UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE MECATRÓNICA

IMPLEMENTACIÓN DE UNA HERRAMIENTA TECNOLÓGICA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE OBJETOS PARA EL ROBOT KUKA KR5-2 ARC HW

Trabajo de titulación previo a la obtención del título de Ingeniero en Mecatrónica

AUTORES: CRISTIAN GEOVANNY JIMÉNEZ QUEVEDO
BRANDO DANIEL CABRERA BONILLA

TUTOR: ING. PAÚL ANDRÉS CHASI PESANTEZ, MSc.

CO-TUTOR: ING. JORGE OSMANI ORDOÑEZ ORDOÑEZ, MSc.

Cuenca – Ecuador

2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Cristian Geovanny Jiménez Quevedo con documento de identificación N° 1728640754 y Brando Daniel Cabrera Bonilla con documento de identificación N° 0106340169; manifestamos que:

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

Cuenca, 18 de enero del 2023

Atentamente,

Cristian Geovanny Jiménez Quevedo	Brando Daniel Cabrera Bonilla
1728640754	0**106340169**

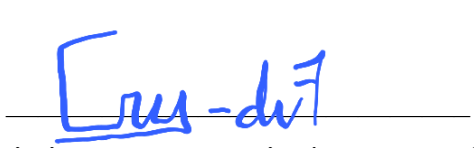
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Cristian Geovanny Jiménez Quevedo con documento de identificación N° 1728640754 y Brando Daniel Cabrera Bonilla con documento de identificación N° 0106340169, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: Implementación de una herramienta tecnológica de identificación y clasificación de objetos para el robot KUKA KR5-2 ARC HW", el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

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

Cuenca, 18 de enero del 2023

Atentamente,



Cristian Geovanny Jiménez Quevedo
1728640754

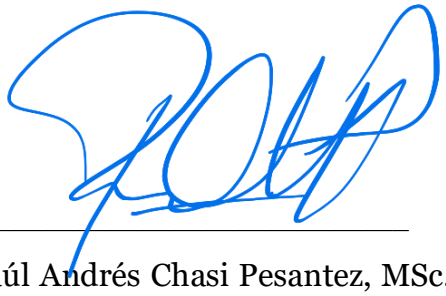
Brando Daniel Cabrera Bonilla
0106340169

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Paúl Andrés Chasi Pesantez, con documento de identificación N° 0103652095, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: IMPLEMENTACIÓN DE UNA HERRAMIENTA TECNOLÓGICA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE OBJETOS PARA EL ROBOT KUKA KR5-2 HW, realizado por Cristian Geovanny Jiménez Quevedo con documento de identificación N° 1728640754 y Brando Daniel Cabrera Bonilla con documento de identificación N° 0106340169, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 18 de enero del 2023

Atentamente,



Ing. Paúl Andrés Chasi Pesantez, MSc.
0103652095

Dedicatoria

Cristian Geovanny Jiménez Quevedo

El presente proyecto de titulación va dedicado a mi familia, por darme su apoyo incondicional, creer y valorar el esfuerzo que doy día a día por aprender algo nuevo, la convicción serena de querer mejorar la sociedad a la escala que Dios me lo permita, mediante mi conocimiento y mis acciones. Así mismo va dedicado a mi propio esfuerzo, porque desde niño he intentado seguir mis metas, no siempre lo he conseguido, pero si de algo estoy orgulloso es de saber que es muy difícil para mí rendirme ante las adversidades de la vida. Finalmente va dedicado a Dios porque siempre ha estado a mi lado para ayudarme a no desfallecer, allá en el cielo tengo personas que me cuidan y esperan mucho de mí. De mi parte este proyecto es para todos ellos.

Brando Daniel Cabrera Bonilla

Este proyecto de titulación va dedicado a todas las personas que confiaron en mi desde un principio especialmente a mis padres, que desde un inicio me mostraron su apoyo incondicional tanto en la carrera como en la vida y me dieron la oportunidad de estudiar algo que yo mismo elegí sin ningún tipo de prejuicio, además de inculcarme los valores que me permitieron seguir adelante día a día y ser una mejor persona una de la que ellos estuvieran orgullosos. De la misma forma, agradezco a Dios y la Virgen María por brindarme salud, no dejar que me desvié del camino que a lo largo de estos 5 años y de toda mi vida, y por ultimo a los muy buenos compañeros que estuvieron presentes a lo largo de toda mi formación Universitaria y compartieron conmigo varias y valiosas experiencias a lo largo de esta.

Agradecimientos

Cristian Geovanny Jiménez Quevedo

Agradezco primeramente a mis padres Carlos y Melida por darme las oportunidades que ellos por adversidades de la vida no tuvieron, sepan que estoy orgulloso de ustedes y los quiero con el alma. Agradezco a mis abuelitos porque sin su guía no sería la persona que soy hoy en día.

Al Ingeniero Paúl Chasi por confiar en mi persona para el presente proyecto y motivarme a pensar en que soy capaz de lograr grandes objetivos. Al ingeniero Osmani Ordoñez por la ayuda, paciencia y humildad que tuvo conmigo durante toda la realización del proyecto. Finalmente agradezco a todas las personas que me han inculcado tanto conocimiento como formas de ver y vivir mi vida. Gracias de corazón.

Brando Daniel Cabrera Bonilla

Agradezco a los que considero los motores de mi vida que son mis padres Clever y Yolanda por siempre apoyarme a cumplir mis sueños, sin objeción. Por ser unos ejemplos a seguir por mi persona y por estar presentes en mis derrotas y triunfos. Y por haber construido a la persona que soy hoy en día, en base a la educación y valores que me han inculcado. De la misma manera, agradezco a mi tutor de titulación el Ing. Paul Chasi por haber confiado desde un principio en nuestra capacidad y tener el compromiso en todo momento con respecto a nuestro trabajo de titulación y tomarse el tiempo de ayudarnos, e igualmente al Ing. Osmani que nos guio al principio de este trabajo y estaba pendiente de consecuentes avances que se conseguía. Por otra parte, agradezco a mi compañero de titulación Cristian con el cual hemos sido compañeros desde primer ciclo y que hoy en día estamos culminando nuestra carrera. Y por último agradezco a mi hermana Maricela por ser la persona que siempre me ha dado fuerzas para continuar en medio de la adversidad cuando nadie más lo ha hecho.

Índice

Lista de Tablas

Lista de Figuras

Resumen

En el presente trabajo de titulación se propone la implementación de una herramienta tecnológica de identificación y clasificación de objetos para el robot KUKA KR5-2 ARC HW, por medio de visión artificial por computadora mediante una cámara web, enviando información captada por la cámara hacia un microcontrolador para la activación de las entradas digitales del robot KUKA KR5-2 HW para su posterior movimiento.

La herramienta tecnológica es capaz de posicionar al robot KUKA KR5-2 ARC HW en el punto físico en el que se encuentran los diferentes objetos captados por la cámara, esto debido a que reconoce tanto su posicionamiento en el espacio de trabajo, así como las dimensiones base-altura de los diferentes objetos. Por consiguiente, la herramienta tecnológica detecta las diferentes áreas de los objetos y los clasifica en un espacio designado considerando si su tamaño es pequeño, mediano o grande.

Se utilizaron tres tipos diferentes de lenguaje de programación, Python para el uso de la visión artificial y creación de un programa en formato de código de KRL para la matriz de puntos. Programación en C para el envío de información desde Python hacia el robot KUKA KR5-2HW y finalmente la programación KRL del robot KUKA el cual cuenta con todos los puntos de la mesa de trabajo.

Palabras clave: Visión artificial, KUKA KR5-2 ARC HW, identificación, clasificación, python, C, KRL.

Abstract

This degree project proposes the implementation of a technological tool for the identification and classification of objects for the KUKA KR5-2 ARC HW robot, that use artificial vision through a webcam, sending the information captured by the camera to a microcontroller for the activation of the digital inputs of the KUKA KR5-2 HW robot for the before movement.

The technological tool can position the KUKA KR5-2 ARC HW robot at the physical point where the different objects captured by the camera are located, because it recognizes the positioning in a workspace, as well as the base-height dimensions of the different objects. Consequently, the technological tool detects the different areas of the objects and classifies them in a designated space considering that their size is small, medium, or large.

Three different types of programming language have been used, python for the use of computer vision and the creation of a program in KRL code format for the point matrix. C programming to send information from Python to KUKA KR5-2HW robot and finally the KRL programming of the KUKA robot that has all the points of the working area.

Keywords: Machine vision, KUKA KR5-2 ARC HW, identification, classification, python, C, KRL

1. Introducción

En la actualidad, existen varios procesos industriales que cuentan con la tecnología de visión artificial, esto debido a que facilita los procesos de identificación y clasificación, mejorando los tiempos de actuación de la maquinaria, así como el aumento de la flexibilidad de los procesos, además mejora el nivel de seguridad de los operarios dado que el brazo robótico puede discernir entre personas que lo operan y objetos-productos que los brazos robóticos manipulan. Existen muchas aplicaciones de sistemas robotizados los cuáles estan sujetos a constantes cambios e innovaciones mejorando la calidad del trabajo, así como la designación de los trabajadores a áreas que no demandan excesivo esfuerzo físico (Jiva, 2019).

En el presente proyecto de titulación se realizó la implementación de una herramienta tecnológica de identificación y clasificación de objetos para el robot KUKA KR5-2 ARC HW del laboratorio de robótica de la Universidad Politécnica Salesiana, mediante el uso y acople de tres lenguajes de programación: (Python, C, KRL). Los mismos que permiten relacionar al controlador KRC2 tanto con periféricos externos como realizar el control del brazo robótico mediante una computadora externa. Logrando que la herramienta tecnológica permita al brazo robótico identificar y clasificar diferentes tipos de objetos de acuerdo con las dimensiones de estos.

La implementación de una herramienta tecnológica abre diversas aplicaciones para el brazo robótico, sentando unas bases de programación que permitirán seguir explorando en la investigación del área de la robótica y como esta repercute en un proceso industrial mediante el control y optimización de estos.

2. Problema

2.1. Antecedentes

El desarrollo computacional permite la realización de investigaciones científicas y tecnológicas con el fin de facilitar actividades propias de los seres humanos. De esta manera, consiguiendo automatizar muchos procesos mecánicos, de cálculo, de almacenamiento de datos, de procesamiento, etc. Desarrollando, cada vez, herramientas de cómputo capaces de auxiliar en forma directa cada una de estas actividades. En varias de ellas se necesita examinar el medio ambiente donde se efectuará la actividad para así realizar un análisis de las situaciones y tomar una decisión siguiendo un razonamiento lógico (Takeyas, 2007).

De la misma forma, a lo largo de los años las razones más frecuentes para la evolución de la robótica han sido la necesidad de operar materiales peligrosos, como sustancias radiactivas. Por ello, los primeros robots, desarrollados en 1940, fueron industriales. Las aplicaciones de la robótica fueron evolucionando de la mano con la tecnología de los ordenadores, así pues en 1954, Devol patentó el primer robot con memoria para repetir acciones y en 1961, se creó la primera compañía productora de robots para la industria, Unimation (Camarillo, Krummel, y Salisbury, 2004). Y con el transcurso de los años, los avances tecnológicos ha alcanzado un punto donde los robots hacen uso de inteligencia artificial para operar, de tal manera que sea capaz de reconocer objetos por cuenta propia a través de una cámara; a continuación, se presenta un resumen de varios estudios que se han realizado acerca del uso de visión artificial en robots.

En el proyecto de Barahona Guamani (2019): "NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT", concluye que el algoritmo de visión artificial implementado en el Robot Kuka YouBot es capaz de reconocer objetos, en este caso la detección de la postura humana utilizando la librería de OpenCV, de la misma forma que propone el uso de diferentes software los cuales son capaces de reconocer objetos y dependiendo el uso que se le quiera dar cada uno tendrá sus ventajas y limitaciones.

Pillajo Lemache (2019) en su trabajo de titulación: "SISTEMA DE FORMACIÓN EN ROBÓTICA INDUSTRIAL BASADA EN CÉLULAS ROBOTIZADAS KUKA Y VISIÓN ARTIFICIAL", determina la viabilidad de Visión Artificial que fue diseñada para fomentar

las habilidades practicas que se le pueden dar a esta herramienta tecnológica, a través de la cámara COGNEX de uso industrial y su implementación aun robot KUKA que cuenta con el controlador KRC4, el cual cuenta con sus propias coordenadas cartesianas.

Muñoz Rodríguez (2011) en su investigación: "DESARROLLO DE APLICACIONES INTEGRANDO ROBÓTICA Y VISIÓN EN UN ROBOT INDUSTRIAL KUKA PARA DEMOSTRAR SUS CAPACIDADES", se comprueba que para trabajar con los robots KUKA no existe una librería que facilite la configuración de dichos robots sobre un lenguaje de programación como lo podría ser "C", siendo de tal manera que el lenguaje de programación KCP del KUKA es el único que puede ser utilizado en su software, además de las dificultades que pueden llegar a presentarse al momento de realizar la comunicación entre el robot y un PC.

2.2. Descripción del problema

Los brazos robóticos KUKA de la Universidad Politécnica Salesiana, permiten conocer a los estudiantes el funcionamiento de control del robot mediante el KCP (kuka control panel), así como el uso de entradas y salidas del controlador KRC2; sin embargo, los robots son utilizados para realizar prácticas únicamente mediante el kcp, esto debido a la carencia de información acerca de la programación KRL, lo que limita el uso de los brazos robóticos del laboratorio. Por tal motivo se propone la implementación de una herramienta inteligente de visión artificial, que permitirá que el autómata KUKA KR5-2 ARC HW identifique, clasifique y dimensione objetos en un espacio de trabajo definido. Esto permitirá la realización de nuevas prácticas de robótica así como nuevos proyectos de grado, generando una nueva utilidad de enseñanza orientada a la visión artificial para el laboratorio de robótica.

2.3. Importancia y alcances

- El implementar la visión artificial al KUKA KR 5-2 ARC HW dentro de la Universidad Politécnica Salesiana, podría dar paso a nuevos temas de titulación que partirían teniendo como base el proyecto de "Implementación de una herramienta tecnológica de identificación y clasificación de objetos para el robot KUKA KR 5-2 ARC HW.
- Actualmente el KUKA KR 5-2 ARC HW en la Universidad Politécnica Salesiana, es únicamente utilizado para realizar prácticas simples de robótica, debido a la antigüedad que tiene el autómata no cuenta con una amplia variedad de aplicaciones dentro de la

institución, por lo cual el proyecto de titulación planteado se enfoca en dar nueva vida al brazo robótico y sus aplicaciones.

- El proyecto de titulación presentado alcanzará el punto donde el KUKA KR 5-2 ARC HW pueda identificar de manera automática figuras con distintas dimensiones y su respectiva clasificación mediante visión artificial.

2.4. Delimitación

El problema de estudio se delimitará en las siguientes dimensiones:

2.4.1. Espacial o geográfica

El proyecto de titulación se realizará en los laboratorios de robótica de la Universidad Politécnica Salesiana, ubicada en Calle Vieja 12 - 30 y Elia Liut, de la ciudad de Cuenca provincia del Azuay.

Figura 1

Mapa de la Universidad Politécnica Salesiana.

2.4.2. Temporal

El proyecto de titulación se comienza en el mes de junio del año 2022 y su finalización será en febrero del año 2023.

2.4.3. Sectorial o institucional

El proyecto de titulación va enfocado al sector industrial, se realizará con el KUKA KR5-2 ARC HW que se encuentra ubicado en los laboratorios de electrónica de la universidad politécnica salesiana.

2.4.4. Problema general

¿Se podrá implementar una herramienta tecnológica para la identificación y clasificación de objetos en el robot KUKA KR5-2 ARC HW?

2.5. Problemas específicos

1.- ¿Es posible identificar y clasificar objetos en un plano cartesiano x - y, a través de una cámara web?

2.- ¿Se podrá establecer la comunicación entre la cámara web y el robot KUKA KR5-2 ARC HW a través de un dispositivo para el envío de datos?

3.- ¿Es posible programar el robot KUKA KR5-2 ARC HW para la recepción de datos y posterior movimiento a través de sus entradas digitales

4.- ¿Es posible implementar la herramienta tecnológica para la identificación y clasificación de objetos, en el espacio de trabajo del robot KUKA KR5-2 ARC HW?

3. Objetivos

3.1. Objetivo general

- Implementar una herramienta tecnológica para la identificación y clasificación de objetos en el robot KUKA KR5-2 ARC HW.

3.2. Objetivos específicos

1.- Identificar y clasificar objetos en un plano cartesiano x - y, a través de una cámara web.

2.- Establecer la comunicación entre la cámara web y el robot KUKA KR5-2 ARC HW a través de un dispositivo para el envío de datos.

3.- Programar el robot KUKA KR5-2 ARC HW para la recepción de datos y posterior movimiento a través de sus entradas digitales.

