_resultado, punto1, punto2, (0, 255, 255), 2)
cv2.circle(imagen_resultado, punto1, 5, (255, 0, 0), -1)
cv2.circle(imagen_resultado, punto2, 5, (0, 0, 255), -1)
mostrar_imagen(imagen_resultado)
```

Figura 4.14: Resultado de puntos de soldadura

Capítulo 5: Integración General

5.1 Secuencia del Proceso

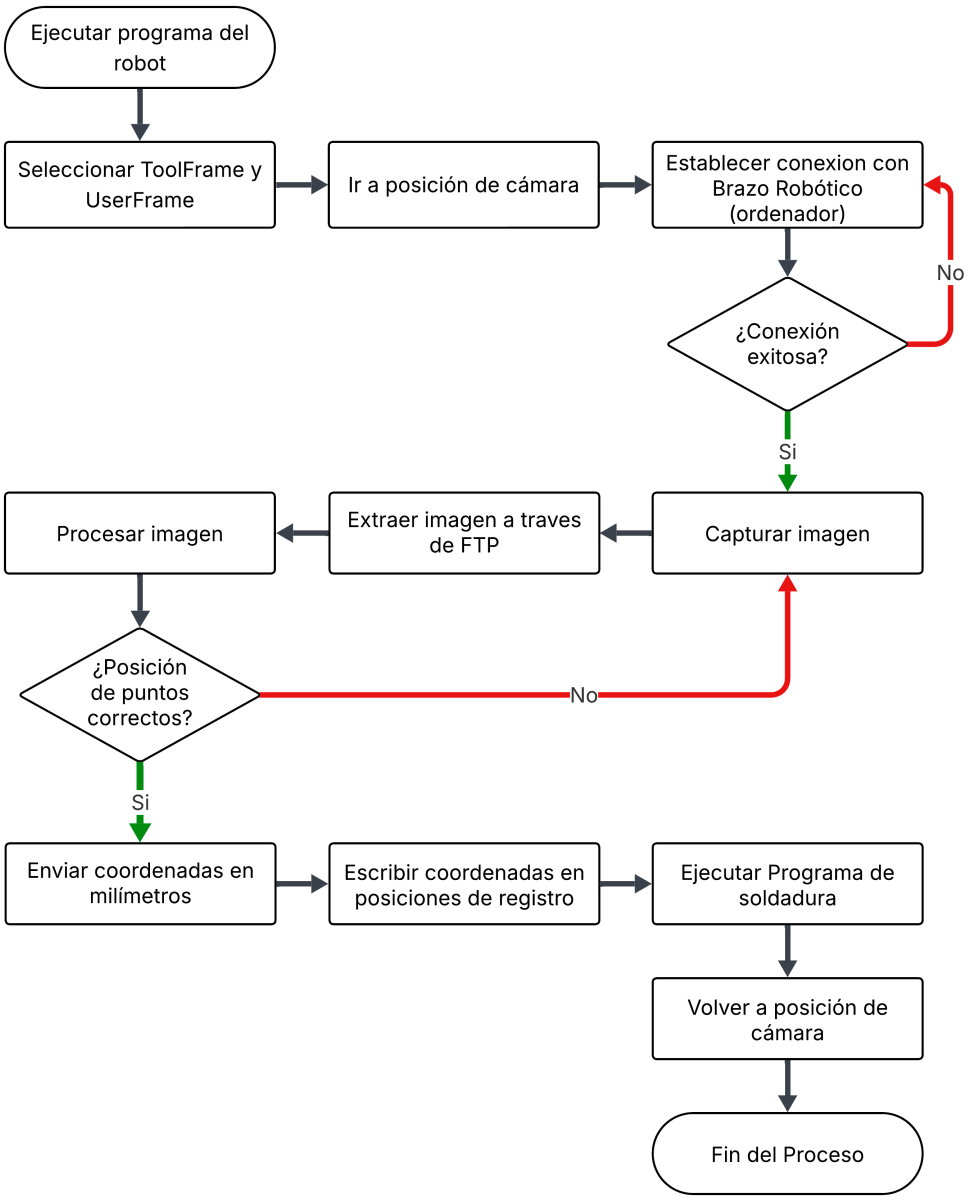


Figura 5.1: Diagrama de flujo del proceso

5.2 Descripción General del Proceso

La integración del sistema de visión artificial con el brazo robótico FANUC se materializó a través de una secuencia automatizada que permite ejecutar procesos de soldadura adaptativos. El objetivo principal fue minimizar la programación manual del robot ante variaciones en la posición y geometría de las piezas, optimizando así los tiempos de ciclo y aumentando la eficiencia operativa.

La arquitectura general del sistema combina hardware industrial (cámara, brazo robótico, controlador R-30iB Plus y soldadora MIG/MAG), software especializado (OpenCV, Python, RoboGuide, FileZilla) y protocolos de comunicación robustos (TCP/IP, FTP), permitiendo una interacción fluida entre la captura de imágenes, su análisis y la ejecución automática de trayectorias de soldadura.

5.3 Secuencia del Proceso Automatizado

El proceso se inicia con la ejecución del programa principal en el controlador del robot (Revisar **Anexo [3]**), el cual selecciona el **ToolFrame** (Revisar **Anexo[7]** para calibración) y **UserFrame** previamente calibrados, además de establecer la velocidad del robot al 100 %. El brazo se posiciona frente al área de trabajo en la posición de la cámara SC130EF2 B/W. En esta posición, el programa espera que se establezca una conexión con el ordenador mediante PCDK y pywin32 a través de un cable Ethernet.

Una vez establecida la conexión, el robot ejecuta la instrucción **RUN FIND** desde el **Teach Pendant**, capturando una imagen del área de trabajo y almacenándola en un directorio específico de la memoria USB conectada al controlador y queda a la espera. A continuación, el ordenador accede a dicha imagen mediante el script principal de Python (Revisar **Anexo [4]**) que utiliza la biblioteca **ftplib**, extrayéndola automáticamente para su análisis.

El procesamiento de la imagen se realiza con OpenCV, aplicando filtros como **Canny** y operaciones morfológicas para identificar la línea blanca entre piezas, la cual define los puntos de inicio y fin del cordón de soldadura. Si los puntos detectados cumplen con los parámetros esperados, se convierten a coordenadas cartesianas (en milímetros), tomando como referencia el sistema de coordenadas calibrado entre la cámara y el robot (**UserFrame**); en caso contrario, se envía una señal, con la cual el robot captura nuevamente otra imagen para obtener mejores resultados.

Las coordenadas se transfieren al controlador FANUC y se almacenan en las posiciones de registros del robot, además de una señal para que continúe el programa. Siguiendo la secuencia, se realiza un llamado a un programa para asignar las coordenadas enviadas a las posiciones de registros a utilizar. Finalmente, se ejecuta el programa de soldadura correspondiente, que utiliza estos valores para realizar un cordón preciso sobre la unión detectada. Tras completar la operación, el robot retorna a la posición de cámara, finalizando así el ciclo.

5.4 Coordinación de Componentes y Software

- **Brazo Robótico FANUC Arc Mate 100iD**: Ejecuta los movimientos de soldadura con una precisión de ± 0.02 mm.
- **Controlador R-30iB Plus**: Coordina la comunicación entre dispositivos, ejecuta programas de proceso y administra la red FTP.
- **Cámara SC130EF2 B/W**: Captura imágenes del área de trabajo con resolución de 1280x1024 píxeles.
- **Python + OpenCV**: Procesamiento de imágenes, extracción de características y generación de trayectorias.
- **Red FTP**: Facilitan la transferencia de imágenes desde el robot al ordenador.
- **Pywin32**: La librería PyWin32 permite a Python interactuar con las API de Windows y componentes COM. En el caso de los robots FANUC, es indispensable para utilizar el PC Developer's Kit (PCDK), ya que este se basa en tecnología COM para permitir la comunicación entre el PC y el controlador. Gracias a PyWin32, es posible acceder y controlar funciones del robot desde Python, facilitando tareas como la supervisión de estados, la manipulación de variables y la ejecución de programas de forma remota.

5.5 Consideraciones Técnicas

Durante la integración se abordaron múltiples desafíos:

- La iluminación en el área de trabajo debía mantenerse estable para garantizar una correcta segmentación de la línea de soldadura.
- La calibración del **UserFrame** y la cámara fue esencial para asegurar que las coordenadas generadas por visión artificial correspondieran correctamente al espacio físico del robot.
- Se desarrolló una estructura de protección para la cámara, fabricada en ASA mediante impresión 3D, adaptada a las condiciones térmicas y mecánicas del entorno de soldadura.
- Se utilizó un marcador blanco para metal con el cual se trazó una línea sobre los bordes comunes de las dos placas a emplear, con el propósito de facilitar el análisis de la imagen.

Capítulo 6: Resultados y Conclusiones

6.1 Resultados Obtenidos

Placa prueba 1:



Figura 6.1: Resultado primera prueba

Evidencias Visuales

- El cordón es estrecho y alargado.
- Presenta picos muy marcados y poco material depositado.
- Posible falta de fusión en los bordes del cordón.

Consecuencias Técnicas

- **Fusión deficiente**: El calor no alcanza a fundir completamente los bordes del metal base.
- **Penetración insuficiente**: El metal de aporte no penetra adecuadamente en la junta.
- **Debilidad estructural**: El cordón puede fallar bajo carga o fatiga.
- **Estética pobre**: El acabado no es uniforme ni visualmente aceptable para aplicaciones visibles.

Recomendaciones Prácticas

- Reducir la velocidad de avance para permitir una mejor fusión y mayor depósito de material.

Placa prueba 2:



Figura 6.2: Resultado segunda prueba

Evidencias Visuales

- El cordón es elevado pero no perfectamente uniforme.
- La superficie de la placa está oxidada, lo que agrava el problema de contacto eléctrico.

Consecuencias Técnicas

- **Falta de continuidad eléctrica**: El arco puede interrumpirse, provocando defectos en la soldadura.

Recomendaciones Prácticas

- Asegurar contacto metálico directo entre la masa y la pieza a soldar con prensas tipo G o similares además de evitar pintura, óxido o suciedad.
- Verificar el estado del cable de masa y su pinza, asegurando que no haya corrosión ni desgaste.
- Limpiar bien las superficies antes de soldar, especialmente en zonas de contacto eléctrico.
- Observar el comportamiento del arco: Si parpadea o suena irregular, puede ser señal de mala conexión.

Placa prueba 3:



Figura 6.3: Resultado tercera prueba

Evidencias Visuales

- El cordón presenta una anchura y altura consistentes a lo largo de su recorrido.
- Las ondas o **ripples** son regulares, lo que indica un buen control del avance y del arco.
- Tiene un perfil bien definido, sin sobreespesores ni socavados visibles.
- La forma es continua y sin interrupciones aparentes.
- Se observa brillante y limpia, lo que sugiere una buena protección gaseosa o correcta técnica de soldadura.

Consecuencias Técnicas

- **Buena fusión** entre el metal base y el material de aporte.
- **Resistencia estructural adecuada**, sin defectos visibles como porosidad o grietas.
- **Zona afectada por el calor (ZAC) controlada**, sin indicios de sobrecalentamiento.
- **Acabado superficial limpio**, sin salpicaduras ni necesidad de retrabajo.
- **Parámetros del proceso bien ajustados**, lo que garantiza uniformidad y eficiencia.

Evaluación General

Este cordón muestra una buena calidad visual, con:

- Buena fusión aparente.
- Perfil uniforme.
- Técnica controlada.

6.2 Análisis de Resultados

Reducción de Tiempos Muertos

Este proyecto contribuye a la reducción de los tiempos muertos en los procesos de soldadura, al incorporar un sistema más dinámico y adaptable para trabajar con piezas de geometrías variables. En los sistemas tradicionales de soldadura robotizada, es necesario programar manualmente el código en el controlador del brazo robótico, especificando coordenadas exactas según la posición de la pieza y la trayectoria deseada para el cordón de soldadura. En contraste, mediante la integración de tecnología de visión artificial, el robot adquiere la capacidad de ajustar automáticamente las coordenadas de soldadura a partir de una imagen capturada, ejecutando este procedimiento de manera autónoma y eficiente.

Para garantizar la precisión del proceso y prevenir posibles errores, se solicita la validación de los puntos generados a un operador humano, quien confirma que dichos puntos de la imagen sean correctos. Una vez verificados, el proceso de soldadura continúa de forma normal, optimizando significativamente el tiempo y la flexibilidad operativa.

Puntos importantes que reducen los tiempos muertos:

- **Eliminación de programación manual repetitiva**: No es necesario reprogramar el robot para cada nueva pieza.
- **Sistema adaptable en tiempo real**: La cámara detecta la geometría y posición de la pieza, generando la trayectoria automáticamente.
- **Automatización del proceso**: Desde la captura de imagen hasta la ejecución de soldadura, el sistema opera con mínima intervención humana.

Precisión en la Soldadura

El sistema implementado logra una alta precisión en el proceso de soldadura debido a los siguientes factores:

- **Alta repetibilidad del brazo robótico**: Con una precisión de ± 0.02 mm, el robot puede ejecutar trayectorias consistentes y confiables.
- **Visión artificial**: Permite identificar los puntos de inicio y fin del cordón en función de la imagen capturada, ajustando la trayectoria automáticamente a la geometría y posición de cada pieza.
- **Precisión de conversión de píxeles a milímetros**: Se definió un sistema de referencia (User Frame) y calibración de cámara que permite convertir coordenadas en píxeles a coordenadas reales del espacio con una precisión promedio de ± 0.01 mm, dependiendo de la calidad de la calibración y el posicionamiento de la cámara.
- **Validación humana**: Previo a la soldadura, los puntos generados se verifican visualmente por el operador para asegurar su precisión.

Ventajas y Desventajas del Sistema Implementado

Ventajas

- Alta precisión y repetibilidad, garantizando cordones de calidad constante.
- Flexibilidad para soldar piezas con diferentes geometrías.
- Reducción de errores humanos y aumento de la productividad.
- Integración con herramientas modernas como Python, OpenCV y comunicación vía TCP/IP.

Desventajas

- Costo inicial elevado, debido a los equipos, sensores y tiempo de integración.
- Complejidad técnica en la programación, calibración y mantenimiento del sistema.
- Menor adaptabilidad ante cambios imprevistos en el diseño de las piezas, que pueden requerir ajustes en el software o calibraciones adicionales.

6.3 Recomendaciones para Trabajos Futuros

Iluminación (constante con luces LED blancas)

- **Importancia**: La calidad de imagen en visión artificial depende de una iluminación estable.
- **Recomendación**: Instalar luces LED blancas (5000K–6500K) con difusores para evitar sombras y reflejos.
- **Control**: Usar fuentes de alimentación reguladas para evitar parpadeos de los leds.
- **Ubicación**: Fijar las luces en la celda, orientadas hacia el área de trabajo sin interferir con la cámara.

Zona de Seguridad (perímetro de la celda robótica)

Objetivo: Proteger al personal y delimitar el área de operación del robot.

Elementos sugeridos:

- Agregar paneles de protección visual (policarbonato tintado o vinilo para soldadura) para bloquear la radiación del arco hacia el exterior.
- Señalización visual y sonora.

Tapa Cubierta Protectora Automática

Función: Abrir/cerrar automáticamente una tapa protectora sincronizada con el proceso de soldadura.

Diseño sugerido:

- Material resistente al calor como el filamento ASA (Impresión 3D) o similares.
- Actuador eléctrico Servomotor SG-90.

Optimización de Código Python (visión artificial y control)