4.- Implementar la herramienta tecnológica para la identificación y clasificación de objetos, en el espacio de trabajo del robot KUKA KR5-2 ARC HW.

4. Hipótesis

4.1. Hipótesis general

- Se implementará una herramienta tecnológica de identificación y clasificación de objetos en el robot KUKA KR5-2 ARC HW.

4.2. Hipótesis específicas

1.- Se Identificará y clasificará objetos en un plano cartesiano x - y, a través de una cámara web.

2.- Se establecerá la comunicación entre la cámara web y el robot Kuka KR5-2 ARC HW a través de un dispositivo para el envío de datos.

3.- Se programará el robot KUKA KR5-2 ARC HW para la recepción de datos y posterior movimiento a través de sus entradas digitales.

4.- Se implementará la herramienta tecnológica para la identificación y clasificación de objetos, en el espacio de trabajo del robot Kuka KR5-2 ARC HW.

5. Marco teórico

5.1. Métodos de visión artificial

Las TIC (Herramientas Tecnológicas) han sido desarrolladas sin descanso a lo largo de la historia humana, una de estas herramientas tecnológicas es la visión artificial la cual ingenieros y científicos llevan trabajando alrededor de 60 años con ella, de la misma manera buscan encontrar las formas que permitan a las maquinas entender los datos visuales que una cámara les puede aportar, en otras palabras, que tengan la capacidad de ver. Además, la experimentación con esta herramienta tecnológica tuvo sus inicios en 1959, dando con el descubrimiento que el procesamiento de imágenes comienza con formas simples, como pueden ser los bordes rectos (Bernard, 2019).

La visión artificial se basa en el análisis de muchos datos de manera consecutiva hasta encontrar diferencias que permita reconocer imágenes, dando como ejemplo para entrenar un PC que sea capaz de reconocer neumáticos y elementos que sean similares es necesario cargar una gran cantidad de imágenes de estos mismos para que pueda llegar aprende a diferenciar un neumático. De la misma manera, las aplicaciones de la visión por computador se han extendido en gran medida en el ámbito industrial, como pueden ser: contar botellas, clasificar objetos, comprobar defectos de una línea de montaje, etc. Además, que dependiendo de la complejidad que sea diseñada el tipo de visión artificial, permitiría a los robots orientarse en un entorno desconocido basándose en colores y contornos de las figuras. (Eloi, 2012).

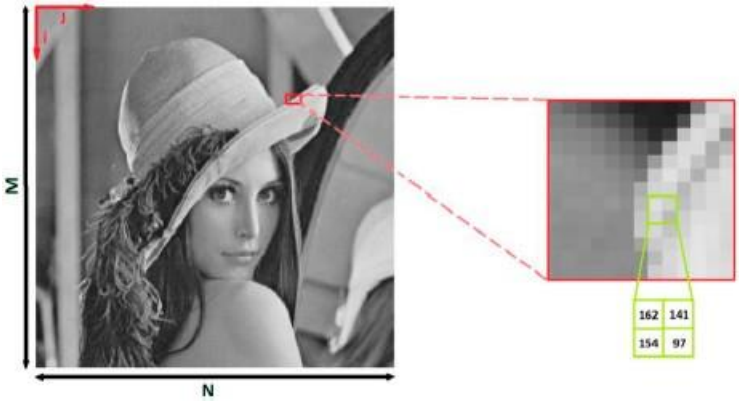
5.1.1. Conceptos teóricos

Concepto de una imagen digital. Una imagen digital está compuesta por píxeles, la cual hace referencia a cada punto en una imagen, suele ser expresada como; ï", como una fila y "j", como una columna haciendo referencia a una matriz MXN que compone a una imagen.

La imagen se puede trabajar dentro de un rango de 8 bits por cada uno de los píxeles representando la intensidad que existe en estos en un intervalo [0, 255]. De tal manera, que el valor de 0 representaría la ausencia de intensidad (píxel negro en la imagen), así mismo cuando este valor aumenta cambia a una escala de grises más claros hasta tener un valor de 255, alcanzando su máxima intensidad (píxel blanco en la imagen) (Jiménez Bravo, 2018).

Figura 2

Rango de intensidad de una imagen digital [2, 255].



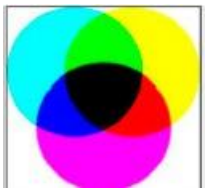
Nota: intensidad 0 (píxel negro), máxima intensidad 255 (píxel blanco) de acuerdo a Jiménez Bravo (2018)

Color La luz blanca del sol está compuesta por radiaciones que a la vez tienen una diversa longitud de onda, a las cuales se les determina un color. Tenemos las ondas cortas (violetas) estas pasan por una desviación superior a las largas (rojas). Estas están clasificadas en dos tipos de modelos; modelo sustractivo y modelo aditivo (Jiménez Bravo, 2018).

Modelo sustractivo: El modelo es llamado de esta manera, debido a que los colores se mezclan entre sí mostrando cada vez menos luminosidad.

Figura 3

Modelo sustractivo.

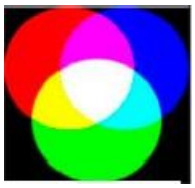


Nota: En este modelo los colores primarios son el cían, magenta y amarillo según Jiménez Bravo (2018)

Modelo aditivo: Son considerados como los colores primarios, las tres longitudes de onda: rojo, verde y azul.

Figura 4

Modelo aditivo.



Nota: el resto de los colores pueden ser obtenido, a partir de la suma de estos de acuerdo con Jiménez Bravo (2018)

5.2. Python

Es un lenguaje de programación que hoy en día es utilizado tanto en el desarrollo de sofware, como en su aplicación en paginas web y manejo de datos sin mencionar el gran uso que se le puede dar por medio de machine learning. Para (Ivet Challenge, 2014) "Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferente". De la misma manera, el lenguaje tiene la capacidad de soportar el uso de paquetes y módulos Lo que le permite crear programas con un diseño orientado a un estilo modular, donde el código pueda ser reutilizado en diferentes tipos de proyectos. Y una vez este tipo de librerías han sido desarrolladas, gracias a la ayuda de su comunidad puede seguir en una constante mejora en su diseño destinándolos al uso en otros proyectos. En consecuente, el importar y exportar este tipo de paquetes y módulos a un intérprete de Python es fácil.

Por otra parte, uno de los beneficios más importantes de Python es que tanto la librería estándar como el intérprete están disponibles gratuitamente, tanto en forma binaria como en forma de fuente. Y no hay excepción aquí, ya que Python y todas las herramientas necesarias están disponibles en las principales plataformas. Por lo tanto, es una solución multiplataforma que atrae a los desarrolladores que no quieren preocuparse por pagar las tarifas de desarrollo (Ivet Challenge, 2014).

5.2.1. Librerías

1. Open CV: Es una biblioteca libre de visión artificial que fue originalmente desarrollada por Intel, la cual ha sido utilizada en una gran variedad de aplicaciones a lo largo de los años desde su creación en el enero de 1999. Es utilizado desde sistemas de seguridad hasta reconocimiento de objetos. Todo esto se debe a la libertad con la que puede ser utilizada, gracias a su publicación bajo la licencia BSD (Jiménez Bravo, 2018).

Figura 5

Logo Opencv.

Nota: Es compatible con el lenguaje de programación Python (Logo OpenCv).

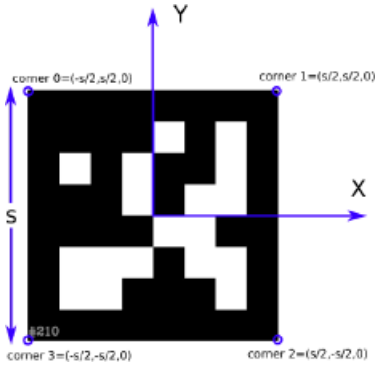
2. Aruco: De acuerdo con (Jiménez Bravo, 2018). Aruco es una la librería de código abierto basada en OpenCV que permite detectar marcadores cuadrados de referencia en imágenes. Además, si la cámara esta calibrada es posible detectar la posición de la cámara respecto al marcador.

3. Marcadores: Según (Jiménez Bravo, 2018): Cada marcador está delimitado por un borde exterior de color negro y una región interior a este que codifica un patrón binario. Este patrón binario es único e identifica cada marcador.

   Así mismo, (Jiménez Bravo, 2018) menciona: "Los marcadores se pueden usar como puntos de referencia 3D para estimación de posición de la cámara. Denotamos como es el tamaño del marcador una vez que está impreso en un trozo de papel".

Figura 6

Ejemplo marcador.



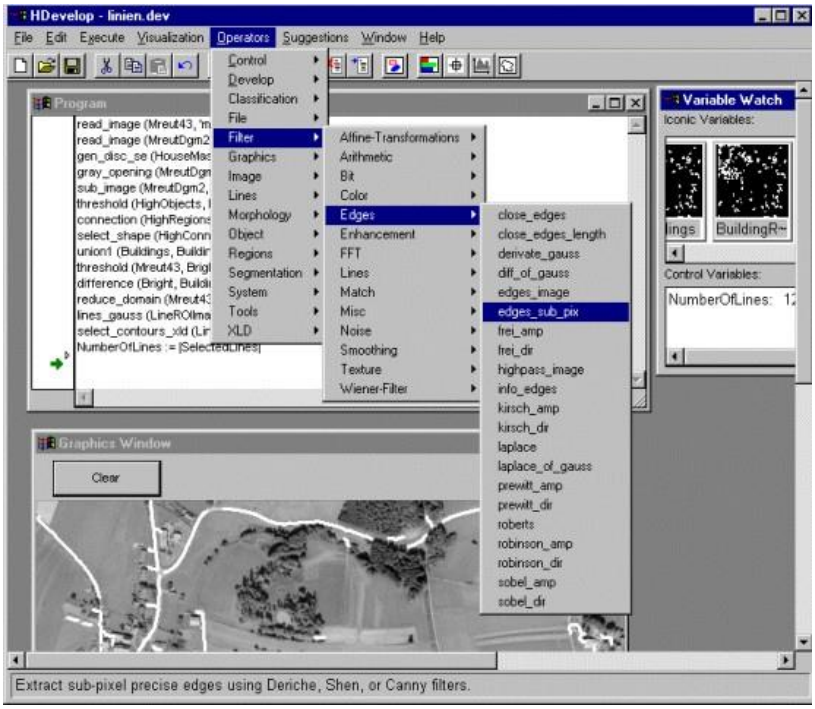
Nota: Existen marcadores estándar y cerrados, el mostrado en la figura es un marcador estándar según Jiménez Bravo (2018).

5.2.2. Otros lenguajes/Programas

1. Halcon para visión artificial Halcon es un software de interfaz gráfica de uso comercial que cuenta con su propio lenguaje de programación, pero con la capacidad de adaptarse a otros lenguajes de programación. Cuenta con alrededor de 800 funciones de procesamiento de imágenes implementadas en código C. Eckstein y Steger (1999)

Figura 7

Estructura software Halcon.



Nota: Opciones y características que dispone el programa de visión artificial Halcón según. Eckstein y Steger (1999).

2. Lenguaje de programación C++: Es considerado como uno de los lenguajes de programación que se encuentra enfocado en objetos, este tiene como base el lenguaje informático "C" siendo considerado "C++" una evolución que surgió a partir de esta herramienta. Por otra parte, aun hoy en día es utilizado para la programación estructurada de alto rendimiento, como por ejemplo su uso en servicios en línea, SO, juegos digitales, etc. (Workana, 2021).

   Además, es considerada dentro del mundo de la programación una de las herramientas mas completas y caprichosas para los usuarios que usen este lenguaje, de la misma manera, un programador con experiencia centrado en C++ cuenta con uno de los perfiles profesionales mas demandados dentro de la industria, siendo capaz de participar en todo tipo de proyectos tecnológicos. De acuerdo con (Workana, 2021): (Es importante

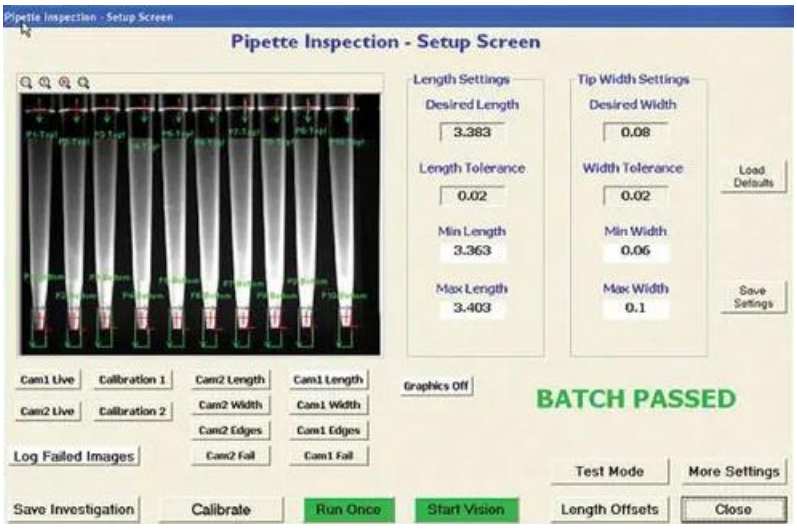
mencionar que no todos los proyectos de TI deben desarrollarse en lenguaje C++. Muchas veces, un lenguaje más visual, intuitivo o especializado es una opción igualmente eficiente y sencilla. C++ actualmente se recomienda para proyectos muy específicos y de alta complejidad, por ejemplo, programas de criptomonedas y videojuegos de realidad aumentada).

3. Sherlock para visión artificial

   Es un software con interfaz gráfica que no requiere de muchos conocimientos de visión artificial. Se basa en definir zonas de interés dentro de una imagen para posteriormente aplicar funciones en dicha zona. Su uso es comercial y cuenta con varias funciones preestablecidas a las cuáles se les puede variar sus parámetros. Vigneswaran, Madhu, y Rajamani (2012)

Figura 8

Software Sherlock.

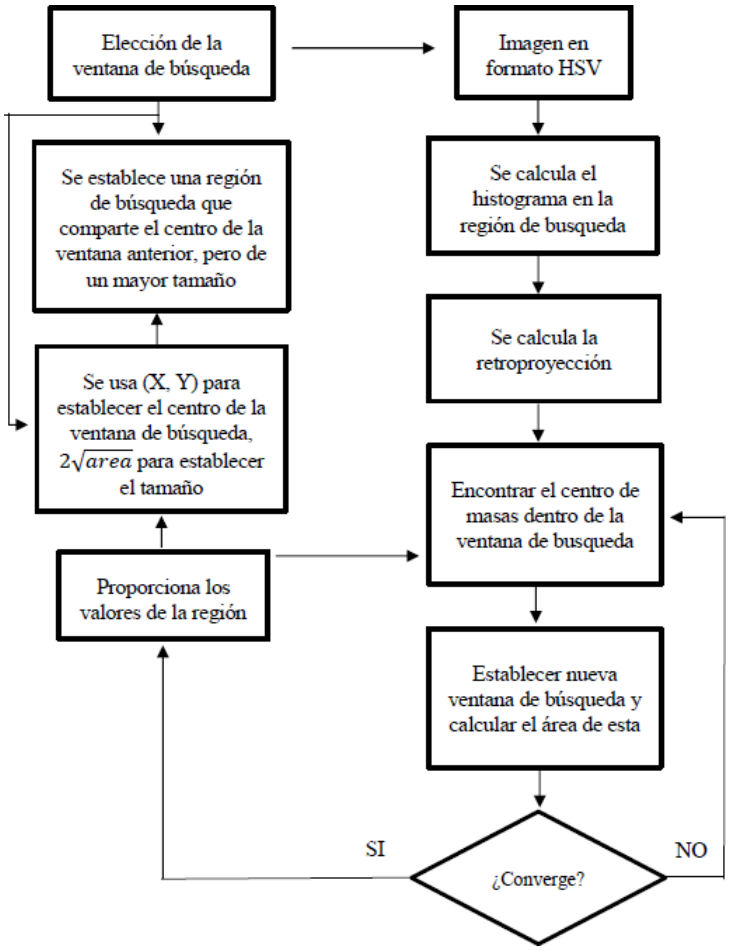


Nota: Entorno gráfico donde se pude apreciar las diferentes opciones y características del software Sherlock Vigneswaran y cols. (2012).

4. CAMshift: CAMShift (Continously Adaptive Mean Shift) es el algoritmo que ha sido creado para el seguimiento de un objetivo en específico. Este puede detectar cambios en el tamaño y posición de los objetos al momento de trasladarse. El algoritmo utiliza imágenes que son basadas del modelo RGB al modelo HSV, debido al uso de los colores.

Figura 9

Diagrama de bloques del algoritmo CAMShift.



Nota: En el diagrama de bloques se presentan los pasos a seguir para el uso del algoritmo CAMShift de acuerdo con Jiménez Bravo (2018).