Objetivo: Mejorar eficiencia, robustez y mantenibilidad del sistema.

Sugerencias clave:

- **Modularización**: Separar el código en archivos por función (visión, FTP, robot, etc.).
- **Mejoras en visión**: Aplicar filtros adaptativos y validaciones de imagen.
- **Registro de errores**: Registrar eventos y errores para análisis posterior.
- **Precisión**: Optimizar el código de obtención de coordenadas (Adaptabilidad a diferentes figuras y formas).

Conclusiones

El informe presenta el desarrollo e implementación de un sistema de soldadura automatizada mediante visión artificial, utilizando un brazo robótico FANUC. El objetivo principal fue optimizar el proceso de soldadura MIG/MAG, permitiendo que el robot identifique de forma autónoma la geometría y posición de las piezas a soldar, eliminando la necesidad de reprogramación manual para cada nuevo componente. Esto se logró mediante la integración de una cámara industrial y el uso de herramientas de procesamiento de imágenes como OpenCV en Python, que permiten detectar bordes y generar trayectorias de soldadura precisas.

El sistema demostró ser altamente eficiente, reduciendo significativamente los tiempos muertos asociados a la programación tradicional. La precisión alcanzada en la conversión de coordenadas visuales a espaciales fue notable, gracias a una calibración rigurosa del sistema de referencia (User Frame) y a la alta repetibilidad del brazo robótico. Además, se desarrolló una infraestructura robusta que incluye una cubierta protectora para la cámara, fabricada en impresión 3D con materiales resistentes al entorno de soldadura, y una red de comunicación basada en protocolos TCP/IP y FTP para la transferencia de datos entre el robot y el sistema de visión.

Los resultados experimentales evidenciaron una mejora sustancial en la calidad de los cordones de soldadura, especialmente en términos de uniformidad, fusión y acabado superficial. Se identificaron también algunas limitaciones, como la dependencia de condiciones óptimas de iluminación y la complejidad técnica del sistema, que requiere conocimientos avanzados para su calibración y mantenimiento.

Finalmente, se proponen recomendaciones para trabajos futuros, como la implementación de iluminación LED controlada, mejoras en la modularización del código y el desarrollo de mecanismos automáticos de protección para la cámara. En conjunto, el proyecto representa un avance significativo hacia la automatización inteligente de procesos industriales, combinando robótica, visión artificial y programación avanzada para lograr un sistema flexible, preciso y eficiente.

Anexos

Anexo 1: Configuración del Sistema de Visión, Calibración de Cámara y Proceso de Visión

Calibración de Cámara

Para llevar a cabo la calibración de la cámara de visión artificial dentro del controlador del robot, es necesario seguir una secuencia de pasos definidos que aseguren una configuración precisa.

Para ilustrar este proceso de forma didáctica y sin comprometer el equipo físico, se utilizará un ejemplo desarrollado en el entorno de programación offline de FANUC, conocido como ROBOGUIDE. Este software permite simular con alta precisión una celda de trabajo robótica, incluyendo el controlador, el robot, periféricos y componentes como cámaras, replicando de manera casi idéntica el comportamiento de un entorno industrial real. De este modo, es posible validar procedimientos como la calibración de visión artificial antes de implementarlos directamente en la planta.

Paso 1: Seguir la siguiente ruta desde el teach pendant.

- **Menú → iRVision → Vision Setup**



Figura 7.1: Vision Setup

Paso 2: Seleccionar la opcion **CREATE**, asignar nombre del dispositivo y seleccionar el tipo de cámara a utilizar en la sección **Camera Data** (En nuestro caso **2D Camera**).

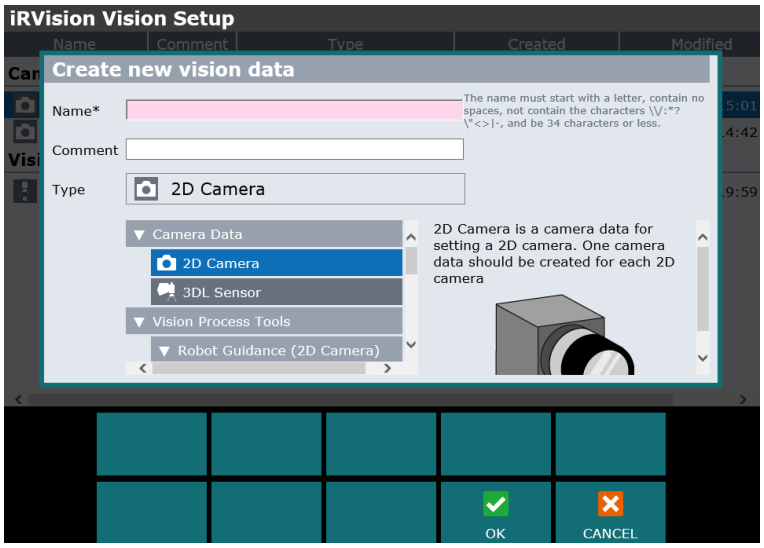


Figura 7.2: Camera Setup

Paso 3: En este paso asignamos las opciones que se muestran en la Figura 7.3.

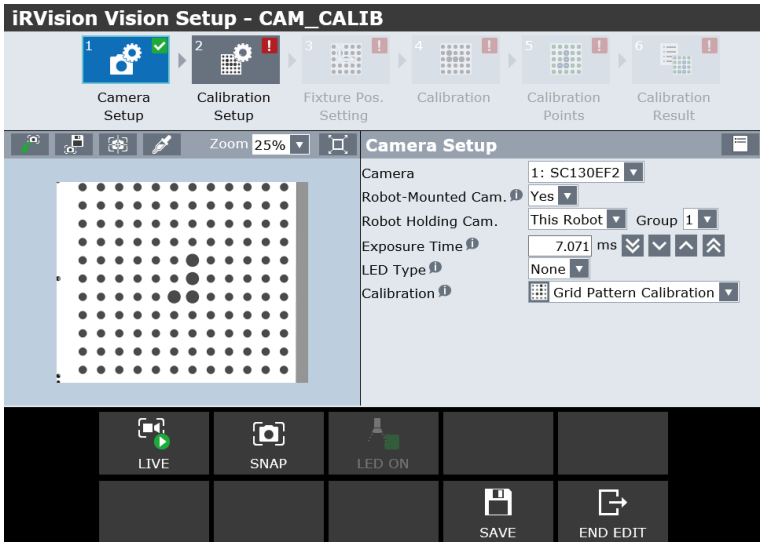


Figura 7.3: Calibration Setup

Paso 4: En este paso seleccionamos la opción **SET**, como se muestra en la figura 7.4.

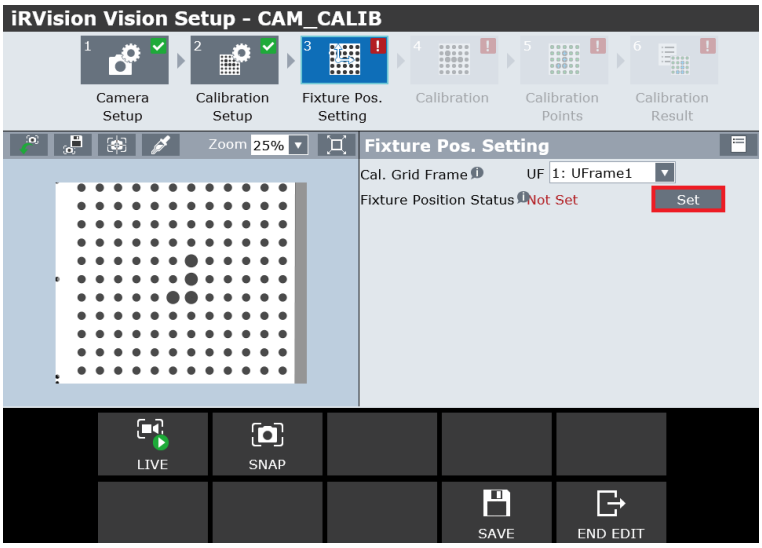


Figura 7.4: Fixture Pos. Setting

Paso 4.1: En la figura 7.5 se pueden apreciar los resultados fixture pos setting.

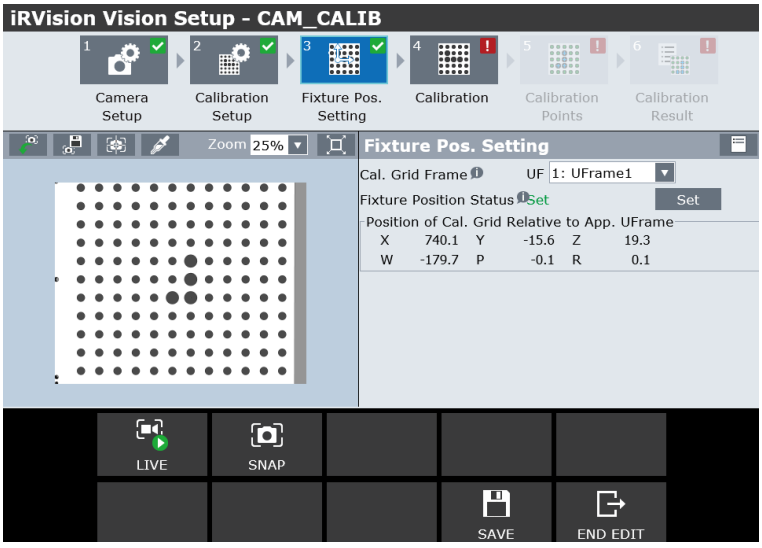


Figura 7.5: Resultados Fixture Pos Setting

Paso 5: En este paso seleccionamos la opción **SNAP**, para capturar una imagen en tiempo real de la cámara con el Grid, luego presionamos en **FIND**.

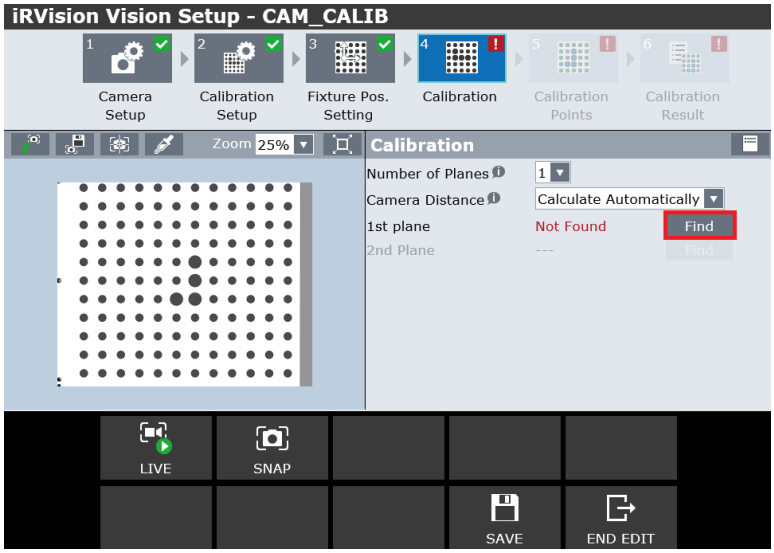


Figura 7.6: Configuración de calibración

Paso 5.1: En este paso enmarcamos los límites del grid y presionamos **Ok**.

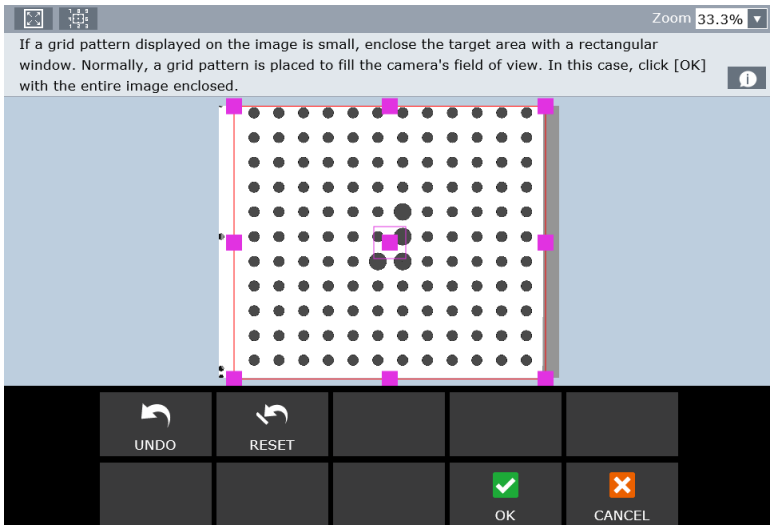


Figura 7.7: Seleccionar área del grid

Paso 5.2: Si todo ha salido bien, deberíamos visualizar marcadores verdes en el centro de cada círculo en el grid.

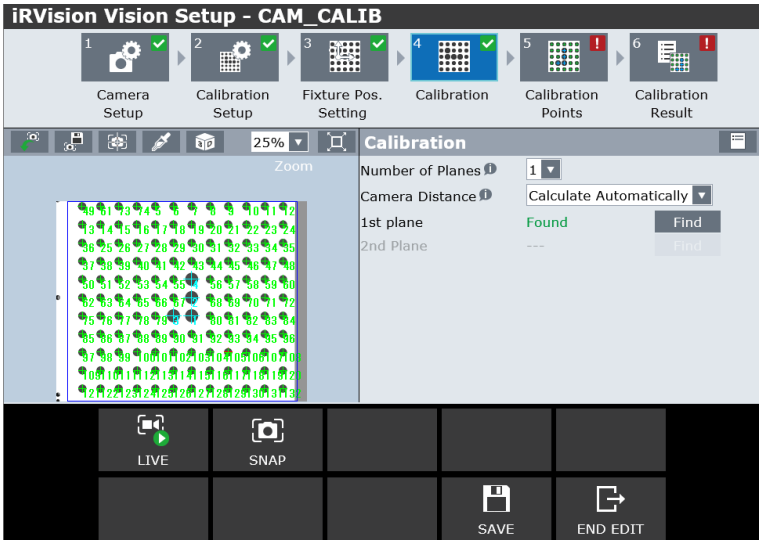


Figura 7.8: Resultados de puntos de calibración

Paso 6: Finalmente podemos visualizar los resultados de la calibración en la Figura 7.9 y Figura 7.10.

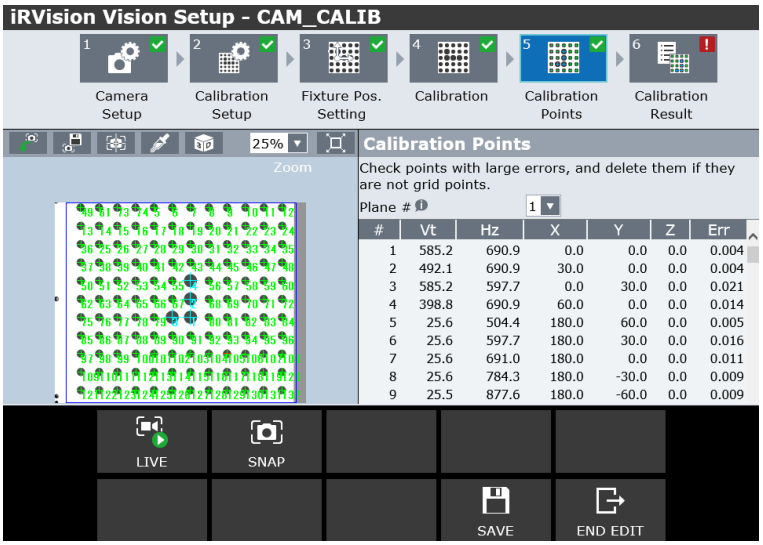


Figura 7.9: Puntos de calibración

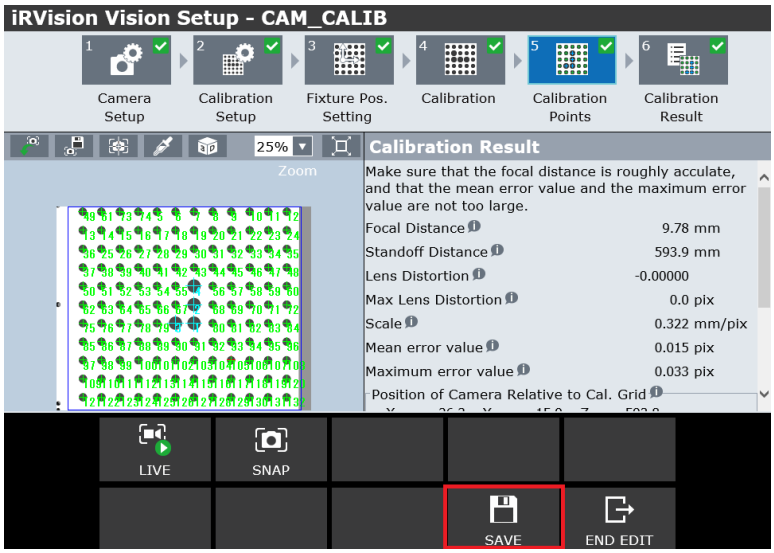


Figura 7.10: Resultados de calibración

Para finalizar el proceso de calibración presionamos en **"SAVE"** para guardar los cambios realizados.

Proceso de Visión

El proceso de visión constituye un componente esencial en los sistemas de automatización de soldadura, ya que permite la captura de imágenes. Esta captura se realiza mediante instrucciones específicas integradas en la programación TP (Teach Pendant) del robot, lo que permite sincronizar la obtención de datos visuales con las acciones del proceso de soldadura. Para lograrlo, es imprescindible crear y configurar correctamente el proceso de visión que se va a utilizar, considerando aspectos como la calibración de la cámara, la definición del tipo de búsqueda, el área de interés, y la calidad de imagen requerida. Esta configuración se adapta según las características del entorno de trabajo, el tipo de cámara empleada y los objetivos del proceso.