5.2.3. ¿Porque Python?

Se escogió Python porque dispone con una de las mayores facilidades al hablar de la programación enfocada a objetos que gracias a la forma de su lenguaje es practica e imprescindible en este ámbito, de esta manera, siendo considerado un lenguaje multiparadigmas. Por otra parte, este lenguaje está basado en el "ABC" de los lenguajes de programación, además de estar influenciado en otros tipos de programación como podría ser "C". Además

de la versatilidad que el lenguaje de programación puede llegar a brindar en comparación a otros como seria java, esto significa que su estructura de datos fue diseñada para revolver cualquier tipo de problema permitiendo el acceso a una gran variedad de herramientas que incluso su comunidad puede llegar a ofrecer, de la misma manera, gracias a su versatilidad se puede llegar aprender Python en tan solo 6-8 semanas. Y una de las razones principales por las que se eligió este lenguaje de programación es por su capacidad de automatizar de tareas y procesos que permiten la creación de programas complejos con mayor facilidad, debido a la capacidad del lenguaje para escribir scripts de diferentes sistemas, creando programas sencillos para la automatización de tareas que disminuyen nuestra productividad.

5.3. Robot KUKA

5.3.1. Tipos

1. KUKA KR 5-2 ARC HW

   Según (Cabrera Sarmiento y Martinez Aviles, 2022) mencionan: (Los brazos antropomórficos de la casa fabricante KUKA tienen seis grados de libertad. permitiendo mover cada articulación. Es importante especificar las características físicas y técnicas de este tipo de manipulador robótico. Este robot antropomórfico, está formado por eslabones unidos por medio de articulaciones rotacionales. El número de articulaciones está basado en el hecho que es necesario un número mínimo de grados de libertad para posicionar y orientar un cuerpo de cualquier manera en el espacio, tres que definen la posición y al menos otros dos para la orientación, dependiendo de la resolución que se desee el robot).

Figura 10

KUKA KR 5-2 ARC HW que se encuentra en el laboratorio de la Universidad Politécnica Salesiana.

Nota: Los grados de libertad están distribuidos de manera que el robot solvente el posicionamiento espacial en cualquier punto de su área de trabajo de acuerdo con Cabrera Sarmiento y Martinez Aviles (2022).

2. KUKA.VisionTech

KUKA.VisionTech es una herramienta creada directamente por la empresa KUKA, el producto ofrece potentes mecanismos que pueden reconocer objetos en dos dimensiones, esto gracias a su cámara de alta calidad que se encuentra protegida por una carcasa IP 61 como se puede observar en la Figura 11, que esta herramienta permita el renacimiento de objetos da paso al uso flexible a la línea de sus robots antropomórficos en los que se puedan incorporar este instrumento. De la misma forma, esta herramienta ofrece el reconocimiento de códigos que puede llegar a facilitar la trayectoria con el que el brazo puede llegar a moverse y de cierta manera asegurar la calidad de sus controles automáticos, permitiéndole reducir costes. Por otra parte, los requisitos para utilizar esta herramienta de visión artificial en los KUKA (KUKA.AG, 2019):

Figura 11

KUKA.VisionTech.

Nota: La cámara de alta calidad permite el reconocimiento de objetos incluso en entornos no estructurados de acuerdo con KUKA.AG (2019).

5.3.2. Elementos

1. Control de ejecución de un programa

   La programación en un robot le permite seguir una serie de instrucciones u operaciones que se le designe tanto de una manera secuencial como paralela. Con la ayuda de las sentencias y bucles se llega a crear un control de flujo de ejecución para el brazo robótico, permitiendo al programador tener un control total tanto del movimiento del robot como de las entradas y salidas del mismo. Así mismo las interrupciones son otro tipo de sentencia, las cuales son utilizadas para priorizar una instrucción de código, útiles al momento de tener señales externas como sensores, pulsantes, etc. Guaraca Medina y Ochoa Ochoa (2015)

2. Hardware del robot KUKA KR5-2 ARC HW

   Está constituido por 4 subsistemas. El subsistema mecánico está conformado por un brazo antropomórfico de 6 ejes. Con una capacidad de carga de 5Kg, cada eje cuenta con topes mecánicos y de software que delimitan el movimiento para cada uno con el fin de evitar colisiones Guaraca Medina y Ochoa Ochoa (2015)

Tabla 1

Rango de trabajo de las articulaciones del manipulador KR5-2 ARC HW.

Ejes	Desplazamiento en grados	Velocidad con carga nominal 5 Kg
Eje 1 (A1)	+155/-150	154 º/s
Eje 2 (A2)	+65-/180	154 º/s
Eje 3 (A3)	+170/-110	228 º/s
Eje 4 (A4)	±165	343 º/s
Eje 5 (A5)	±140	384 º/s
Eje 6 (A6)	±350	721 º/s

Nota: En la tabla se observa los grados de libertad de cada uno de los ejes del robot KUKA KR5-2 ARC HW, así como la velocidad que soporta con carga nominal de 5kg de acuerdo con Guaraca Medina y Ochoa Ochoa (2015).

De la misma manera, también se cuenta con el subsistema de programación que se encuentra integrado en la unidad de programación (UMP) y su respectivo lenguaje de programación. Este dispositivo permite la comunicación entre el operador y el sistema robótico. Se encuentra conformado por varios teclados, botones de accionamiento, una pantalla gráfica y un ratón con seis grados de libertad (Guaraca Medina y Ochoa Ochoa, 2015).

Figura 12

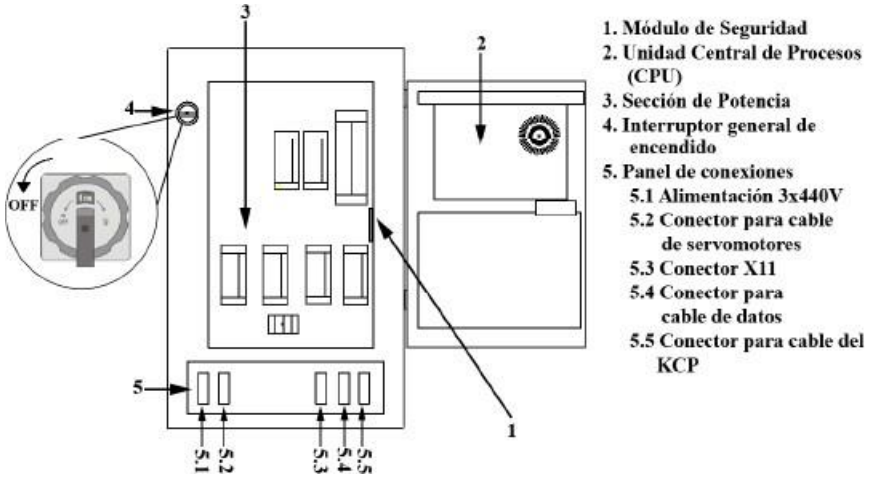
Panel de control (KCP) robot KUKA.

Nota: El UMP mostrado en la figura es el KUKA Control Panel (KCP) del fabricante KUKA Jiménez Bravo (2018).

Por último, tenemos al subsistema de control o también conocido como unidad de control del robot, el cual se encarga del control general del KUKA. Esta diseñado en la forma de un armario en donde convergen las respectivas instalaciones de alimentación, controladores de movimiento y el sistema de seguridad del autómata. Se encuentra conformado por la unidad de procesos, el módulo de seguridad, la sección de potencia el panel de conexiones Guaraca Medina y Ochoa Ochoa (2015).

Figura 13

Esquema subsistema de control.



Nota: el esquema presentado pertenece al KUKA KR 5-2 ARC HW de acuerdo con Jiménez Bravo (2018).

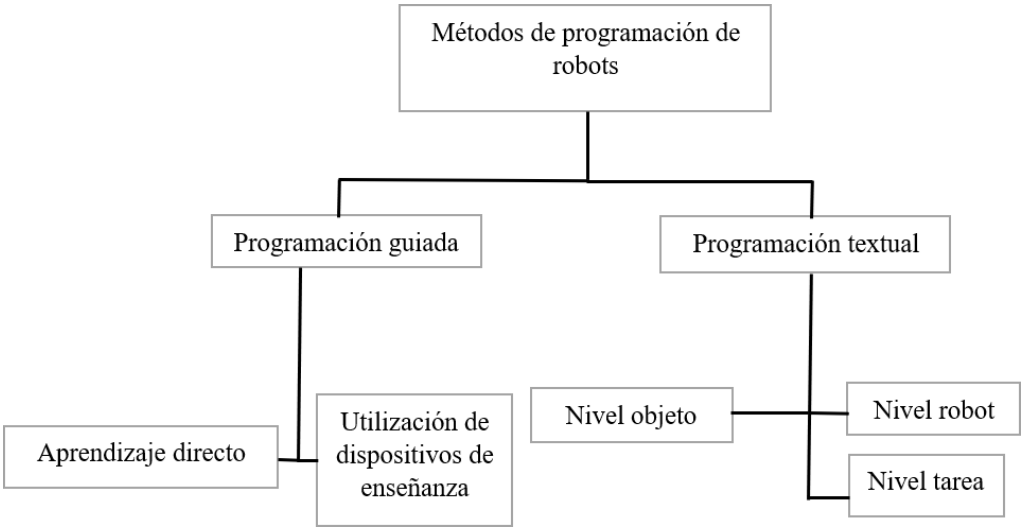
5.3.3. Programación

Para utilizar robots en la industria se requiere de las respectivas indicaciones que el autómata pueda interpretarlas y ejecutarlas. Inclusive, los fabricantes utilizan lenguajes de programación para sus robots industriales. Estos facilitan la programación del usuario y permite una interpretación y ejecución rápida con las tareas introducidas al androide. Guaraca Medina y Ochoa Ochoa (2015).

Uno de los métodos de programación; son los pasos necesarios que se deben seguir para enseñar a un sistema robótico acciones que este tiene que realizar. Además, existen dos métodos de programación, los cuales se encuentran separados en programación guiada y textual. De la misma manera, se pueden encontrar robots los cuales puedan tener incorporados estos dos tipos de programación, teniendo así una capacidad de realizar acciones mucho más complejas.

Figura 14

Métodos de programación de robots.



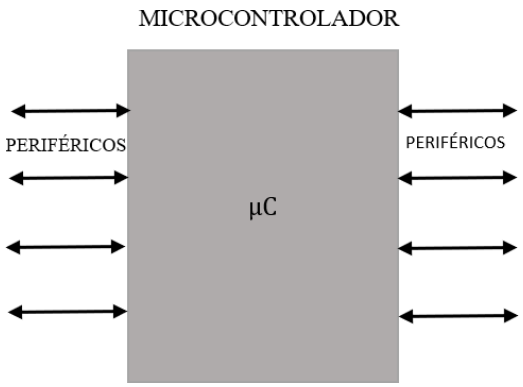
Nota: Dentro de las categorías mostradas, existen grupos con características propias.

5.4. Microcontrolador

Un microcontrolador es considerado un dispositivo que cuenta con la capacidad de realizar procesos lógicos a través de un lenguaje programación que permita ejecutar las acciones que el usuario introduzca dentro de este, contando únicamente con un circuito integrado que en su interior contiene los componentes electrónicos necesarios que son necesarios para el funcionamiento de un microprocesador; de tal manera el dispositivo contiene en únicamente un solo integrado su unidad de proceso, memoria RAM, puertos de entradas, salidas y diversos tipos de periféricos como se puede observar en la Figura 15, pero con el inconveniente que cuenta con un espacio reducido. Se puede decir que en definitiva este dispositivo electrónico cuenta con todos los componentes de un computador (Aguayo, 2004).

Figura 15

Microcontrolador.



Nota: El microcontrolador es un sistema cerrado. Todas las partes del procesador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

5.4.1. Familia de los PIC

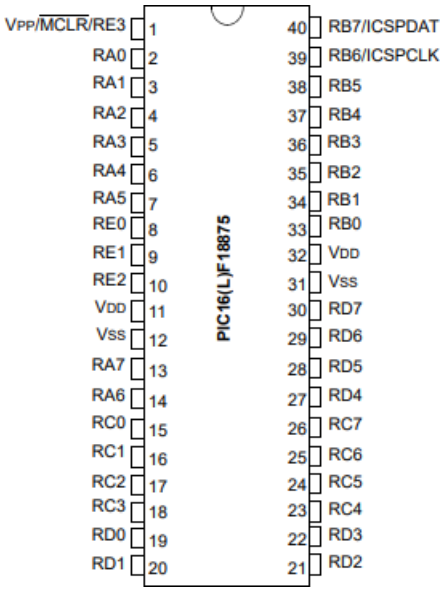
La familia de los microcontroladores PIC han sido capaces de revolucionar el mundo de las aplicaciones electrónicas. Siendo preferidos por los desarrolladores y programadores, debido a la facilidad que ofrece al momento de la integración y programación. De la misma manera, hay que mencionar que esta familia de microprocesadores pertenece a la casa de Microchip, la cual los tiene divididos en 4 gammas: gamma baja, media y alta. De esta manera, las diferencia que presentan en cada una de ellas consiste en el número de instrucciones que el microprocesador puede ejecutar, además en el número de puertos y funciones que cada una de las gammas puede llegar a ofrecer (Aguayo, 2004).

5.4.2. ¿Por qué se eligió al microcontrolador PIC 16f18875?

Según (Microchip, 2019): (Los microcontroladores PIC16(L)F1885X/7X cuentan con periféricos analógicos independientes del núcleo y periféricos de comunicación, combinados con eXtreme Low Power (XLP) para una amplia gama de aplicaciones de uso general y de bajo consumo. La familia cuenta con CRC/SCAN, HLT y Windowed WDT para ayudar a los clientes que buscan agregar seguridad a su aplicación. Además, esta familia incluye hasta 56 KB de memoria Flash, junto con un ADC de 10 bits con Computación (ADC2) para el análisis de señales automatizado para reducir la complejidad de la aplicación.)

Figura 16

PIC 16f18875.



Nota: Esquema microcontrolador 16f18875 de acuerdo con Microchip (2019)

El microcontrolador 16f18875 es un microcontrolador de bajo costo que cuenta con 21 pines como se puede observar en la Figura 16, lo cual lo convierte en una opción viable gracias a las características que ofrece por un precio moderado, de esta manera, el controlador lleva consigo un oscilador interno ajustable hasta 32MHz gracias a esto se permite hacer conexiones más sencillas obviando el conectar otro oscilador de manera física al microcontrolador. Por otra parte, presenta la característica de tener un núcleo mejorado con 49 instrucciones y a parte tener 16 niveles de pilas, permitiendo tener una mayor velocidad al momento de enviar, recibir y procesar datos (Microchip, 2019).

6. Marco metodológico

En este apartado se presentará el marco metodológico que es utilizado en el proyecto tecnológico presentado. Aquí se plantea los distintos pasos o actividades de forma enumerada que fueron realizadas para alcanzar los resultados esperados del proyecto, además de explicar cada uno de estos puntos de manera detallada. Así mismo la presentación de todas las

herramientas y métodos que han sido utilizados para el desarrollo del proyecto, por otra parte, se presentara el uso y funcionamiento completo del sistema de visión artificial que fue implementado en el robot KUKA KR5-2 ARC HW.

6.1. Visión artificial

6.1.1. Obtención de un sistema coordenado aplicado a la cámara web.

La herramienta tecnología de visión artificial desarrollada por medio de la librería de Open Cv da la posibilidad de detectar la distancia que existe entre diferentes tipos de objetos teniendo como referencia algún otro el cual tenga dimensiones fijas cuales pueden ser utilizadas como base al momento de detectar el espacio que existe entre los objetos. De la misma forma, este instrumento inteligente una vez haya detectado el espacio entre una referencia y los objetos, nos entregará en tiempo real la distancia que exista entre estos en un sistema coordenado (x-y), además la herramienta tecnológica permitirá delimitar el espacio de trabajo que se va a encontrar monitorizada por una cámara que es la encargada de enviar los datos al sistema de visión por computador.

La cámara que se utilizó en el proyecto se trata de una cámara web que cuenta con una resolución de (1920x1080) y 60 cuadros por segundo sin ninguna clase de zoom, pero se trabajó a una resolución de (1280x270) debido a, que se vio pertinente el usar la calidad de imagen que la cámara brindaba a esa resolución, en comparación a la calidad inferior que se apreciaba a una resolución mayor.

1. Acople de la cámara web en un punto fijo en el espacio de trabajo

   Se necesita mantener a la cámara Web en una posición lo más estable y centrada posible con respecto al área de trabajo del KUKA KR5-2 ARC HW, para tener una imagen lo suficientemente clara y estable que sea capaz de reconocer sin problemas los objetos colocados sobre la mesa y la referencia, por otra parte, el programa está diseñado de tal manera que si la referencia se cambie de ubicación el sistema de visión artificial sea capaz de distinguirla y tenerlo como primer objeto al cual se va a basar para recabar la información de sus dimensiones . Además, el mantener la cámara en un buen espacio con el ángulo perfecto grabando al espacio de trabajo, nos permitirá tener medidas más precisas.

Figura 17

Acople de Cámara Web en el laboratorio de robótica.

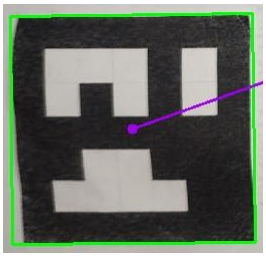
Nota: Colocación de la cámara web en un trípode con dirección visual hacia la zona de trabajo.

2. Localización de la referencia en el plano coordenado X-Y.

   Como se puede observar en la Figura 18, la referencia se trata de una figura en forma cuadriculada de blanco y negro, que cuenta con unas dimensiones de 5cm x 5cm y son reconocidas por medio de la inteligencia artificial, de tal modo, que detecta sus bordes aplicando cierto número de filtros a la imagen que la visión por computador recibe por medio de la cámara web, de esta manera, detectando la referencia que se encuentra programada de modo que siempre será detectada al lado derecho del plano. De la misma manera, el programa utiliza los datos de la referencia para calcular el espacio que existe entre esta y las figuras colocadas en el espacio de trabajo, que por medio de una relación existente entre píxeles y centímetros se obtienen los datos y se presentan en tiempo real en la pantalla del computador, teniendo la capacidad de variar las posiciones de los objetos y la herramienta detectara sin problemas sus posiciones, siempre y cuando se encuentren dentro del espacio delimitado por la cámara web.

Figura 18

Referencia de 5x5cm.



Nota: Reconocimiento del objeto de referencia y sus medidas para determinar las distancias con los objetos.

3. Localización de los objetos en el plano coordenado X-Y.

   La identificación de los objetos que se encuentran sobre el área de trabajo, es imprescindible para conocer su localización en el plano coordenado X-Y y se puede lograr, utilizando distintos tipos de filtros sobre la imagen que transmite la cámara web, que permite que los contornos de los objetos puedan ser de una manera más sencilla detectables por el programa mediante de la librería .imutils que se encarga de contornear los objetos, de la misma manera este datos de contorno se almacenara dentro de una variable llamada .box, y con el uso de la librería .cv2 se procede a dibujar los contornos que fueron identificados alrededor de los objetos de manera consecutiva, cabe mencionar que los contornos de las figuras serán clasificados de tal manera que se mostraran en pantalla de izquierda a derecha

Figura 19

Localización de objetos dentro del plano coordenado X-Y

Nota: Reconocimiento de un objeto dentro del área de trabajo.

4. Reconocimiento de las distancias en cm en un plano coordenado X-Y con referencia.

   Como se puede observar en la Figura 20 la visión por computador detecta al objeto más a la izquierda como su punto de referencia, de esta manera, se estableció a la figura cuadriculada como la misma teniendo en cuenta sus dimensiones para posteriormente se pueda calcular las dimensiones que separa a la referencia con los respecto a las demos objetos que se encuentran en el área de trabajo. El programa funciona de tal manera, que guarda los puntos coordenados de la referencia en refCoords, de la misma manera, almacena los puntos en el plano de los objetos como objCoords.

   - refCoords: Coordenada del objeto de referencia
   - objCoords: Coordenadas de los objetos que se encuentran repartidos en el área de trabajo

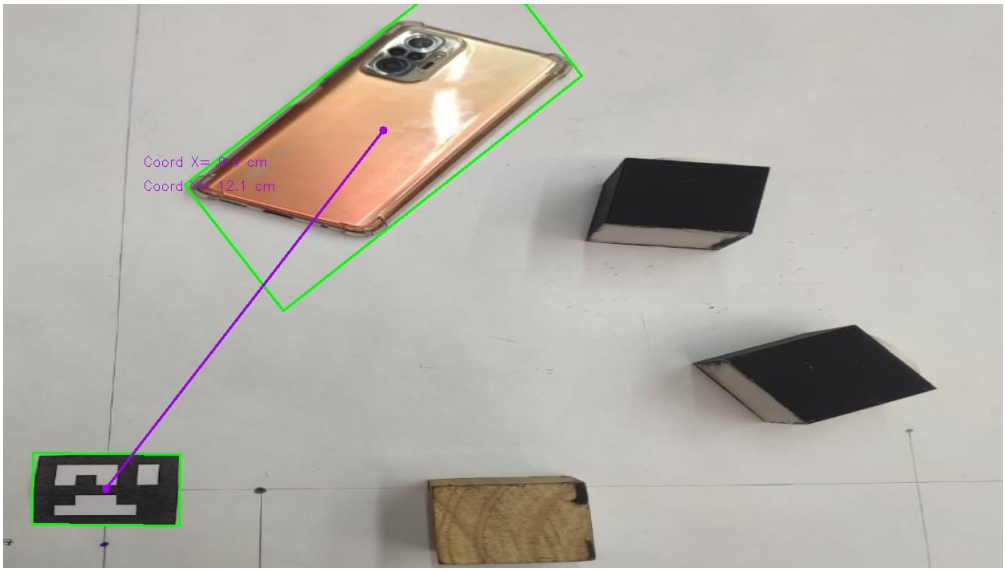
   En consiguiente, los datos del objeto de referencia y los objetos que se encuentran sobre el área de trabajo entran en un bucle donde se procede a encontrar el centro de las figuras y se guardan dentro de la variable D2, posteriormente se dibuja las líneas que una el centro de la referencia con el centro de todos los objetos, de la misma manera, en este apartado se triangula la posición de los objetos en la coordenadas x-y, de esta forma pasan a ser almacenadas dentro de las variables D201 para la distancia en Y que existe entre el objeto de referencia y los objetos, de la misma forma, se procede hacer

los mismo con la variable D2A1 para almacenar los datos en centímetros entre referencia y los objetos en las coordenadas en X.

- D2: Centro de los objetos
- D2O1: Coordenadas en Y de los objetos
- D2O1: Coordenadas en X de los objetos

Figura 20

Visualización de las coordenadas entre los objetos sobre espacio de trabajo.



Nota: Coordenadas en X-Y de los objetos que se encuentra sobre el espacio de trabajo.

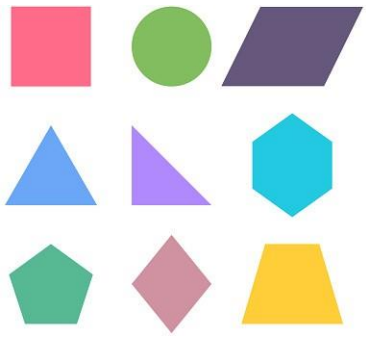
6.1.2. Creación de un programa para obtener las dimensiones de objetos base y altura a través de una cámara web.

Se utilizó la librería aruco con el fin de obtener una referencia para la detección de los parámetros de los objetos (base, altura), además esta librería permite calibrar la cámara para que independientemente de la altura en la que se encuentre la cámara referente al objeto, se devuelva el mismo valor en centímetros de los objetos, puesto que se utiliza un marcador referencia de dimensiones 5x5, no obstante existe una diferencia entre 2 y 5 milímetros debido a que se detecta los contornos de los objetos. A continuación, se presentan los pasos realizados:

1. Instanciar la clase detector junto con sus atributos: Para la detección de los objetos es necesario una conversión a escala de grises con el fin de que la cámara web detecte los mismos sobre un fondo sólido, para esto se crea una máscara que separa los objetos del fondo y adicionalmente una condición de área mínima para despreciar objetos de un tamaño no deseado. Una vez realizado esto se procede a obtener las coordenadas o valores de los objetos mediante la función cv2.minAreaRect, obteniendo cinco parámetros en píxeles por cada objeto detectado (posición x, posición y, base, altura, ángulo).

Figura 21

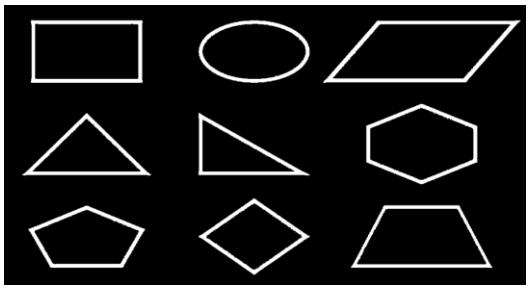
Imagen original utilizada para prueba de contornos.



Nota: Se representa diferentes tipos de figuras geométricas.

Figura 22

Resultado de aplicar la máscara a la imagen original.

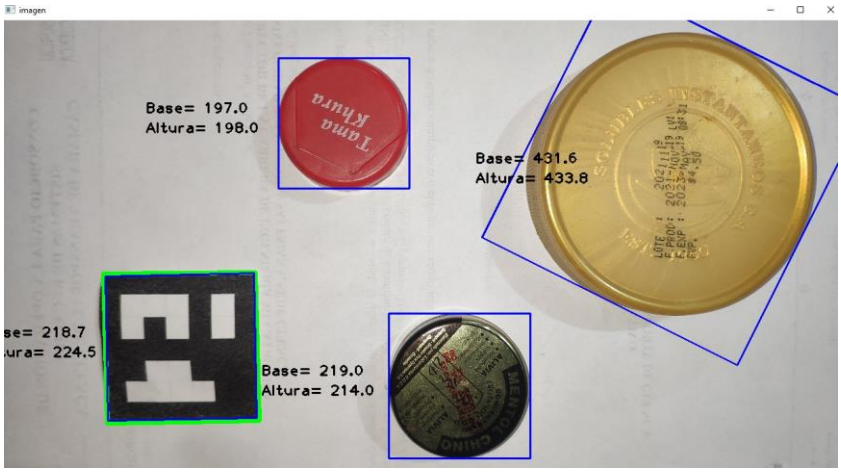


Nota: Se aprecia como el programa detecta correctamente el contorno de las figuras.

2. Uso de la librería aruco para calibración: Una vez obtenidos los parámetros de cada objeto, se procede a utilizar el marcador aruco de 5x5cm para la calibración del programa, con el fin de obtener dimensiones en valor de cm de los diferentes objetos. Primeramente se instancia las funciones aruco.DetectorParameters, la cual nos guardará los parámetros del marcador aruco, seguidamente se instancia el marcador aruco con la función cv2.aruco.Dictionaryget(), para que la cámara web pueda reconocer y funcionar solo cuando el marcador aruco especificado se encuentre dentro de la imagen. Finalmente coloca en pantalla textos que indiquen el valor en píxeles de los objetos reconocidos.

Figura 23

Visualización de las dimensiones en pantalla del marcador aruco y objetos aleatorios.

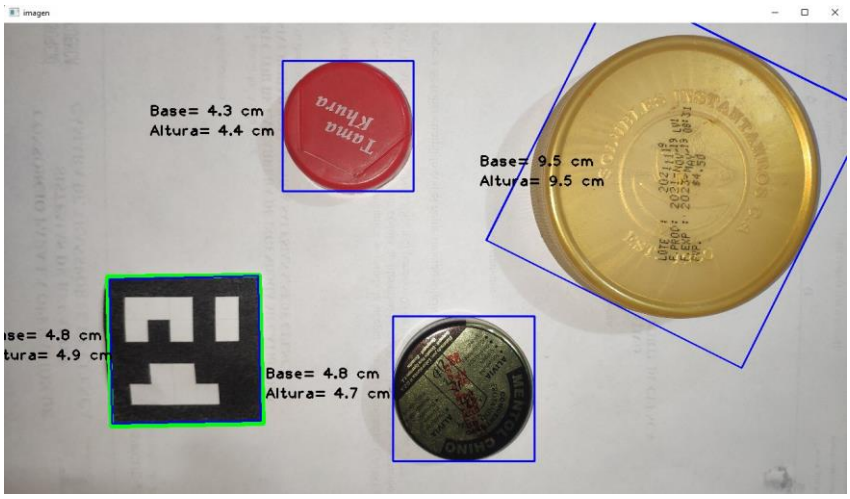


Nota: Información en píxeles de base y altura para los diferentes objetos en pantalla.

3. Instanciar las dimensiones de los objetos en cm: Obtenidos los valores en píxeles del marcador aruco, estos se colocan en un arreglo para obtener una matriz que se encuentra alrededor del marcador aruco, con estos valores se utiliza la función cv2.arcLenght para obtener el valor de perímetro de dicha matriz. Conociendo que las medidas del marcador aruco son de 5x5cm, su perímetro será de 20cm por lo que relacionando el perímetro en píxeles como en centímetros, se obtiene una relación para transformar los valores de píxeles a centímetros.

Figura 24

Visualización de las dimensiones en centímetros del marcador aruco y objetos aleatorios.



Nota: Información en centímetros de base y altura para los diferentes objetos en pantalla.

6.1.3. Elaboración de un programa que permita clasificar objetos de acuerdo con sus dimensiones.

Para la programación de clasificación se optó por seguir un orden de derecha a izquierda, esto con la finalidad de que los objetos no se lleguen a colisionar entre sí al momento de la clasificación, se pretende clasificar los objetos de mayor a menor área en tres secciones distintas. Se realizaron los siguientes pasos para que el programa clasifique los objetos:

1. Cálculo:

   Dado que ya se conoce las dimensiones de base y altura en cm de los diferentes objetos y el contorno de los mismos es rectangular se procede a calcular el área de cada uno de estos.

2. Redondeo:

   Se utiliza el método round para redondear los valores de las áreas en un solo decimal con el fin de que el envío por comunicación serial sea óptimo.

3. Creación de una lista:

Seguidamente se crea una lista en la que se almacena los valores de las diferentes áreas con el método append,finalmente los datos que se almacenan en la lista van en el siguiente orden:

- Elemento 0: Marcador Aruco
- Elemento 1: Objeto 1
- Elemento 2: Objeto 3
- Elemento 3: Objeto 2

6.1.4. Creación de un programa para el envío de información a través de conexión serial.

Para lograr enviar los datos tanto de áreas como de posicionamiento en el plano x-y hacia un microcontrolador y posteriormente al robot KUKA KR5-2 ARC HW se realizaron los siguientes pasos:

1. Obtención de las variables como salida:

   Obtenidos los programas de identificación y clasificación, se obtienen las siguientes variables de salida anteriormente mencionadas:

   - Detecciones: Lista que contiene como valores las diferentes áreas de los objetos analizados.
   - Objeto1x, Objeto2x, Objeto3x: Contiene las dimensiones verticales en centímetros de los objetos capturados por la cámara web.
   - Objeto1y, Objeto2y, Objeto3y: Contiene las dimensiones horizontales en centímetros de los objetos capturados por la cámara web

Tabla 2

Variables de salida.

Área 1	[76.2]	x 23. 6	y 44.7
Área 2	[16.7]	x 22.0	y 17.1
Área 3	[40.0]	x 22.4	y 29.5

Nota: Se observa las variables de salida de identificación y clasificación.

2. Envío de las variables al microcontrolador:

   Mediante Python se enviarán los variables que representan las distancias de los objetos con respecto a la referencia, las variables se encuentran clasificadas desde el número 1 hasta el número 255, mismas que corresponden a las 255 posiciones impuestas, es decir con las que cuenta el área de trabajo del KUKA KR5-2 ARC HW. Las variables son enviadas al microcontrolador a través de la librería pyserial, que se ejecuta en python.

3. Instanciar pyserial:

   Se utilizó un USB-TTL para realizar la comunicación serial entre la computadora y el microcontrolador, se definió como parámetros dentro de la programación el puerto de entrada en el que se encuentra el USB-TTL y una tasa de baudios de 9600, la cual representa la velocidad de transmisión.

4. Uso de pyserial:

   Se utilizó el método .write para realizar el envío de datos hacia el microcontrolador, es importante adicionar que se utilizó el atributo (encode) para enviar la información en formato string de manera codificada para su correcta recepción en el microcontrolador, en este caso se utilizó codificación ASCII.

6.2. Programación KRL.

KRL es el acrónimo para el lenguaje de programación del KUKA versión KRC2, mediante programación en Python se creó todo el código de programación tanto para las posiciones del espacio de trabajo como la activación de las entradas y salidas digitales del robot KUKA KR5-2 ARC HW.