A continuación, se detallan los pasos necesarios para llevar a cabo dicha configuración, en concordancia con los requerimientos técnicos del sistema desarrollado.

Paso 1: Seguir la misma ruta que seguimos para llegar a la calibración de la cámara desde el teach pendant como se muestra en la Figura 7.1.

Paso 2: Seleccionar la opción **CREATE**, asignar nombre de la configuración y seleccionar el tipo de proceso de visión a utilizar en la sección **Vision Process** (En nuestro caso **2D Single-View Vision Process**) y le damos **OK**.

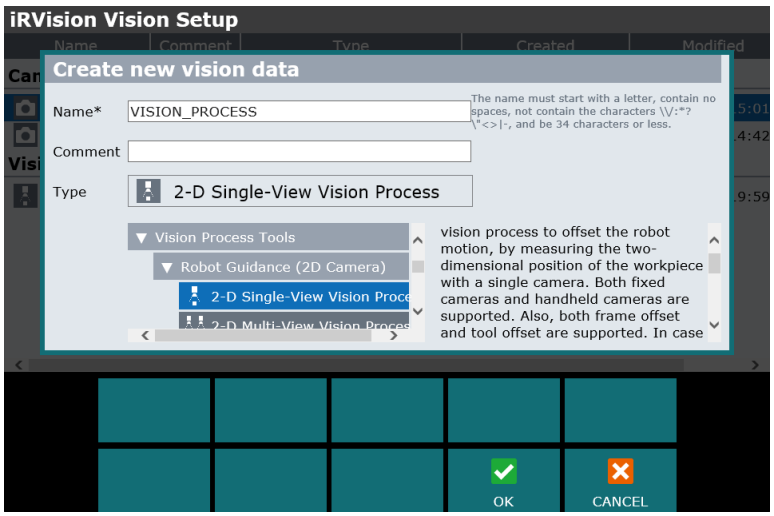


Figura 7.11: Seleccionar visión de proceso

Paso 3: En este paso colocamos un objeto de referencia en nuestra área de trabajo y asignamos las opciones que se muestran en la Figura 7.12 (El apartado **Camera** asignamos la calibración que realizamos anteriormente).

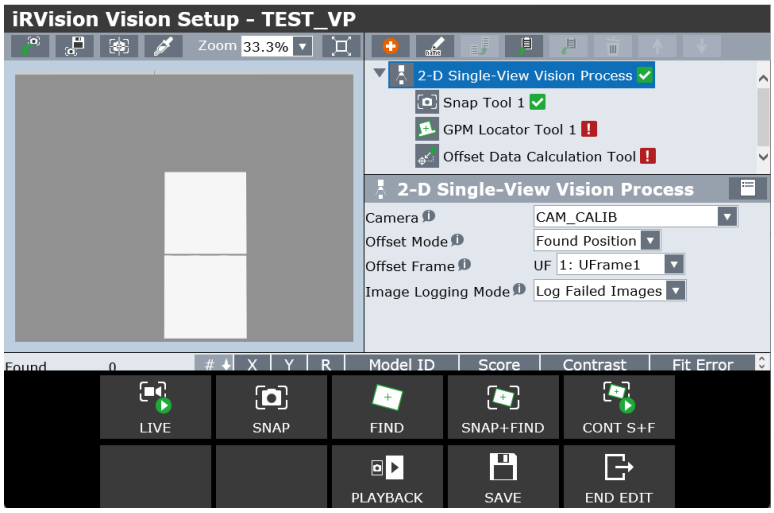


Figura 7.12: Configuraciones iniciales

Paso 4: En el apartado **Snap Tool 1** nos aseguramos de seleccionar la opción **Image Display Mode**, como se muestra en la figura 7.13.

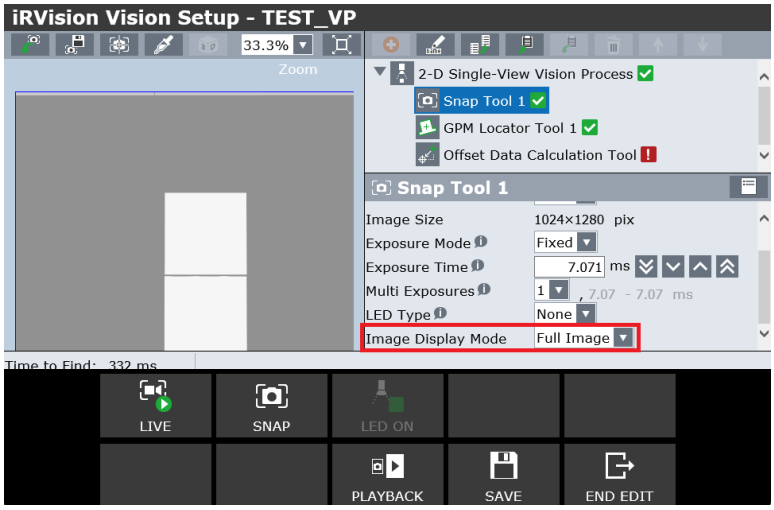


Figura 7.13: Configuración Snap tool 1

Paso 5: En este apartado presionamos el botón **Teach** para seleccionar el área donde se encuentra nuestro objeto de referencia y seleccionamos **OK**, con esto este apartado está listo.

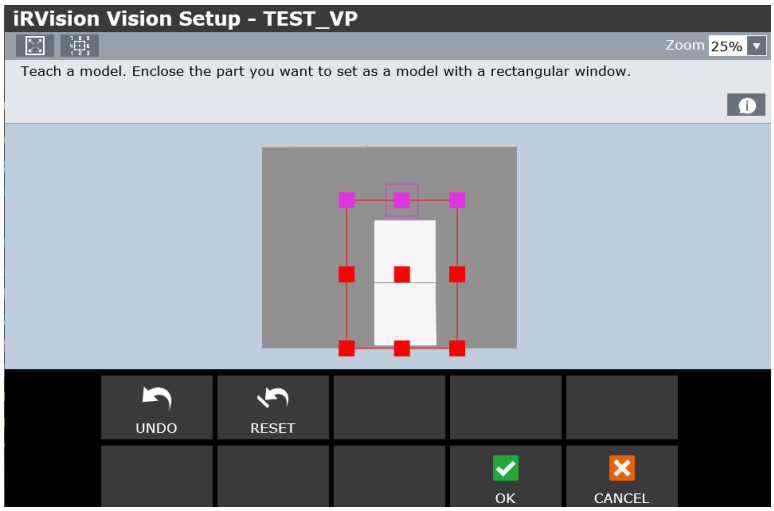


Figura 7.14: Delimitación objeto de referencia

Paso 6: Pasamos al último paso que es ingresar la altura en milímetros del objeto que usaremos.

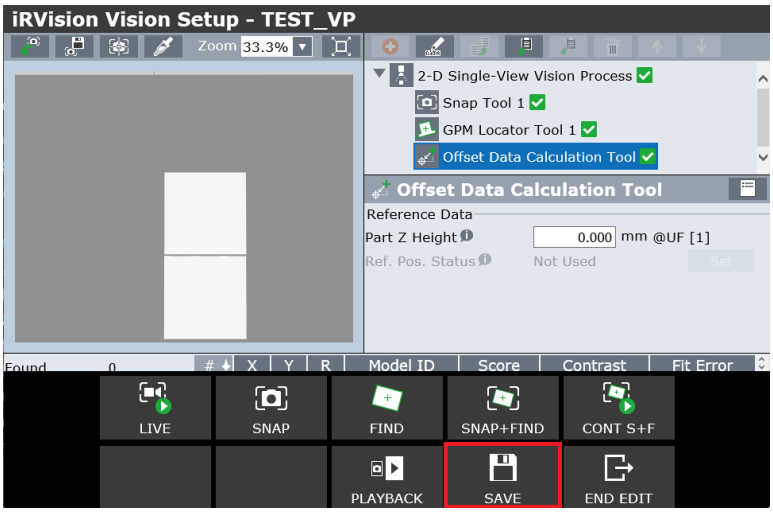


Figura 7.15: Guardar configuración

Anexo 2: Instalación de Software y Librerías

Instalación de lenguaje Python

A continuación, se detalla el procedimiento para instalar Python 3.12.8 en el equipo que operará con sistema operativo Windows de 64 bits, incluyendo la herramienta **pip**, utilizada para la gestión e instalación de paquetes.

1. Ingresar al sitio oficial de Python: `https://www.python.org/downloads/release/python-3128/`.
2. Descargar el instalador para Windows x64, específicamente el archivo **python-3.12.8-amd64.exe**.
3. Ejecutar el instalador descargado. En la ventana inicial:
   - Marcar la casilla **Add Python 3.12 to PATH**.
   - Seleccionar **Install Now** o **Customize installation** si se desea cambiar la ubicación o componentes opcionales.
4. Esperar a que finalice la instalación. Una vez completada, presionar **Close**.
5. Verificar la instalación abriendo la terminal (`CMD` o `PowerShell`) y ejecutando:

   ```
   python --version
   ```

   La salida esperada debe ser: `Python 3.12.8`

6. Confirmar que `pip` se encuentra disponible con:

   ```
   pip --version
   ```

   Esto mostrará la versión de `pip` instalada por defecto.

7. Actualizar `pip` a la última versión disponible con el siguiente comando:

   ```
   python -m pip install --upgrade pip
   ```

8. Para comprobar la funcionalidad de `pip`, se recomienda instalar un paquete de prueba, por ejemplo:

   ```
   pip install numpy
   ```

Este procedimiento asegura una instalación completa y funcional de Python 3.12.8, con acceso a herramientas de gestión de paquetes necesarias para entornos de desarrollo y automatización.

Instalación y configuración de OpenCV y librerías adicionales

Una vez completada la instalación de Python 3.12.8 y con la herramienta **pip** operativa, se procede a instalar una serie de librerías esenciales para el funcionamiento del sistema de visión artificial y comunicación con el entorno del robot. A continuación, se detalla el procedimiento paso a paso, ejecutando cada instrucción desde la terminal de Windows (CMD o PowerShell):

1. **Instalar OpenCV con módulos adicionales:** La biblioteca **OpenCV** permite el procesamiento de imágenes y visión por computadora. Se instalará la versión extendida que incluye módulos contrib:

   ```
   pip install opencv-contrib-python
   ```

2. **Instalar NumPy:** **NumPy** es una librería fundamental para el cálculo numérico en Python. Permite la manipulación eficiente de matrices y vectores, y es requerida por OpenCV y otras dependencias:

   ```
   pip install numpy
   ```

3. **Instalar Pillow (PIL):** **Pillow** es una biblioteca para la manipulación de imágenes, basada en el antiguo PIL (Python Imaging Library). Es utilizada para la carga, edición y guardado de imágenes en múltiples formatos:

   ```
   pip install pillow
   ```

4. **Instalar rembg:** **rembg** permite eliminar automáticamente el fondo de una imagen. Se instala una versión específica compatible con Python 3.12.8:

   ```
   pip install rembg==2.0.65
   ```

5. **Instalar onnxruntime (dependencia de rembg) onnxruntime** es el motor de inferencia que utiliza **rembg** para ejecutar modelos de aprendizaje automático. La versión indicada es compatible con el sistema operativo y versión de Python:

   ```
   pip install onnxruntime==1.20.0
   ```

6. Instalar pywin32 para interfaces con Windows La librería **pywin32** permite el acceso a funciones nativas de Windows desde Python, como la automatización de aplicaciones mediante **win32com.client**, útil en entornos industriales:

   ```
   pip install pywin32
   ```

Este procedimiento asegura que el entorno Python esté completamente preparado para ejecutar scripts relacionados con la visión artificial, el preprocesamiento de imágenes, y la comunicación con el controlador del sistema robotizado.

Instalación de PC Interface en controlador

Verificar si la opción está instalada

- MENU → NEXT → SYSTEM → STATUS → VERSION ID → ORDER FI.
- Busca si aparece la opción **R641 - PC Interface**.
- Si no está, deberás adquirirla a través de FANUC.

Instalar la opción (si no está presente)

- FANUC proporciona un archivo de licencia (Código **PAC**) junto con el paso a paso para la liberación de la opción R641 - PC Interface.

Instalación de PCDK y Roboguide

- **Nota:** Para instalar PC Developer’s Kit (PCDK) el procedimiento es similar a la instalación de Roboguide.

FANUC ROBOGUIDE

OPERATOR'S MANUAL

B-83234EN/03

Anexo 3: Código Brazo Robótico

Código TP – Programa Principal de Soldadura con Visión Artificial