6.2.1. Dimensionamiento del espacio de trabajo.

Para lograr mapear diferentes puntos en el espacio de trabajo del robot KUKA KR5-2 ARC HW, se utilizó una programación en Python para generar los diferentes puntos de posición en lenguaje KRL (KUKA ROBOT LANGUAGE). A continuación, se detallan los pasos que se realizaron:

1. Programar manualmente puntos de referencia:

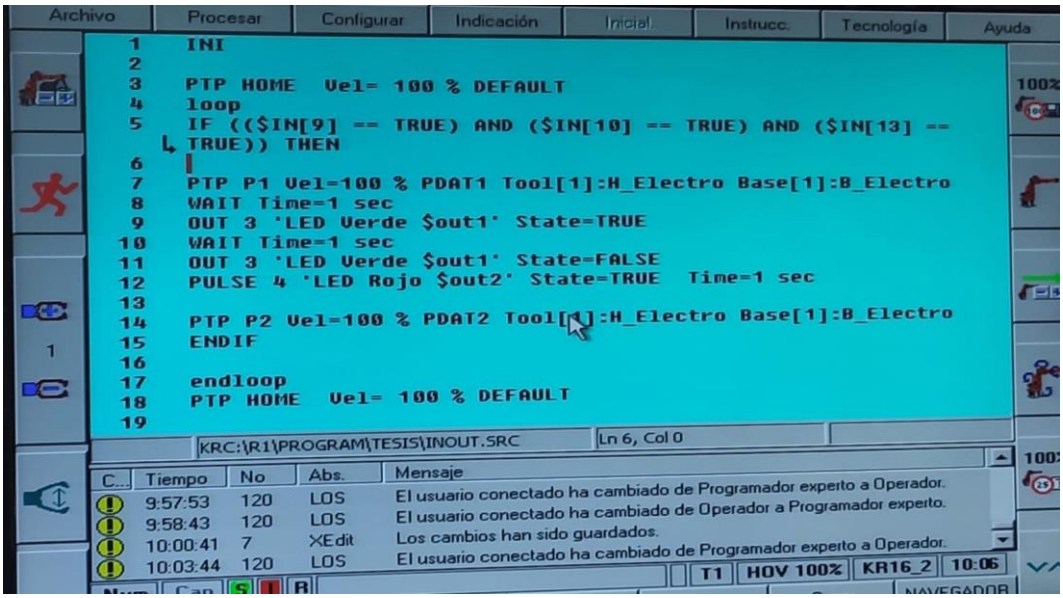
   Se comenzó por programar directamente con el KCP (kuka Control panel) desde la posición de reposo del robot hasta una posición en la cual el eje Z se encuentre a una

distancia deseada de los objetos, seguidamente se procedió a guardar en el programa KRL las cuatro esquinas de la mesa de trabajo para obtener los puntos de referencia en donde actuará el robot KUKA KR5-2 ARC HW.

- Punto 1: (x 964.3 ; y 347.08)
- Punto 2: (x 964.3 ; y -459.8)
- Punto 3: (x 1453.8 ; y -459.8)
- Punto 4: (x 1453.8 ; y 347.08)

Figura 25

Ejemplo de programación KRL mediante el KCP.



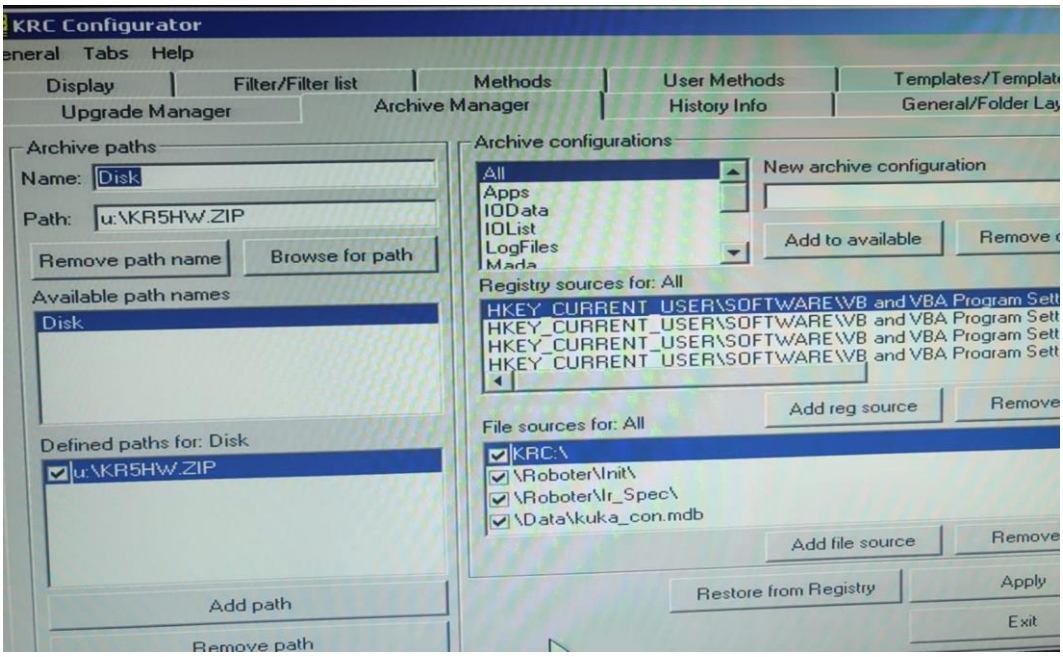
Nota: Entorno de programación KRC2 del robot KUKA KR5-2 ARC HW.

2. Parchear Unidad USB:

   Una vez obtenidos los puntos de referencia se añade un parche a una unidad de almacenamiento extraíble USB para poder traspasar archivos desde el robot KUKA KR5-2 ARC HW hacia un ordenador, este parche se realiza mediante un ejecutable encontrado dentro de la carpeta de programación del robot, llamado KrcConfigurator.exe.

Figura 26

Interfaz KRC configurator.



Nota: Permite configurar el administrador de archivos.

3. Transferencia de archivos KRL hacia el ordenador:

   Agregado el parche a la unidad de almacenamiento extraíble USB, para la transferencia de archivos se debe activar el modo experto del robot KUKA KR5-2 ARC HW. Finalmente se selecciona los archivos que se desea pasar a la unidad de almacenamiento extraíble USB desde la misma interfaz del robot KUKA KR5-2 ARC HW. Se debe aclarar que existen dos archivos los cuales permiten la programación en KRL.

   - Archivo src: Es el archivo principal el cual contiene información de: (posiciones, secuencias, condiciones, velocidades, etc.).
   - Archivo dat: Archivo secundario que contiene información de la posición de los diferentes puntos programados tales como: (posición x, posición y, posición z, giro A, giro B, giro C, etc).

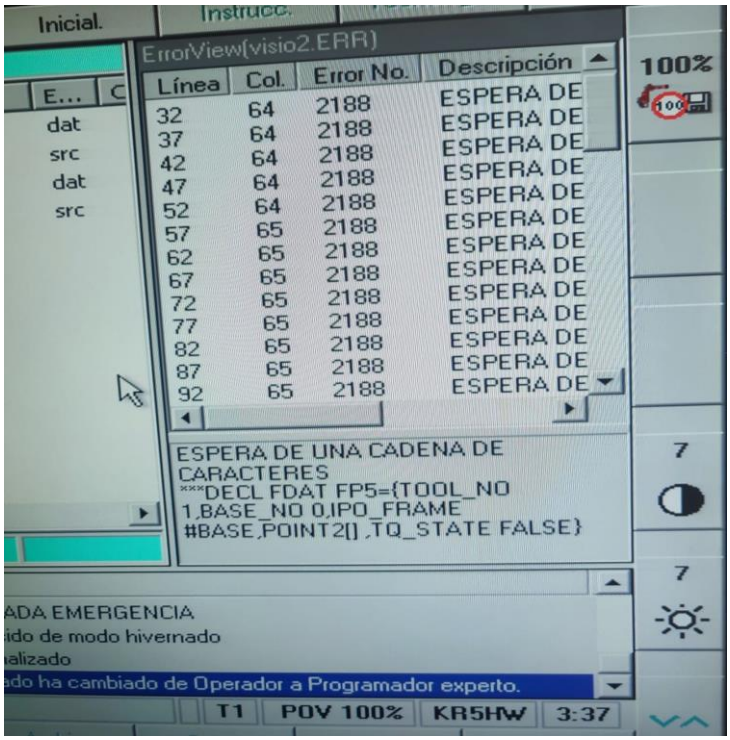
6.2.2. Programación en Python para generar el código del espacio de trabajo para el robot KUKA KR5-2 ARC HW.

Obtenidos los puntos de referencia para el espacio de trabajo, se procedió a transferir los archivos correspondientes a la programación KRL del robot KUKA hacia el computador mediante una memoria USB. Para obtener la matriz de 255 puntos se utilizaron los extremos de las posiciones y se dividió en puntos con separaciones tanto horizontales y verticales de 16 puntos restando una distancia de 32.61875 para el eje x vertical y sumando una distancia de 53.79375 para el eje y horizontal. Realizando un doble bucle for para el eje vertical y horizontal. Las líneas de código KRL se obtuvieron realizando una programación base que genere todo el formato KRL desde Python, esto con el atributo .write. Obteniendo un total de 255 puntos totales para el espacio de trabajo, esto debido a que el punto 1 no se considera debido a que en ese punto se encuentra el marcador de referencia aruco. Finalmente, Python generó los archivos src y dat. Cabe mencionar que se realizó una programación variable con el fin de incluir un mayor número de puntos en la matriz resultante de la programación KRL.

Finalmente se procede a añadir la matriz de puntos deseada a los archivos src y dat respectivamente para almacenarlos en la unidad de almacenamiento extraíble USB. Al momento de traspasar los archivos al robot KUKA KR5-2 ARC HW, el mismo robot compila la información y dará un mensaje de error en caso de que el programa KRL cuente con programación errónea. En caso de que no existan errores se puede ejecutar el código KRL y verificar el movimiento del robot KUKA KR5-2 ARC HW hacia las posiciones impuestas en python.

Figura 27

Pantalla de error del compilador KRC2.



Nota: Se aprecia el mensaje que se obtiene en caso de que exista un error en la programación.

6.2.3. Obtención de condicionales para la activación del posicionamiento del robot KUKA KR5-2 ARC HW, a través de sus entradas digitales.

Se realizó la programación mediante python de 255 condicionales para cada punto del espacio de trabajo asignado del robot KUKA KR5-2 ARC HW, siendo el condicional agregado en el archivo SRC. Los condicionales se realizaron de manera booleana activando 8 entradas digitales del Robot KUKA KR5-2 ARC HW. La lógica de programación empieza por obtener la forma binaria de los 255 puntos de trabajo, una vez obtenidos se incluyen en un arreglo que contendrá todos los puntos en forma binaria, seguidamente se crea una nueva lista dentro de un bucle for, donde en cada ciclo, la nueva lista recorre uno a uno los números binarios de la primera lista creada. Mediante condicionales se pregunta la existencia de 0 y 1 en el número binario de cada ciclo y así se crea la lógica que permite crear los 255 condicionales activando desde el bit 00000001 (uno decimal) y 11111111 (doscientos cincuenta y cinco decimales).

6.3. Comunicación C.M.K.

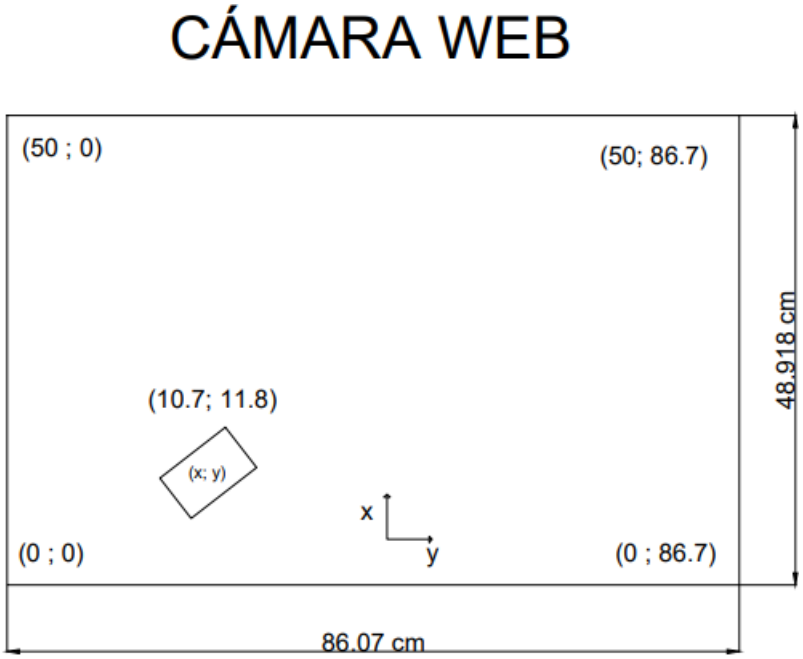
En la siguiente sección se explicará la conexión entre la cámara, el microcontrolador y finalmente el robot KUKA KR5-2 ARC HW.

6.3.1. Redimensionamiento de los objetos captados en cámara hacia el espacio de trabajo del robot KUKA KR5-2 ARC HW.

A continuación, se explica el proceso por el cual se obtiene información en un principio desde centímetros dados por el código de dimensionamiento utilizado para la cámara, hasta dimensiones referenciadas al espacio de trabajo total del robot KUKA KR5-2 HW.

Figura 28

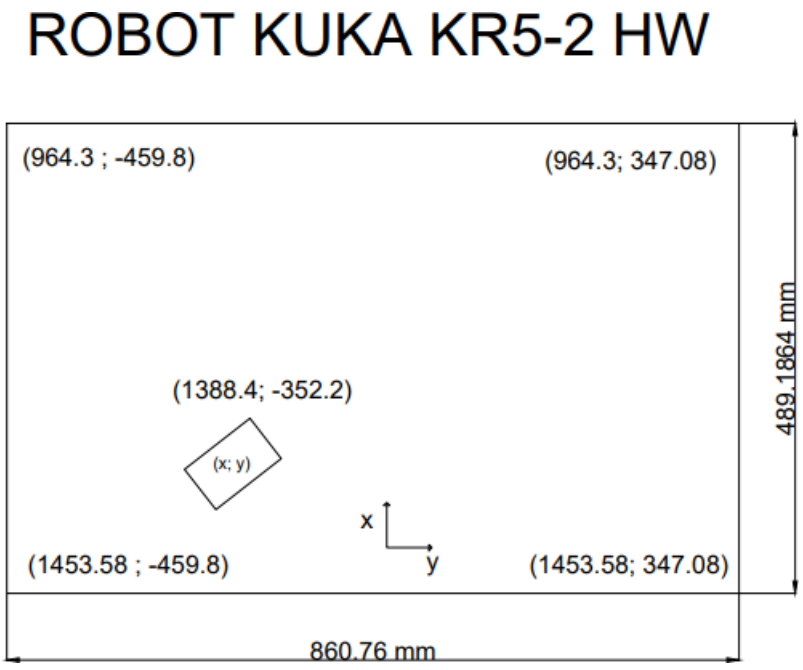
Plano de trabajo cámara web.



Nota: Plano de trabajo captado por la cámara web en resolución 720p.

Figura 29

Plano de trabajo robot KUKA KR5-2 ARC HW.



Nota: Plano de trabajo real designado para la herramienta de clasificación.

1. Transformar valores dados por la cámara en valores de posición globales del robot KUKA KR5-2 ARC HW:

   Obtenidos los datos en centímetros de base y altura de los objetos captados por la cámara, se procede a relacionar los siguientes parámetros:

   - Punto inicial x kuka (1453.58 cm)
   - Punto inicial y kuka (-459.8 cm)
   - Longitud total en el eje x Kuka (489.1866 cm)
   - Longitud total en el eje y Kuka (860.7555 cm)
   - Longitud total en el eje x cámara (50 cm)
   - Longitud total en el eje y cámara (86.7 cm)
   - Punto del objeto en el eje x cámara (objeto x cm)

- Punto del objeto en el eje y cámara (Objeto y cm)

Los parámetros mencionados se utilizan en las siguientes ecuaciones para pasar del plano de trabajo de la cámara web hasta el plano de trabajo del robot KUKA KR5-2 ARC HW.

Figura 30

Ecuaciones.

$$Dato\ x = x\ punto\ inicial\_kuka - \frac{(x_{ref\ kuka} * objeto\ x\ cm)}{Longitud\ total\ en\ el\ eje\ x\ cámara}$$

$$Dato\ y = y\ punto\ inicial_{kuka} + \frac{y_{ref\ kuka} * objeto\ y\ cm)}{Longitud\ total\ en\ el\ eje\ y\ cámara}$$

Nota: La variable (x) es el eje vertical mientras la variable (y) es el eje horizontal del espacio de trabajo.