```
 1:  LBL[2]
 2:  UFRAME_NUM=1
 3:  UTOOL_NUM=5
 4:  PR[11,4:P_START_WELD]=0
 5:  PR[12,4:P_END_WELD]=0
 6:  OVERRIDE=100%
 7:  R[9:REG_SEC]=0

 8:  !Posicion Toma de Foto
 9:L PR[10:POS_CAM] 100mm/sec FINE
10:  WAIT   2.00(sec)
11:  WAIT R[9:REG_SEC]=10

12:  !Captura de Foto
13:  VISION RUN_FIND 'TESTPROGRAM'
14:  WAIT   2.00(sec)
15:  WAIT (R[9:REG_SEC]=20 OR R[9:REG_SEC]=21)
16:  IF R[9:REG_SEC]=21,JMP LBL[2]
17:  IF R[9:REG_SEC]=20,JMP LBL[1]
18:  LBL[1]
19:  WAIT   2.00(sec)
20:  CALL ASIGNAR_R
21:  GO[1:Current]=(120)

22:  !Pinchazo inicial ;
23:L PR[14:ESC_POINT] 100mm/sec FINE
24:L PR[12:P_END_WELD] 300cm/min FINE
25:  DO[81:Weld start      ]=ON
26:  WAIT   1.30(sec)
27:  DO[81:Weld start      ]=OFF
28:L PR[14:ESC_POINT] 100mm/sec FINE

29:  !Pinchazo final
30:L PR[13:POINT_APROX] 100mm/sec FINE
31:L PR[11:P_START_WELD] 300cm/min FINE
32:  DO[81:Weld start      ]=ON
33:  WAIT   1.30(sec)
34:  DO[81:Weld start      ]=OFF
35:L PR[13:POINT_APROX] 200mm/sec FINE
36:  PR[11,4:P_START_WELD]=(-20)
```

```
 37:  PR[12,4:P_END_WELD]=(-20)
 38:  WAIT   1.00(sec)

 39:  !Soldar cordon
 40:L PR[11:P_START_WELD] 300cm/min FINE
 41:  WAIT   1.00(sec)
 42:  Weave Sine[1.3Hz,3.0mm,0.500s,0.500s]
 43:  DO[81:Weld start      ]=ON
 44:L PR[12:P_END_WELD] 12cm/min FINE
 45:  Weave End
 46:  DO[81:Weld start      ]=OFF
 47:  WAIT   1.00(sec)
 48:L PR[14:ESC_POINT] 100mm/sec FINE
 49:  WAIT   1.00(sec)
 50:L PR[10:POS_CAM] 100mm/sec FINE
 51:  OVERRIDE=10%
[End]
```

Código TP – Programa Asignación de coordenadas

```
  1:  PR[11,1:P_START_WELD]=R[10:P_START_O_X]
  2:  WAIT   1.00(sec)
  3:  PR[11,2:P_START_WELD]=R[11:P_START_O_Y]
  4:  WAIT   1.00(sec)
  5:  PR[12,1:P_END_WELD]=R[13:P_END_O_X]
  6:  WAIT   1.00(sec) ;
  7:  PR[12,2:P_END_WELD]=R[14:P_END_O_Y]
  8:  WAIT   1.00(sec)
  9:  PR[13,1:POINT_APROX]=R[10:P_START_O_X]
 10:  WAIT   1.00(sec)
 11:  PR[13,2:POINT_APROX]=R[11:P_START_O_Y]
 12:  WAIT   1.00(sec)
 13:  PR[14,1:ESC_POINT]=R[13:P_END_O_X]
 14:  WAIT   1.00(sec)
 15:  PR[14,2:ESC_POINT]=R[14:P_END_O_Y]
 16:  WAIT   1.00(sec)
[End]
```

Anexo 4: Código General de Python

Código Python – Análisis y Procesamiento de imagen

```
# Importación de librerías necesarias
from ftplib import FTP  # Para conexión FTP con el controlador FANUC
import tempfile  # Para guardar la imagen descargada temporalmente
import re  # Para detectar nombres con patrón de fecha
import cv2  # OpenCV para procesamiento de imágenes
import numpy as np  # Para operaciones numéricas con matrices
# Componentes GUI
from tkinter import Tk, filedialog, Button, Label, Frame, Canvas
import tkinter as tk  # Interfaz gráfica
from PIL import Image, ImageTk  # Para mostrar imágenes en Tkinter
from rembg import remove  # Para eliminar fondo de imágenes
import win32com.client  # Para automatización con componentes COM en Windows

# Variables globales
# Lista de puntos seleccionados (posiblemente para análisis)
puntos_seleccionados = [200]
# Kernel para operaciones morfológicas
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
imagen_capturada = None  # Imagen obtenida desde cámara o FTP
imagen_path = None  # Ruta temporal de la imagen descargada

# Función para capturar imagen desde cámara conectada
def capturar_imagen():
    global imagen_capturada, imagen_resultado, imagen_path
    cap = cv2.VideoCapture(1)  # Inicia captura desde cámara (índice 1)
    if not cap.isOpened():
        print("No se pudo abrir la cámara.")
        return

    ret, frame = cap.read()  # Captura un frame
    cap.release()  # Libera la cámara

    if ret:
        imagen_capturada = frame
        imagen_resultado = frame  # Se puede redimensionar si se desea
        mostrar_imagen(imagen_resultado)  # Muestra la imagen en la GUI
    else:
        print("No se pudo capturar la imagen.")
```

```
# Función para descargar la última imagen válida desde el controlador
# FANUC vía FTP
def descargar_imagen_fanuc():
    global imagen_capturada, imagen_path
    ftp_host = '192.168.1.110'  # IP del controlador FANUC
    ftp_user = 'anonymous'  # Usuario FTP
    ftp_pass = ''  # Contraseña vacía
    log_root_dir = r'ud1:\vision\log'  # Ruta base de logs de visión
    sub_dir = 'sub00000'  # Subdirectorio donde se almacenan imágenes

    try:
        ftp = FTP()
        ftp.connect(ftp_host, 21)  # Conexión al servidor FTP
        ftp.login(ftp_user, ftp_pass)
        ftp.cwd(log_root_dir)  # Cambia al directorio raíz de logs

        # Busca directorios con nombre que coincida con patrón de fecha
        pattern = re.compile(r'y\d{2}[a-z]{3}\d{2}')
        directorios = ftp.nlst()
        directorios_validos = sorted([d for d in directorios
                                      if pattern.fullmatch(d)], reverse=True)

        if not directorios_validos:
            print("No se encontraron directorios válidos.")
            ftp.quit()
            return

        ultimo_dir = directorios_validos[0]
        ruta_final = f'{log_root_dir}\\{ultimo_dir}\\{sub_dir}'

        try:
            ftp.cwd(ruta_final)  # Entra al subdirectorio más reciente
        except Exception as e:
            print(f'No se pudo entrar a {ruta_final}: {e}')
            ftp.quit()
            return

        archivos = ftp.nlst()
        # Filtra imágenes PNG
        imagenes = [f for f in archivos if f.lower().endswith('.png')]
        # Excluye imágenes con nombres pares
        imagenes_buenas = [f for f in imagenes
                           if not re.search(r'im\d{6}[02468]\.png$', f)]

        if not imagenes_buenas:
```

```
            print("No se encontraron imágenes buenas.")
            ftp.quit()
            return

        imagenes_buenas.sort()
        # Selecciona la más reciente
        ultima_imagen_buena = imagenes_buenas[-1]

        # Guarda la imagen temporalmente
        temp_file = tempfile.NamedTemporaryFile(delete=False, suffix='.png')
        with open(temp_file.name, 'wb') as archivo:
             # Descarga la imagen
            ftp.retrbinary(f'RETR {ultima_imagen_buena}', archivo.write)

        ftp.quit()

        # Carga la imagen con OpenCV
        imagen_path = temp_file.name
        imagen_capturada = cv2.imread(imagen_path)
        if imagen_capturada is not None:
            # Se puede redimensionar si se desea
            imagen_redimensionada = imagen_capturada
            mostrar_imagen(imagen_redimensionada)
            print(f'Descargando imagen --> {ultima_imagen_buena}')
        else:
            print("No se pudo cargar la imagen descargada.")

    except Exception as e:
        print(f'Error durante descarga: {e}')

# Función que retorna la imagen capturada si está disponible,
# o la carga desde el disco si solo se tiene la ruta
def obtener_imagen():
    if imagen_capturada is not None:
        return imagen_capturada  # Devuelve la imagen ya cargada en memoria
    elif imagen_path is not None:
        # Carga la imagen desde la ruta almacenada
        return cv2.imread(imagen_path)
    return None  # Si no hay imagen ni ruta, retorna None

# Función que agrupa puntos cercanos entre sí según una distancia umbral
def agrupar_puntos(puntos, dist_umbral=50):
    grupos = []  # Lista para almacenar grupos de puntos
    for p in puntos:
        # Bandera para saber si el punto fue agregado a un grupo
```

```
            agregado = False
            for grupo in grupos:
                # Calcula la distancia entre el punto y el centro del grupo
                if np.linalg.norm(np.array(p) -
                                  np.mean(grupo, axis=0)) < dist_umbral:
                    grupo.append(p)  # Agrega el punto al grupo existente
                    agregado = True
                    break
            if not agregado:
                grupos.append([p])  # Crea un nuevo grupo con el punto actual
        # Retorna el punto promedio (centroide) de cada grupo como tupla de enteros
        return [tuple(map(int, np.mean(grupo, axis=0))) for grupo in grupos]

    def detectar_linea_de_union():
        # Variables globales
        global imagen_resultado
        global offs_start_x, offs_start_y, offs_end_x, offs_end_y
        # Llamado funcion de obtencion de imagen
        imagen = obtener_imagen()
        if imagen is None:
            print("No se ha seleccionado ninguna imagen.")
            return

        # Transforma la imagen a escala de grises
        gris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        # Filtros de suavizado
        gris_suavizado = cv2.GaussianBlur(gris, (7, 7), 0)
        gris_suavizado = cv2.medianBlur(gris_suavizado, 7)

        #Remover fondo de la imagen (Se conserva solo los objetos a trabajar)
        fondo_removido = remove(gris_suavizado)
        # rellenar regiones internas y mejorar la continuidad de las estructuras
        # en la imagen.
        fondo_removido = cv2.morphologyEx(fondo_removido, cv2.MORPH_CLOSE, kernel)

        # Asegurarse de que fondo_removido sea tipo uint8 y de 1 canal
        if len(fondo_removido.shape) == 3:
            fondo_removido = cv2.cvtColor(fondo_removido, cv2.COLOR_BGR2GRAY)
        fondo_removido = cv2.normalize(fondo_removido, None, 0, 255, cv2.NORM_MINMAX)
        fondo_removido = fondo_removido.astype(np.uint8)
        # Filtro para conservar solo píxeles blancos
        _, mascara_blanco = cv2.threshold(fondo_removido, 140, 255, cv2.THRESH_BINARY)
        # Asegurar tamaños iguales
        if fondo_removido.shape != mascara_blanco.shape:
            mascara_blanco = cv2.resize(mascara_blanco, (fondo_removido.shape[1],
```

```
                                                    fondo_removido.shape[0]))

# Aplicar la máscara
fondo_removido = cv2.bitwise_and(fondo_removido, fondo_removido,
                                 mask=mascara_blanco)

bordes = cv2.Canny(fondo_removido, 50, 150)

lineas = cv2.HoughLinesP(bordes, 1, np.pi / 180,
                         threshold=80, minLineLength=30, maxLineGap=70)

if lineas is None:
    print("No se detectaron líneas.")
    return
extremos = []
lineas = lineas[:, 0, :]  # Simplificar estructura

imagen_resultado = imagen.copy()

# print("Puntos extremos de las líneas detectadas:")
for x1, y1, x2, y2 in lineas:
    extremos.append(((x1, y1), (x2, y2)))
    # print(f"({x1}, {y1}) -> ({x2}, {y2})")
    cv2.circle(imagen_resultado, (x1, y1), 3, (0, 255, 0), -1)
    cv2.circle(imagen_resultado, (x2, y2), 3, (0, 165, 255), -1)