2. Aproximar valores de la cámara con posiciones globales del robot KUKA KR5-2 ARC HW a los 255 puntos programados para el espacio de trabajo de la herramienta de clasificación:

   Obtenidas las dimensiones de Datox y Datoy se procede primeramente a aproximar estas dimensiones a los puntos más cercanos relacionados con los 255 puntos programados para el espacio de trabajo del robot KUKA KR5-2 ARC HW. Para su realización se creó funciones tanto para los datos en el eje vertical x como en el eje horizontal y. Utilizando un condicional **if** con la estructura:

   ```
   if (Objx<Dotkukax or abs(Dotkukax – Objx) < 15) and (Objx > Dotkukax-16.3)
   ```

   Siendo el valor de 16.3 la mitad entre cada punto para la coordenada vertical de los 255 puntos programados, devolviendo el valor de cada punto kuka x en caso de que se cumpla la condición. Este mismo proceso se realizó para el eje horizontal y con la estructura:

   ```
   if (Objy>Dotkukay or abs(Dotkukay - Objy) <24) and (Objy < Dotkukay+26.9)
   ```

Con el uso de estas funciones se logra aproximar desde los puntos captados por la cámara web hasta los puntos programados para el robot KUKA KR5-2 ARC HW. Obtenidos estos valores, se asignan variables resultadox y resultadoy, esto con el fin de crear una nueva función para el envío tanto de la coordenada x como la coordenada y hacia el microcontrolador mediante conexión serial. Para esto se toman los datos de resultadox y resultadoy con la siguiente estructura:

```
if resultadox == Dotkukax and resultadoy == Dotkukay:
```

Devolviendo un valor en formato string el cual será uno de los 255 puntos asignados para el espacio de trabajo del robot KUKA KR5-2 ARC HW.

6.3.2. Programación del microcontrolador para la recepción de datos y posterior activación de relés para la activación de posicionamiento del robot.

1. Creación de un programa para la activación de las salidas del microcontrolador:

   El valor en formato string que se envía al microcontrolador es de tres caracteres, esto debido a que el microcontrolador recibe carácter por carácter, a continuación, se muestra un ejemplo del valor enviado desde Python hacia el microcontrolador:

```
if dotx == 1453.6 and doty == -406.0:
Datok = "001"
return Datok
if dotx == 964.6 and doty == 347.2
Datok = "255"
return Datok
```

   Se utilizó el PIC 16F18875 como microcontrolador para la activación del módulo de relés. Se programó mediante el software MPLAB, utilizando todo un puerto para el envío máximo de 8 bits, que corresponden a las posiciones desde el número 1 al número 255 del espacio de trabajo utilizado para el robot KUKA KR5-2 ARC HW. Adicionalmente se crearon las líneas de código C mediante Python con el objetivo de tener la posibilidad de obtener un espacio de trabajo mayor en caso de ser necesario.

   La transferencia de archivos se realizó mediante código ASCII, empleando funciones que envíen desde la programación en python datos string hacia el microcontrolador, el mismo que recibe el código string en formato ASCII para posteriormente convertir

la información a un valor entero, esto con el uso de la función atoi. Finalmente se realizó un Switch Case en el código del microcontrolador con el objetivo de activar en forma binaria las entradas (1 a 8) del robot KUKA KR5-2 ARC HW dependiendo de la posición en la cual se encuentre el objeto.

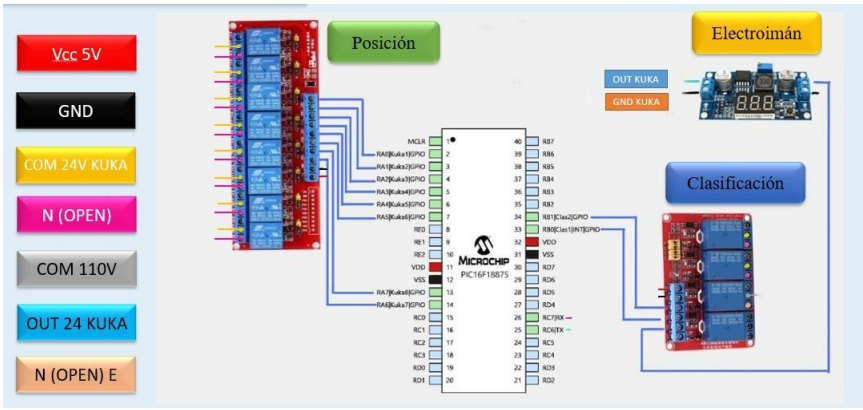
2. Activación de los puntos en el lenguaje KRL:

   La activación de los puntos en el lenguaje KRL se realiza mediante condiciones If booleanas las cuales preguntarán si están en un estado TRUE las diferentes entradas del robot KUKA KR5-2 ARC HW, estas entradas son activadas mediante los módulos relés conectados al microcontrolador. En cada secuencia If se encontrará el punto al cual debe desplazarse el robot KUKA KR5-2 ARC HW.

   Realizados estos pasos el robot KUKA KR5-2 ARC HW será capaz de colocarse en cualesquiera de los 255 puntos programados, este mismo código se utilizará más adelante para el apartado del código KRL de clasificación de objetos.

Figura 31

Esquemático posición.



Nota: Conexión realizada para el posicionamiento del robot KUKA KR5-2 HW en un objeto captado por la cámara web.

6.3.3. Condición de clasificación de acuerdo con el área de los objetos.

1. Obtención de las dimensiones para el cálculo de área:

Debido a que la información de las áreas de los objetos se encuentran dentro de una lista, se procedió a extraer la información de la lista individualmente para cada objeto mediante el código '.join(map(str, detecciones[1])'), el cuál retorna un valor en formato string del área, dado que se trabaja con decimales se procede a convertirlo a un formato float.

2. Designación del tamaño de áreas:

   Para la consideración de tamaño de las áreas se realizó una función llamada Clasificación la cual recibe los valores de las áreas en tipo float. Se considera la siguiente información como referencia para designar si un objeto es grande, pequeño o mediano respectivamente.

   - Objeto grande: Área entre (160 Y 80 cm^2)
   - Objeto mediano: Área entre (35 Y 79.9 cm^2)
   - Objeto pequeño: Área entre (5 y 34.9 cm^2)

   Obtenidos los valores de tamaño de referencia de los objetos se procede a preguntar dentro de la función Clasificación en que valor se encuentra el área del objeto al cual se va a clasificar, devolviendo la función una variable en formato String de la siguiente forma:

   - Objeto grande: Retorna un String (G)
   - Objeto mediano: Retorna un String (M)
   - Objeto pequeño: Retorna un String (P)

   Se realizó de esta forma la lógica de programación debido a que el microcontrolador recibe caracteres en formato ASCII.

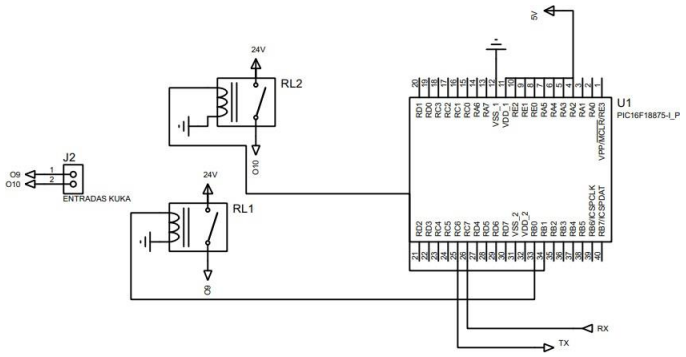
3. Envío y recepción de áreas hacia el microcontrolador para posterior activación de relés:

   Una vez se tiene la variable de retorno de la función Clasificación se procede a enviar al microcontrolador con el atributo pic.write(envio_tamaño.encode()). Dado que el microcontrolador convierte los valores en formato ASCII recibidos desde Python se obtiene como entrada la letra del tamaño del área. En la programación de lenguaje C se realizó el mismo código de envío de datos de posicionamiento del robot para el envío de

datos de clasificación. En este caso se utilizan las entradas 9 y 10. A continuación se indica el esquema de conexión usado para la clasificación.

Figura 32

Esquemático clasificación.



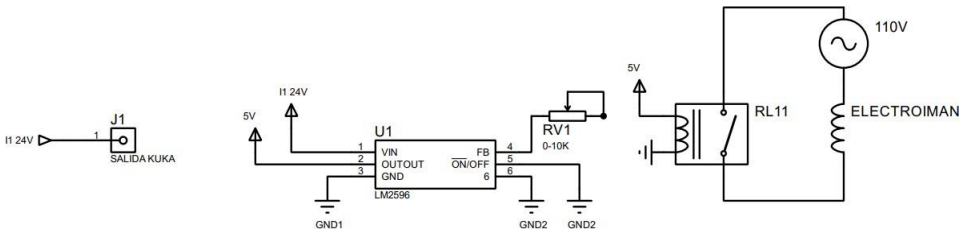
Nota: Conexión utilizada para la clasificación de objetos de la herramienta tecnológica.

4. Programación KRL para la clasificación de áreas:

   El código de programación en formato KRL para que clasifique las áreas de los objetos se basa en la activación de las entradas 9 Y 10 del robot KUKA KR5-2 ARC HW. Se utilizó una condición `If` de forma booleana para cada tipo de tamaño, asignando un espacio específico para la colocación del objeto. Así mismo se agregó una activación de salida de la entrada 1 del robot KUKA KR5-2 ARC HW para la activación de un relé adicional que energiza el electroimán, el cuál es el encargado de la toma del objeto desde su punto de ubicación hasta la zona de clasificación. A continuación, se indica el esquema de conexión del electroimán.

Figura 33

Esquemático electroimán.



Nota: Conexión realizada para la activación del electroimán.

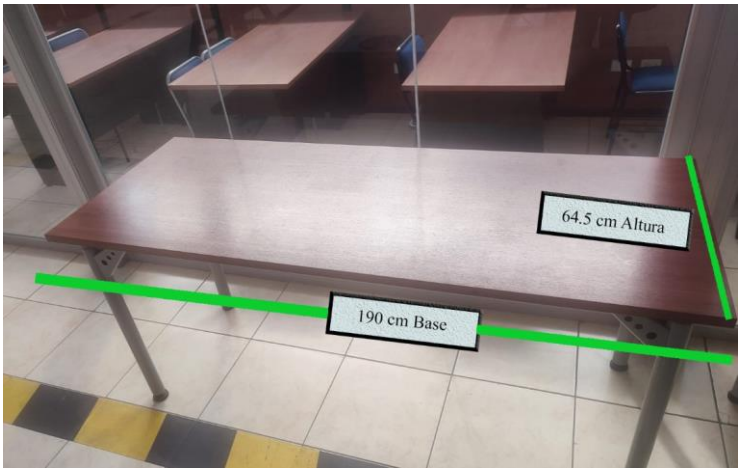
6.4. Adecuación del espacio de trabajo para la herramienta tecnológica.

A continuación, se muestran las distintas adecuaciones que se realizaron en el laboratorio de robótica para la implementación de la herramienta tecnológica.

Para el espacio de trabajo de la herramienta tecnológica se utilizó una mesa proporcionada por el laboratorio de robótica con las dimensiones que muestra la figura 34:

Figura 34

Mesa de trabajo UPS.



Nota: Dimensiones totales de la mesa de trabajo.

Adicionalmente se utilizó una tela color blanco con las dimensiones designadas para el espacio de trabajo. Con el objetivo de que la cámara logre diferenciar los objetos que se coloquen en la mesa de trabajo.

A partir de una plancha de metal se realizaron doce objetos , dichos objetos se cortaron milimétrica-mente para comprobar la efectividad de la herramienta de clasificación de acuerdo a dimensiones impuestas, estos objetos se utilizarán para que la herramienta tecnológica detecte los mismos y clasifique de acuerdo a sus dimensiones.

Figura 35

Elementos de clasificación.

Nota: Objetos metálicos de diferentes dimensiones.

Para la toma de los objetos se utilizó un electroimán el cual se acopla en el efector final del robot KUKA KR5-2 ARC HW, es alimentado mediante una conexión a la red eléctrica a 110V y su activación se realiza mediante una salida digital del brazo robótico KUKA KR5-2 ARC HW conectado al módulo relé de 4 canales.

Figura 36

Electroimán.

Nota: Dispositivo utilizado para la toma de objetos.

Finalmente se realizó un gabinete para la colocación de todos los componentes electrónicos inmiscuidos en la herramienta tecnológica con el fin de que las conexiones sean sencillas y óptimas para la instalación rápida de la herramienta tecnológica de identificación y clasificación de objetos. A continuación, se enlistan los componentes del gabinete.

- PIC 16 F18875
- Módulo relé 8 canales 5v
- Módulo relé 4 canales 5v
- Fuente 5 voltios
- Regulador reductor Step Down LM2596 3A
- Usb TTL

7. Resultados

7.1. Desempeño del programa de visión artificial para la identificación y clasificación de objetos.

Para la evaluación del desempeño del programa de visión artificial de identificación y clasificación de objetos, se procedió a medir físicamente las dimensiones de los objetos captados por la cámara web para comprobar que los datos que devuelve el software son los correctos. De la misma manera se realizó la comprobación para el posicionamiento de los objetos en el plano X-Y del espacio asignado para la identificación.

Se obtuvieron los siguientes resultados:

Tabla 3

Resultados posición de objetos.

Nro	Coordenada (x-y) medible	Coordenada (x-y) software	Error absoluto X	Error absoluto Y	Resultado
1	X=5 ; Y =5	X=5,2 ; Y =5,1	X=0,2	Y=0,1	Correcto
2	X=5 ; Y =10	X=5,1 ; Y =9,6	X=0,1	Y=0,4	Correcto
3	X=10; Y =13	X=10,3 ; Y =13,3	X=0,3	Y=0,3	Correcto
4	X=15 ; Y =15	X=14,8 ; Y =14,8	X=0,2	Y=0,2	Correcto
5	X=20 ; Y =20	X=19,5 ; Y =20,9	X=0,5	Y=0,1	Correcto
6	X=25 ; Y =25	X=27,9 ; Y =24,5	X=2,1	Y=0,5	Correcto
7	X=30 ; Y =30	X=29,9 ; Y =29,4	X=0,1	Y=0,6	Correcto
8	X=30 ; Y =35	X=29,6 ; Y =34,8	X=0,4	Y=0,2	Correcto
9	X=40 ; Y =40	X=39,7 ; Y =39,5	X=0,3	Y=0,5	Correcto
10	X=40 ; Y =45	X=39,6 ; Y =45,3	X=0,4	Y=0,3	Correcto

Nota: Tabla resultados posición de objetos, como se puede observar en la tabla, existe un promedio de error absoluto de 0.32 siendo las coordenadas estimadas por la cámara aceptables frente a la medición real que se realizó mediante un flexómetro en el espacio del trabajo.

Tabla 4

Resultados dimensión de objetos.

Nro	Base- Altura real (cm)	Base- Altura software	Error absoluto	% Error relativo	Resultado
1	25	24,8	0,2	0,81	Correcto
2	30,25	29,8	0,45	1,51	Correcto
3	36	35,7	0,3	0,84	Correcto
4	42,25	41	1,25	3,05	Correcto
5	49	48,9	0,1	0,20	Correcto
6	56,25	61	4,75	7,79	Correcto
7	64	70,2	6,2	8,83	Correcto
8	72,25	78,7	6,45	8,20	Correcto
9	81	86,4	5,4	6,25	Correcto
10	90,25	97,3	7,05	7,25	Correcto
11	100	108,1	8,1	7,49	Correcto
12	110,25	120,6	10,35	8,58	Correcto

Nota: Tabla resultados dimensión de objetos, se puede observar como a medida que la dimensión del objeto crece, el error relativo aumenta, sin embargo, considerando los rangos de clasificación se considera aceptable para el proceso de clasificación de la herramienta tecnológica.

7.2. Comprobación de la comunicación del microcontrolador para recepción y envío de datos.

Para la comprobación de la comunicación del microcontrolador se procedió a utilizar un bucle For que recorra los 255 puntos impuestos en las salidas digitales del puerto A del microcontrolador, activando los canales del módulo de relés mediante envío de datos desde Python hacia el microcontrolador mediante el usb ttl, comprobando que la comunicación tanto de envío como recepción es correcta.

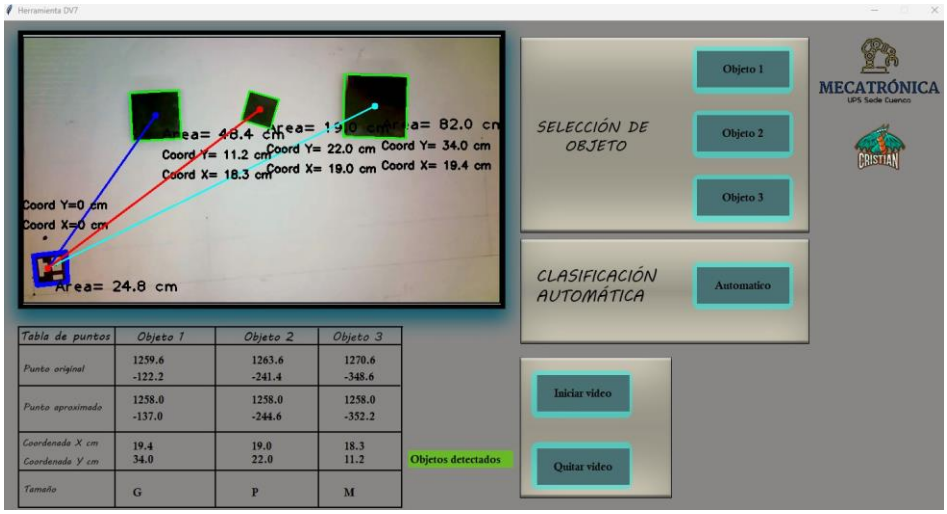
7.3. Creación de una interfaz gráfica para el envío de datos hacia el robot KUKA KR5-2 HW para la clasificación de objetos a través de sus entradas digitales.

Se utilizó la librería TKINTER para la realización de la interfaz gráfica, además se diseñó un modelo base en Adobe Photoshop con el fin de que tenga un entorno estético y fácil de

operar. La interfaz gráfica cuenta con 4 botones los cuáles se encargan del envió de datos para accionar al brazo robótico KUKA KR5-2 ARC HW, adicionalmente indica la posición en la cual se encuentra el objeto, así como su punto aproximado en el espacio de trabajo designado para la herramienta tecnológica. Finalmente, el entorno gráfico cuenta con dos botones los cuáles cumplen la función on/off para la identificación de los objetos.

Figura 37

Interfaz gráfica.



Nota: Se visualiza la interfaz gráfica de la herramienta tecnológica de identificación y clasificación.

7.4. Evaluación del desempeño de la herramienta tecnológica de identificación y clasificación de objetos para el robot KUKA KR5-2 ARC HW

Realizadas las pruebas de posicionamiento y clasificación, se procede a realizar pruebas de todo el sistema de la herramienta tecnológica con el fin de determinar si la herramienta tecnológica es o no funcional, para esto se colocaron en posiciones aleatorias 12 diferentes figuras para corroborar si el brazo robótico KUKA KR5-2 ARC HW se posiciona por encima del objeto y lo traslada a la zona de clasificación mediante la activación del electroimán de acuerdo con dimensiones del objeto. A continuación, se presentan los resultados obtenidos.

Tabla 5

Resultados finales.

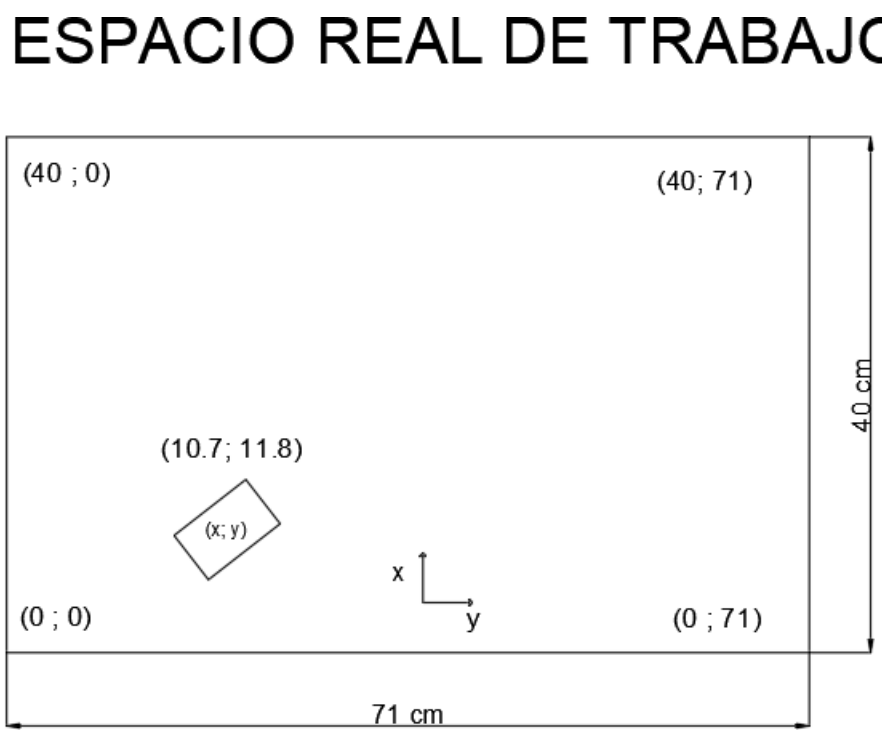
Nro	Dimensión (Área cm^2)	Punto objeto	Punto calculado	Resultado
1	25	x= 1347,5 y=52,28	x=1355,8 y=78,2	No
2	30,25	x=1299,6 y=34,2	x=1290,6 y=24,4	Si
3	36	x=1273,3 y=72,2	x=1258 y=-137	Si
4	42,25	x=1270,6 y=-356,5	x=1258 y=-352,2	No
5	49	x=1128,6 y=-244,4	x=1290,6 y= -137	Si
6	56,25	x=1296,6 y=-128,2	x=1290,6 y= -137	Si
7	64	x=1243,6 y=-53.7	x=1258 y=-29,4	Si
8	72,25	x=1280,6 y=-366,5	x=1290,6 y=-352,2	Si
9	81	x=1331,6 y=-289	x=1323,2 y=-294,4	Si
10	90,25	x=1120,6 y=-74,6	x=1127,6 y=-83,2	Si
11	100	x=1351,6 y=-380,4	x=1355,8 y=-406	Si

Nota: Se visualiza la tabla de resultados finales de la herramienta tecnológica de identificación y clasificación, dando como resultados finales que la herramienta tecnológica se logra aproximar a los objetos para poder clasificarlos, sin embargo, dado la separación de 5.38cm entre puntos de la matriz, existe ciertas posiciones de objetos tal es el caso del nro. 1 y 4 de la tabla 4. Para solucionar esta separación se debería aumentar la cantidad de puntos de la matriz para disminuir la separación entre puntos.

Finalmente se comprobó que el espacio de trabajo dada la altura y posición de la cámara web disminuye del espacio original calculado, por tal motivo se presenta a continuación las dimensiones reales de operación de la herramienta de trabajo.

Figura 38

Dimensiones espacio de trabajo.



Nota: Estas dimensiones pueden llegar a ser iguales al espacio original calculado, sin embargo, dadas las condiciones del laboratorio se optó por utilizar el espacio de trabajo anteriormente mencionado.

7.5. Análisis de costos

El análisis de costos es el proceso de identificación de los recursos necesarios para llevar a cabo el proyecto. Determina la calidad y cantidad de recursos necesarios. Entre otros factores, se analizó el costo del proyecto en términos de dinero. Está enfocado en el análisis de los factores del proyecto que se encuentra en ejecución o ya se ejecutó.

7.5.1. Costo presupuestado

Los gastos presupuestarios se establecen desde un punto de vista contable, para concretar la relación de gastos. En primer lugar, se consideró necesario determinar los costos de los

materiales y el equipo implementado que fue utilizado para el desarrollo del proyecto, tomando como referencia visualizar la tabla 6.

Tabla 6

Análisis de costo presupuestado de los materiales y equipos

Equipos/Materiales	Cantidad (u)	Costo ($)	Subtotal ($)
PIC 16F18875	2	6.50	13.00
Cámara Web	1	50.00	50.00
USB TTL	1	5.00	5.00
Trípode	1	20.00	20.00
Electroimán	1	45.00	45.00
Módulo LM2596	1	5.00	5.00
Cables	1	25.00	25.00
Módulos de Relés	2	20.00	40.00
Área de Trabajo	1	40.00	40.00
Gabinete	1	12.00	12.00
Figuras para la clasificación	8	7.50	60.00
		Total ($)	243.00

Nota: Se puede observar los materiales y equipos utilizados para la implementación de la herramienta tecnológica de identificación y clasificación de objetos.

Después de haber seleccionado todos los recursos necesarios con respecto a todos los equipos, materiales y herramientas necesarias para implementar el proyecto, se obtuvo un valor total de 243.00.

7.5.2. Costo por actividades

Los costos por actividades están estipulados en función de la cantidad de tareas que se requieren realizar para la implementación de la herramienta tecnológica, para ello se tomó como referencia el costo unitario por hora en relación con el costo por hora de un ingeniero como de un tecnólogo experimentado.

Una vez designados estos valores se planteó una tabla referente al precio de cada actividad a desarrollar, dentro de las actividades más demandantes se encuentra la programación python.

Tabla 7

Costo por actividades.

Actividad	Encargado	Personas	Tiempo de duración (horas)	Costo de mano de obra
Estudio del proceso	Ingeniero	1	16	130
Software de identificación de objetos	Ingeniero	1	24	259.88
Software de clasificación de objetos	Ingeniero y tecnólogo	2	20	216
Programar PIC para envió de datos	Ingeniero	1	20	162.50
Programar KRL para posicionamiento del robot	Ingeniero	1	24	259.88
Calibración de cámara web	Ingeniero	1	20	216.56
Transformación de píxel a centímetro	Tecnólogo	1	32	260.00
Aproximar matriz puntos cámara web a matriz puntos KRL	Ingeniero	1	12	97.50
Conexión de la herramienta tecnológica	Ingeniero y tecnólogo	2	8	67.70
Adecuación del espacio físico	Tecnólogo	1	12	97.50
Instalación del electroimán	Tecnólogo	1	8	67.70
Diseño de la interfaz gráfica	Tecnólogo	1	8	195.00
Conexión PC-KRC2 en tiempo real	Tecnólogo	1	8	32.00
Clasificación de diferentes objetos	Tecnólogo	1	8	21.25
Charla de capacitación	Ingeniero	1	16	130.00

Nota: Tabla de actividades realizadas en la implementación de la herramienta tecnológica.

7.5.3. Costo por paquete

Tabla 8

Tabla costo por paquete.

Paquete	Actividad	Sub Total	Total
Creación del código orientado a visión artificial	Estudio del proceso	130	605.88
	Software de identificación de objetos	259.88	
	Software de clasificación de objetos	216	
Envío y posicionamiento del robot	Programar PIC para envió de datos	162.50	422.38
	Programar KRL para posicionamiento del robot	259.88	
Transformación matriz de puntos	Calibración de cámara web	216.56	574.06
	Transformación de píxel a centímetro	260.00	
	Aproximar matriz puntos cámara web a matriz puntos KRL	97.50	
Entorno Físico	Conexión de la herramienta tecnológica	67.70	232.90
	Adecuación del espacio físico	97.50	
	Instalación del electroimán	67.70	
Implementación de la herramienta tecnológica	Diseño de la interfaz gráfica	195.00	227.00
	Conexión PC-KRC2 en tiempo real	32.00	
Pruebas del Sistema	Clasificación de diferentes objetos	21.25	21.25
Capacitación de los usuarios	Charla de capacitación	130.00	130.00

Nota: Tabla de paquete de actividades realizadas para la implementación de la herramienta tecnológica.

7.5.4. Costo total del proyecto

Realizado el análisis de los costos relacionados con la obtención del material, equipos y herramientas, además de los costos de cada actividad para la realización del proyecto, basándose en el costo de la mano de obra, se realizó una suma de los análisis para obtener el valor total de la implementación del proyecto. Donde, el costo total presupuestado es CTP, el costo presupuestado inicial es CP y el costo presupuestado por actividades es CPA. A continuación, se presenta el cálculo para la obtención del costo total:

$$CTP = CP + CPA$$
$$CTP = 243.00 + 2213.47$$
$$CTP = 2456.47$$

Luego de realizar el respectivo cálculo, se determinó que el costo total del proyecto está valorado por una cantidad de: $2456.47.

8. Conclusiones

- La utilización de la librería Opencv orientado a visión artificial permitió identificar y clasificar diferentes objetos con diferentes dimensiones en el espacio de trabajo designado para el robot KUKA KR5-2 ARC HW.
- Mediante programación en lenguaje C y uso del microcontrolador PIC 16F18875 se estableció la comunicación entre la cámara web conectada a una computadora y el controlador del robot KUKA KR5-2 ARC HW. Enviando datos tanto de dimensión de área como posicionamiento de los objetos en el espacio de trabajo designado. Con el fin de que el brazo robótico se posicione en el punto del objeto deseado exitosamente.
- Con el uso del lenguaje de programación Python se realizó la programación KRL del robot KUKA KR5-2 ARC HW. Dicha programación contiene los diferentes puntos de trabajo, así como los condicionales de posicionamiento y clasificación. Permitiendo obtener diferentes matrices de posición dependiendo de la precisión deseada en el espacio de trabajo del robot KUKA KR5-2 ARC HW.
- Con la evaluación de desempeño de la herramienta tecnológica se concluye que el sistema es capaz de reconocer diferentes dimensiones de objetos, así como la posición de los mismos, dado que se realizó una matriz de 16x16 puntos, el brazo robótico se aproxima al punto más cercano al objeto posible, desplazando el mismo hacia la zona de clasificación mediante un electroimán colocado en el actuador final del brazo robótico. Todo este sistema se visualiza en una interfaz gráfica la cual contiene información de cada objeto permitiendo clasificarlos tanto de forma individual como automática.

9. Recomendaciones

- Se debe realizar una calibración previa del sistema, comprobando el posicionamiento de la cámara web y el espacio de trabajo del brazo robótico con el fin de que el sistema sea preciso, así mismo para el modo automático se tomó en cuenta una velocidad media del brazo robótico por lo que si se aumenta o disminuye la misma, se debe considerar este aspecto.

- Para la programación del brazo robótico es recomendable obtener el formato de código KRL para que mediante Python se creen los diferentes códigos KRL de manera automática, considerando que el archivo con la extensión DAT contiene la información de posición de los puntos mientras que el archivo de extensión SRC se encarga del código en general.
- En caso de implementar un sistema basado en tres dimensiones, se debe considerar los límites totales de trabajo del brazo robótico, esto con el fin de que el brazo robótico no utilice una parada de emergencia por límites, teniendo que reiniciar todo el sistema para que el brazo robótico se pueda volver a operar.
- Al momento de manipular las entradas del brazo robótico, en caso de que exista un corto circuito, se activará el fusible del módulo de entradas y salidas del brazo robótico por lo que se debe colocar un fusible nuevo y reiniciar todo el sistema para que vuelva a funcionar el módulo de entradas y salidas.

Referencias

Aguayo, P. (2004). *Introducción al microcontrolador*. paguayo@olimex.cl.

Barahona Guamani, E. S. (2019). *Navegación autónoma basada en maniobras bajo estimación de posturas humanas para un robot omnidireccional kuka youbot* (B.S. thesis).

Bernard, M. (2019). *7 amazing examples of computer and machine vision in practice.* Descargado 2022-12-13, de https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=7175a2141018

Cabrera Sarmiento, B. F., y Martinez Aviles, M. J. (2022). *Diseño de una herramienta de acople para el proceso ~de soldadura mig aplicada al robot kuka kr5* (B.S. thesis).

Camarillo, D. B., Krummel, T. M., y Salisbury, J. K. (2004, 10). Robotic technology in surgery: Past, present, and future. *The American Journal of Surgery*, *188* (4), 2-15S. (Copyright - © 2004 Excerpta Medica Inc; Última actualización - 2013-02-07)

Eckstein, W., y Steger, C. (1999). The halcon vision system: an example for flexible software architecture. En *Proceedings of 3rd japanese conference on practical applications of real-time image processing* (pp. 18–23).

Eloi, G. (2012). *Visión artificial*. FUOC Fundación para la Universitat Oberta de Catalunya.

Guaraca Medina, P. J., y Ochoa Ochoa, J. L. (2015). *Estudio de la programación y operación de los robots industriales kuka kr16-2 y kr5-2 arc hw* (B.S. thesis).

Ivet Challenge, R. A. B., Yanet Díaz. (2014). El lenguaje de programación python.

Jiménez Bravo, R. (2018). Sistema de seguimiento de objetos usando opencv, aruco y filtro de kalman extendido.

Jiva, E. I. (2019). Desarrollo de la teleoperación de robots industriales y colaborativos mediante técnicas avanzadas de visión artificial.

KUKA.AG. (2019). *Kuka.visiontech.* Descargado 2019-10-04, de https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/software/ software-de-aplicaciÃşn/kuka_visiontech

Microchip. (2019). *Pic16(l)f18855/75.* Descargado 2022-12-13, de https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/PIC16%28L%29F18855-75-Data-Sheet-40001802H.pdf

Muñoz Rodríguez, P. (2011). *Desarrollo de aplicaciones integrando robótica y visión en un robot industrial kuka para demostrar sus capacidades* (B.S. thesis).

Pillajo Lemache, A. M. (2019). *Sistema de formación en robótica industrial basada en células robotizadas kuka y visiÓn artificial* (B.S. thesis).

Takeyas, B. L. (2007). Introducción a la inteligencia artificial. *Instituto Tecnológico de Nuevo Laredo. Web del autor: http://www. itnuevolaredo. edu. mx/takeyas*.

Vigneswaran, C., Madhu, M., y Rajamani, R. (2012). Inspection and error analysis of geneva gear on machine vision system using sherlock™ and vb 6.0 algorithm. En *2012 international conference on machine vision and image processing (mvip)* (pp. 193–196).