# Calcular extremos promedio
# Calcular extremos promedio para la línea central
# (izquierdo y derecho por separado)
# Calcular extremos promedio para la línea central
# (izquierdo y derecho por separado)

puntos_izq = []
puntos_der = []
todos_x = []

# Se agrupan los extremos
for (x1, y1), (x2, y2) in extremos:
    if x1 < x2:
        puntos_izq.append((x1, y1))
        puntos_der.append((x2, y2))
    else:
        puntos_izq.append((x2, y2))
        puntos_der.append((x1, y1))
```

```
    todos_x.extend([x1, x2])

# Agrupar y reducir puntos similares
puntos_izq_filtrados = agrupar_puntos(puntos_izq, dist_umbral=4)
puntos_der_filtrados = agrupar_puntos(puntos_der, dist_umbral=4)

# Calcular punto medio entre mínimo y máximo de cada lado
x_izq_vals, y_izq_vals = zip(*puntos_izq_filtrados)
x_der_vals, y_der_vals = zip(*puntos_der_filtrados)

x_prom1 = int((min(x_izq_vals) + max(x_izq_vals)) / 2)
y_prom1 = int((min(y_izq_vals) + max(y_izq_vals)) / 2)
x_prom2 = int((min(x_der_vals) + max(x_der_vals)) / 2)
y_prom2 = int((min(y_der_vals) + max(y_der_vals)) / 2)

# Calcular pendiente de la línea promedio
dx = x_prom2 - x_prom1
dy = y_prom2 - y_prom1

if dx == 0:
    # Línea vertical
    punto1 = (min(todos_x), y_prom1)
    punto2 = (max(todos_x), y_prom1)
else:
    m = dy / dx  # pendiente
    x_min = min(todos_x)
    x_max = max(todos_x)
    y_min_ext = round(m * (x_min - x_prom1) + y_prom1)
    y_max_ext = round(m * (x_max - x_prom2) + y_prom2)
    punto1 = (x_min, y_min_ext)
    punto2 = (x_max, y_max_ext)

# Dibujar la línea extendida y puntos
cv2.line(imagen_resultado, punto1, punto2, (0, 255, 255), 2)
cv2.circle(imagen_resultado, punto1, 5, (255, 0, 0), -1)
cv2.circle(imagen_resultado, punto2, 5, (0, 0, 255), -1)
mostrar_imagen(imagen_resultado)

# Datos de entrada
pixel_coords_1, pixel_coords_2 = punto1, punto2
center_pixel = (717, 513)
pixel_ref_y1, pixel_ref_y2 = (448, 244), (987, 243)
pixel_ref_x1, pixel_ref_x2 = (448, 244), (447, 784)
dist_mm_y, dist_mm_x = 153, 153
```

```
    # Escalas mm/pixel
    mm_per_pixel_y = dist_mm_y / np.linalg.norm(np.array(pixel_ref_y2)
                                                - np.array(pixel_ref_y1))
    mm_per_pixel_x = dist_mm_x / np.linalg.norm(np.array(pixel_ref_x2)
                                                - np.array(pixel_ref_x1))

    # Conversión a mm con ajustes
    def convert(pixel_coords):
        delta = np.array(pixel_coords) - np.array(center_pixel)
        mm = delta * np.array([mm_per_pixel_y, mm_per_pixel_x])
        mm[1] = -mm[1]
        mm = mm[::-1]
        mm[0] += 0
        mm[1] += 1
        return mm

    mm_coords_1 = convert(pixel_coords_1)
    mm_coords_2 = convert(pixel_coords_2)

    # Variables finales
    offs_start_x, offs_start_y = mm_coords_1
    offs_end_x, offs_end_y = mm_coords_2

    print(f"Inicio: ({offs_start_x}, {offs_start_y})")
    print(f"Fin: ({offs_end_x}, {offs_end_y})")

    # imagen_resultado = cv2.resize(imagen_lineas, (1280, 1024))
    mostrar_imagen(imagen_resultado)
# Función para mostrar una imagen de OpenCV en un canvas de Tkinter
def mostrar_imagen(imagen_cv):
    # Convierte de BGR (OpenCV) a RGB (PIL)
    imagen_rgb = cv2.cvtColor(imagen_cv, cv2.COLOR_BGR2RGB)
    # Convierte la imagen a formato PIL
    imagen_pil = Image.fromarray(imagen_rgb)
    # Convierte la imagen PIL a formato compatible con Tkinter
    imagen_tk = ImageTk.PhotoImage(imagen_pil)
    # Dibuja la imagen en el canvas
    canvas.create_image(0, 0, anchor=tk.NW, image=imagen_tk)
    # Guarda la referencia para evitar que se elimine
    canvas.image = imagen_tk

# Función que se ejecuta al presionar el botón "Continuar"
def confirmar_boton():
    global btn_9
    print("Enviando coordenadas...")
```

```
    # Accede a registros numéricos del robot
    reg_numeric10 = robot.RegNumerics(10)
    reg_numeric11 = robot.RegNumerics(11)
    reg_numeric13 = robot.RegNumerics(13)
    reg_numeric14 = robot.RegNumerics(14)

    # Asigna valores a los registros (coordenadas)
    reg_numeric10.Value.RegLong = offs_start_x
    reg_numeric11.Value.RegLong = offs_start_y
    reg_numeric13.Value.RegLong = offs_end_x
    reg_numeric14.Value.RegLong = offs_end_y
    r9.Value.RegLong = 20  # Señal de confirmación
    btn_9 = 1  # Marca que se presionó "Continuar"
    root.destroy()  # Cierra la ventana
    return btn_9

# Función que se ejecuta al presionar el botón "Cancelar"
def cancelar_boton():
    global btn_9
    print("Cancelando...", btn_9)
    r9.Value.RegLong = 21  # Señal de cancelación
    btn_9 = 2  # Marca que se presionó "Cancelar"
    root.destroy()  # Cierra la ventana
    return btn_9

# Variables de control
btn_9 = 0  # Estado del botón (1: continuar, 2: cancelar)
r9 = 0  # Registro numérico 9 del robot

# Crear objeto de conexión con el robot FANUC
robot = win32com.client.Dispatch("FRRobot.FRCRobot")

# Dirección IP del robot (en RoboGuide suele ser localhost o una IP simulada)
IP_ROBOT = "192.168.1.110"
robot.Connect(IP_ROBOT)  # Conecta al controlador

# Bucle principal de conexión
while robot.IsConnected:

    # Si se presionó continuar o cancelar, desconecta y termina
    if btn_9 == 1 or btn_9 == 2:
        if robot.IsConnected:
            robot = None  # Libera el objeto COM
            print("Desconectado correctamente.")
            break
```

```
# Si está conectado, inicializa el registro 9
if robot.IsConnected:
    print("Conectado al controlador Fanuc.")
    r9 = robot.RegNumerics(9)
    r9.Value.RegLong = 10  # Señal para iniciar proceso

# Crear ventana principal con Tkinter
root = Tk()
root.title("Detección de Bordes de Placas")
root.state('zoomed')  # Maximiza la ventana

# Crear marco lateral para botones
frame = Frame(root)
frame.pack(side=tk.LEFT, padx=10, pady=10)
# Crear canvas para mostrar la imagen
canvas = Canvas(root, width=1280, height=1024)
canvas.pack(side=tk.RIGHT, padx=10, pady=10)

# Botón para continuar
btn_continuar = Button(frame, text="Continuar", command=confirmar_boton)
btn_continuar.pack(pady=5)
# Botón para cancelar
btn_cancelar = Button(frame, text="Cancelar", command=cancelar_boton)
btn_cancelar.pack(pady=5)

# Verifica si el robot ha enviado la señal para descargar imagen
aux_r9 = robot.RegNumerics(9)
aux_r9 = aux_r9.Value.Reglong

if aux_r9 == 10:
    descargar_imagen_fanuc()  # Descarga la imagen desde el robot
    if imagen_capturada is not None:
        # Ajusta tamaño
        imagen_redimensionada = cv2.resize(imagen_capturada, (1280, 1024))
        mostrar_imagen(imagen_redimensionada)  # Muestra la imagen en la GUI
        print(f'Imagen Descargada')
        print(f'Procesando imagen...')
        # Procesa la imagen (función no incluida aquí)
        detectar_linea_de_union()
    else:
        print("No se pudo cargar la imagen descargada.")

root.mainloop()  # Inicia el bucle de eventos de la GUI
```

Anexo 5: Manual iRVision 2D R-30iB Plus

FANUC Robot series

R-30*i*B Plus CONTROLLER

*i*RVision 2D Camera Application

OPERATOR'S MANUAL

B-83914EN-2/01

Anexo 6: Manual de Operador R-30iB Funciones Ethernet

FANUC Robot series

R-30*i*A/R-30*i*A Mate/R-30*i*B CONTROLLER

Ethernet Function

OPERATOR'S MANUAL

B-82974EN/02

Anexo 7: Manual de Programación y Operación de Robots FANUC



ROBÓTICA PROGRAMACIÓN FANUC

Autor: FANUC Robotics Ibérica S.L.
Actualización y revisión: Salvador García Chinchilla

Centro de Formación _____

Anexo 8: Manual de Operador R-30iB Plus Ethernet/IP

FANUC Robot series

R-30*i*A/R-30*i*A Mate/R-30*i*B/R-30*i*B Mate/R-30*i*B Plus/
R-30*i*B Mate Plus/R-30*i*B Compact Plus CONTROLLER

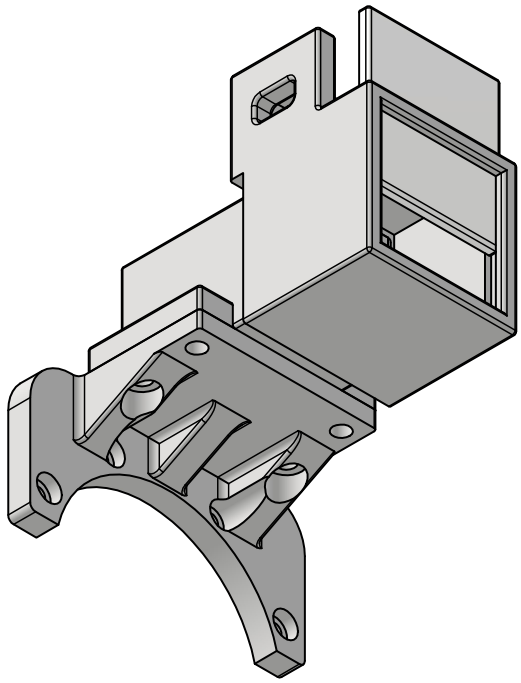
EtherNet/IP

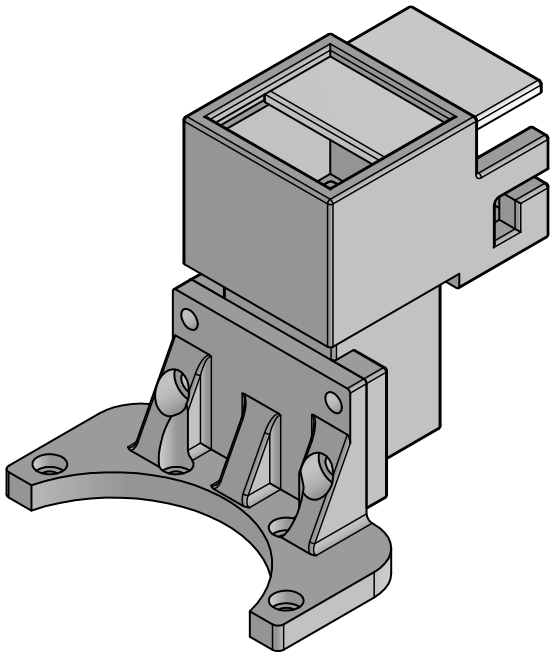
OPERATOR'S MANUAL

B-82854EN/03

Anexo 9: Planos Soporte y Cubierta Protectora

Ensamblaje Final



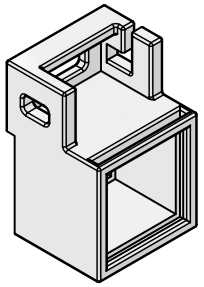


Diseño de	Revisado por	Aprobado por	Fecha	Fecha	
Abner Rodríguez - Cristobal Díaz				24-01-2025	

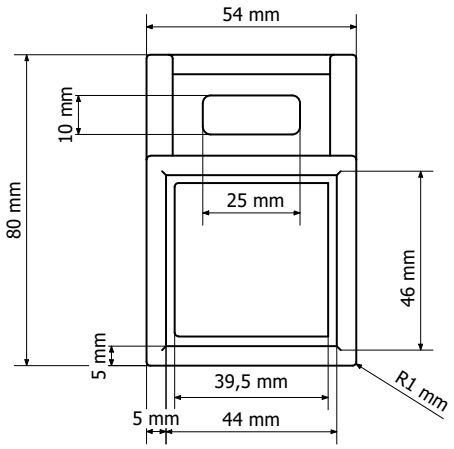
UTFSM	Plano Cubierta Protectora		
		Edición	Hoja 1 / 4

Proteccion del Lente y Base Sevomotor

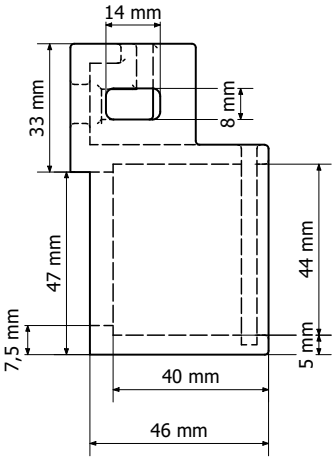
Base Servomotor



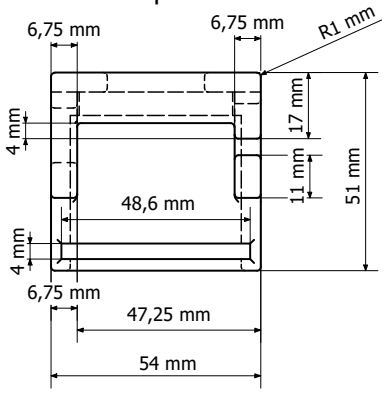
Frontal



Lateral Izq.



Superior

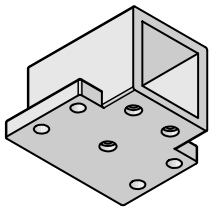


Diseño de	Revisado por	Aprobado por	Fecha		Fecha
Abner Rodríguez - Cristobal Díaz					24-01-2025

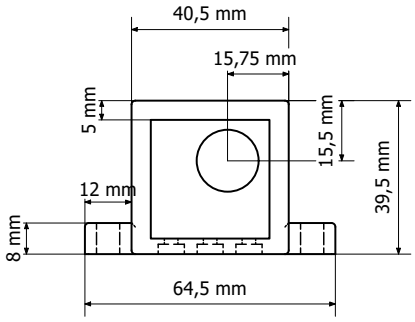
UTFSM	Plano Cubierta Protectora		
		Edición	Hoja 2 / 4

Descanso de Cámara

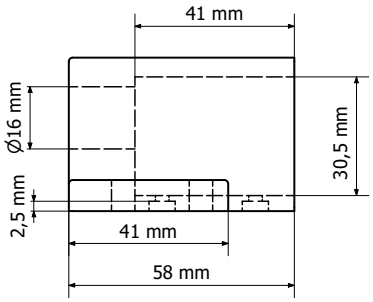
Descanso Cámara



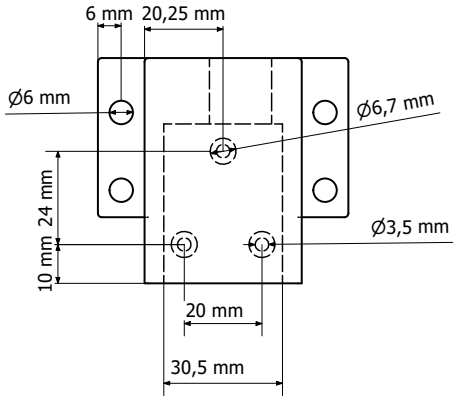
Frontal



Lateral Izq.



Superior

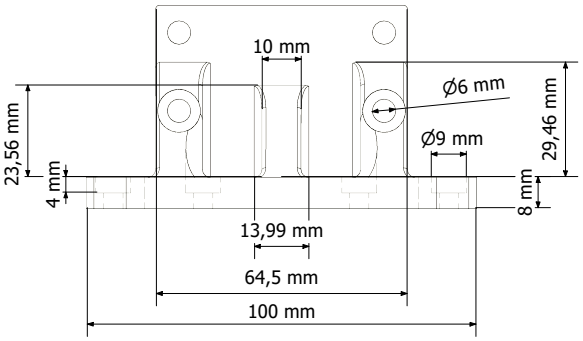


Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Abner Rodríguez - Cristobal Díaz					24-01-2025	
UTFSM				Plano Cubierta Protectora		
					Edición	Hoja 3 / 4

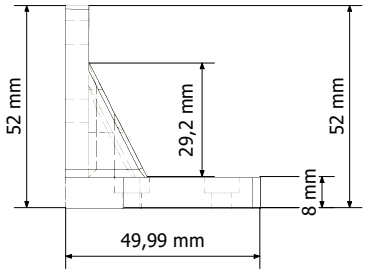
Soporte y Tapa de Cubierta Protectora

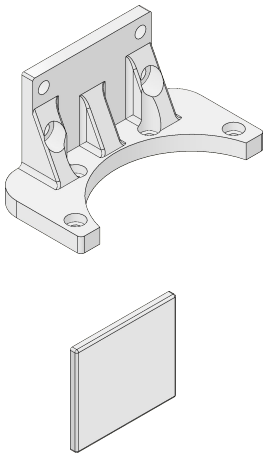
Acople Cámara

Frontal



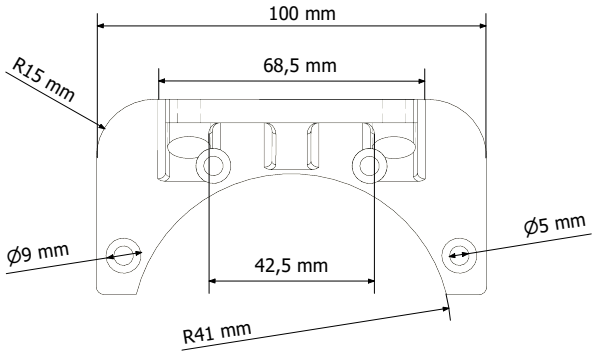
Lateral Izq.



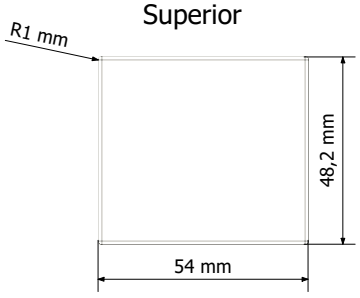


Tapa Cámara

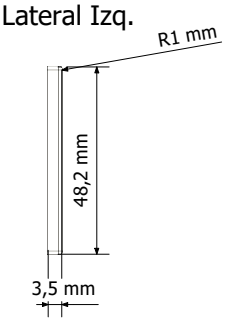
Superior



Superior



Lateral Izq.



Diseño de	Revisado por	Aprobado por	Fecha		Fecha
Abner Rodríguez - Cristobal Díaz					24-01-2025

UTFSM	Plano Cubierta Protectora		
		Edición	Hoja 4 / 4