Workana. (2021). *C++: Qué es, para qué sirve, ventajas y desventajas.* Descargado 2019-04-06, de https://i.workana.com/glosario/que-es-c/

ANEXOS

Anexo A: Programación principal de la herramienta tecnológica con interfaz gráfica

Figura 39

Programación principal de la herramienta tecnológica parte 1

```python
import cv2
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import messagebox
import numpy as np
from scipy.spatial import distance as dist
from imutils import perspective
from imutils import contours
from _objeto_deteccion import*
import serial,time
import os

from Base_Puntos import Datoskukax, Datoskukay,Kukadot,clasificacion
import imutils
def midpoint(ptA, ptB):return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5)
parametros=cv2.aruco.DetectorParameters_create()
aruco_dict=cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_50)
#referencia
detector=Clase_detector()

azul = (255, 0, 0)
verde = (0, 255, 0)
rojo = (0, 0, 255)
cian=(0,255,255)
negro=(0,0,0)


x_ref_kuka=489.1866
y_ref_kuka=860.7555
kuka_x_inicial=1486.2
kuka_y_inicial=-513.6

#Creación del.exe
def resource_path2(relative_path):...
ventana=tk.Tk()
ventana.geometry("1440x750")
ventana.title("Herramienta DV7")
ventana.resizable(width=False, height=False)
path=resource_path2("fondo.png")
fondo=tk.PhotoImage(file=path)
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 40

Programación principal de la herramienta tecnológica parte 2

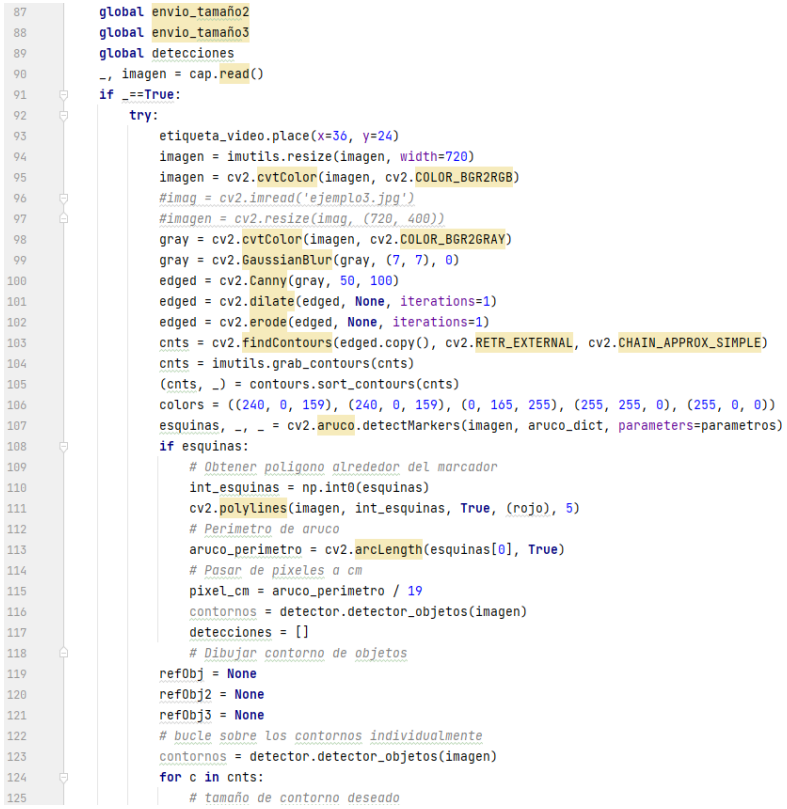
```
fondo1=tk.Label(ventana,image=fondo).place(x=0, y=0, relwidth=1, relheight=1)
global detecciones
global dato1x
global dato2x
global dato3x
global dato1y
global dato2y
global dato3y
global  envio1
global  envio2
global  envio3
global envio_tamaño1
global envio_tamaño2
global envio_tamaño3
global puertocam
cap= None
puertopic = input("Ingrese el puerto de conexión serial ")
pic=serial.Serial("COM"+puertopic,9600)
puertocam = int(input("Ingrese el puerto de la cámara web "))
def iniciar_video():
    global cap

    cap = cv2.VideoCapture(puertocam)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
    iniciar()
def iniciar():
    global cap
    global Objeto1x
    global Objeto2x
    global Objeto3x
    global Objeto1y
    global Objeto2y
    global Objeto3y
    global envio1
    global envio2
    global envio3
    global envio_tamaño1
    global envio_tamaño2
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 41

Programación principal de la herramienta tecnológica parte 3

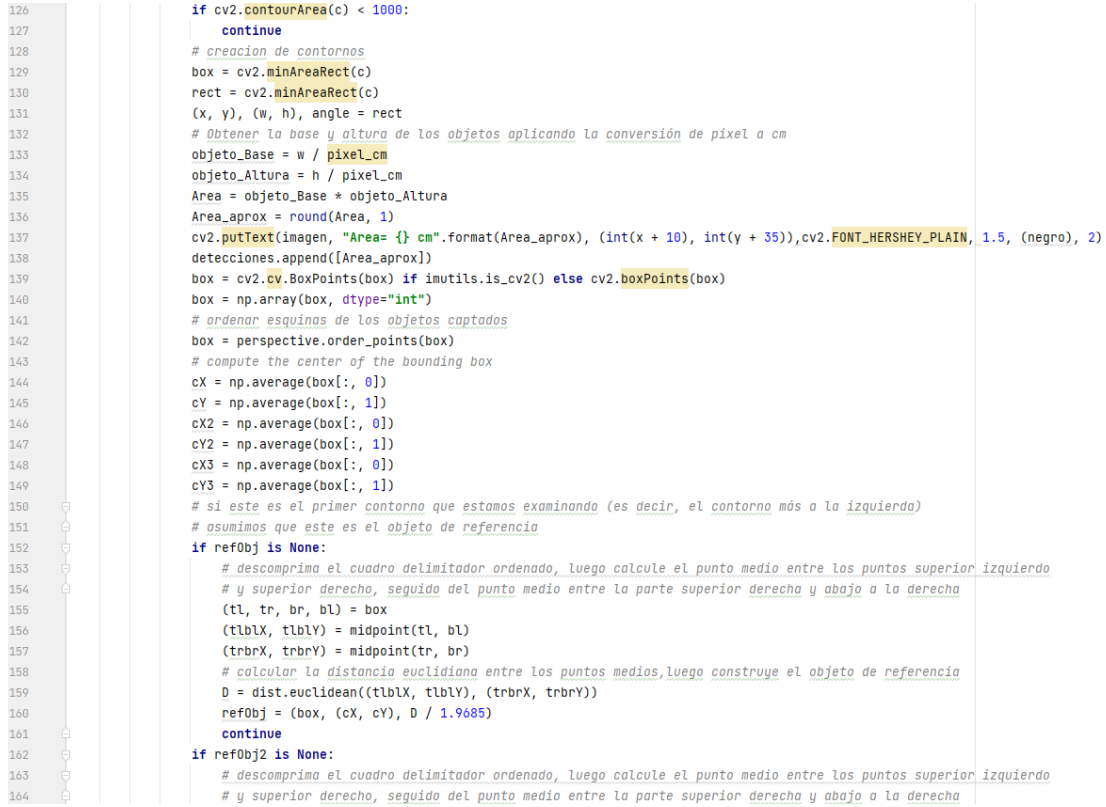


```
    global envio_tamaño2
    global envio_tamaño3
    global detecciones
    _, imagen = cap.read()
    if _==True:
        try:
            etiqueta_video.place(x=36, y=24)
            imagen = imutils.resize(imagen, width=720)
            imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
            #imag = cv2.imread('ejemplo3.jpg')
            #imagen = cv2.resize(imag, (720, 400))
            gray = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (7, 7), 0)
            edged = cv2.Canny(gray, 50, 100)
            edged = cv2.dilate(edged, None, iterations=1)
            edged = cv2.erode(edged, None, iterations=1)
            cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
            cnts = imutils.grab_contours(cnts)
            (cnts, _) = contours.sort_contours(cnts)
            colors = ((240, 0, 159), (240, 0, 159), (0, 165, 255), (255, 255, 0), (255, 0, 0))
            esquinas, _, _ = cv2.aruco.detectMarkers(imagen, aruco_dict, parameters=parametros)
            if esquinas:
                # Obtener poligono alrededor del marcador
                int_esquinas = np.int0(esquinas)
                cv2.polylines(imagen, int_esquinas, True, (rojo), 5)
                # Perimetro de aruco
                aruco_perimetro = cv2.arcLength(esquinas[0], True)
                # Pasar de pixeles a cm
                pixel_cm = aruco_perimetro / 19
                contornos = detector.detector_objetos(imagen)
                detecciones = []
                # Dibujar contorno de objetos
            refObj = None
            refObj2 = None
            refObj3 = None
            # bucle sobre los contornos individualmente
            contornos = detector.detector_objetos(imagen)
            for c in cnts:
                # tamaño de contorno deseado
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 42

Programación principal de la herramienta tecnológica parte 4



```
            if cv2.contourArea(c) < 1000:
                continue
            # creacion de contornos
            box = cv2.minAreaRect(c)
            rect = cv2.minAreaRect(c)
            (x, y), (w, h), angle = rect
            # Obtener la base y altura de los objetos aplicando la conversión de pixel a cm
            objeto_Base = w / pixel_cm
            objeto_Altura = h / pixel_cm
            Area = objeto_Base * objeto_Altura
            Area_aprox = round(Area, 1)
            cv2.putText(imagen, "Area= {} cm".format(Area_aprox), (int(x + 10), int(y + 35)),cv2.FONT_HERSHEY_PLAIN, 1.5, (negro), 2)
            detecciones.append([Area_aprox])
            box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box)
            box = np.array(box, dtype="int")
            # ordenar esquinas de los objetos captados
            box = perspective.order_points(box)
            # compute the center of the bounding box
            cX = np.average(box[:, 0])
            cY = np.average(box[:, 1])
            cX2 = np.average(box[:, 0])
            cY2 = np.average(box[:, 1])
            cX3 = np.average(box[:, 0])
            cY3 = np.average(box[:, 1])
            # si este es el primer contorno que estamos examinando (es decir, el contorno más a la izquierda)
            # asumimos que este es el objeto de referencia
            if refObj is None:
                # descomprima el cuadro delimitador ordenado, luego calcule el punto medio entre los puntos superior izquierdo
                # y superior derecho, seguido del punto medio entre la parte superior derecha y abajo a la derecha
                (tl, tr, br, bl) = box
                (tlblX, tlblY) = midpoint(tl, bl)
                (trbrX, trbrY) = midpoint(tr, br)
                # calcular la distancia euclidiana entre los puntos medios,luego construye el objeto de referencia
                D = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))
                refObj = (box, (cX, cY), D / 1.9685)
                continue
            if refObj2 is None:
                # descomprima el cuadro delimitador ordenado, luego calcule el punto medio entre los puntos superior izquierdo
                # y superior derecho, seguido del punto medio entre la parte superior derecha y abajo a la derecha
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 43

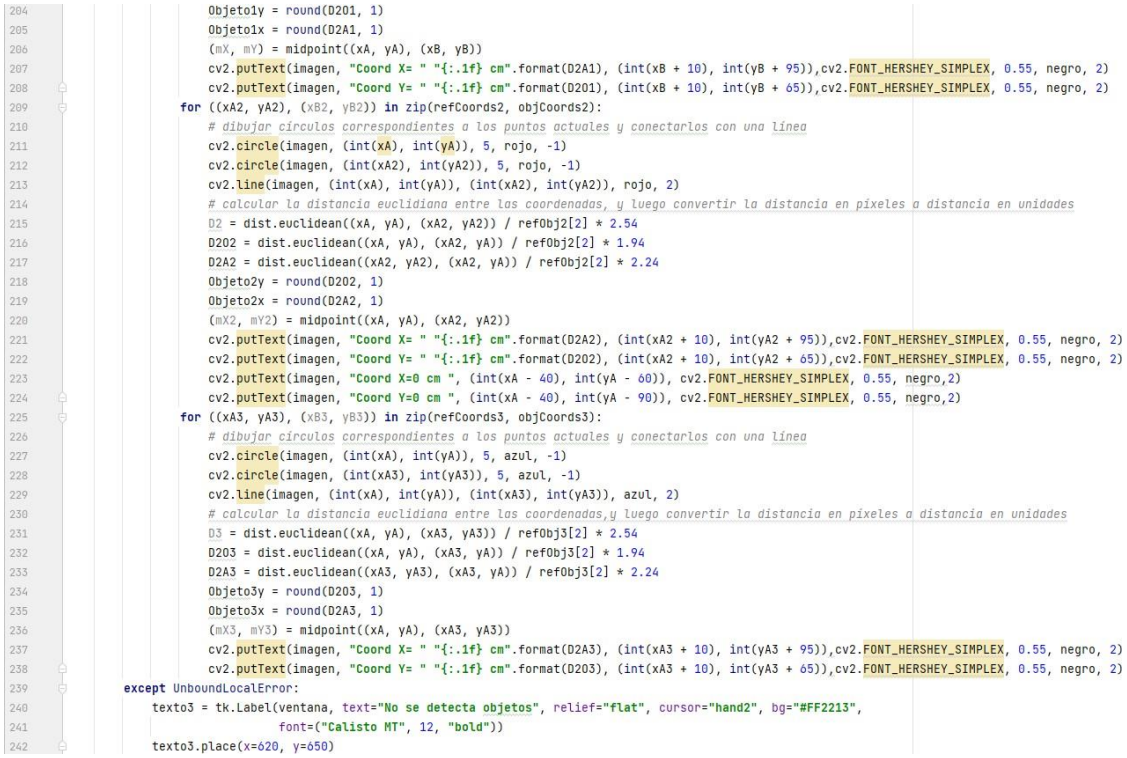
Programación principal de la herramienta tecnológica parte 5

```
            (tl2, tr2, br2, bl2) = box
            (tlblX2, tlblY2) = midpoint(tl2, bl2)
            (trbrX2, trbrY2) = midpoint(tr2, br2)
            # calcular la distancia euclidiana entre los puntos medios, luego construye el objeto de referencia
            D2 = dist.euclidean((tlblX2, tlblY2), (trbrX2, trbrY2))
            refObj2 = (box, (cX2, cY2), D / 1.9685)
            continue
        cv2.drawContours(imagen, [box.astype("int")], -1, (0, 255, 0), 2)
        if refObj3 is None:
            # descomprima el cuadro delimitador ordenado, luego calcule el punto medio entre los puntos superior izquierdo
            # y superior derecho, seguido del punto medio entre la parte superior derecha y abajo a la derecha
            (tl3, tr3, br3, bl3) = box
            (tlblX3, tlblY3) = midpoint(tl3, bl3)
            (trbrX3, trbrY3) = midpoint(tr3, br3)
            # calcular la distancia euclidiana entre los puntos medios, luego construye el objeto de referencia
            D3 = dist.euclidean((tlblX3, tlblY3), (trbrX3, trbrY3))
            refObj3 = (box, (cX, cY), D / 1.9685)
            continue
        # dibujar los contornos en la imagen
        cv2.drawContours(imagen, [box.astype("int")], -1, (0, 255, 0), 2)
        cv2.drawContours(imagen, [refObj2[0].astype("int")], -1, (0, 255, 0), 2)
        # apilar las coordenadas de referencia y las coordenadas del objeto para incluir el centro del objeto
        refCoords = np.vstack([refObj[1]])
        objCoords = np.vstack([(cX, cY)])
        refCoords2 = np.vstack([refObj2[1]])
        objCoords2 = np.vstack([(cX2, cY2)])
        refCoords3 = np.vstack([refObj3[1]])
        objCoords3 = np.vstack([(cX3, cY3)])
        # bucle sobre los puntos originales
        for ((xA, yA), (xB, yB), color) in zip(refCoords, objCoords, colors):
            # dibujar círculos correspondientes a los puntos actuales y
            # conectarlos con una línea
            cv2.circle(imagen, (int(xA), int(yA)), 5, cian, -1)
            cv2.circle(imagen, (int(xB), int(yB)), 5, cian, -1)
            cv2.line(imagen, (int(xA), int(yA)), (int(xB), int(yB)),cian, 2)
            # calcular la distancia euclidiana entre las coordenadas, y luego convertir la distancia en píxeles a distancia en unidades
            D = dist.euclidean((xA, yA), (xB, yB)) / refObj[2] * 2.54
            D2O1 = dist.euclidean((xA, yA), (xB, yA)) / refObj[2] * 1.94
            D2A1 = dist.euclidean((xB, yB), (xB, yA)) / refObj[2] * 2.24
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 44

Programación principal de la herramienta tecnológica partee 6



```
                Objeto1y = round(D2O1, 1)
                Objeto1x = round(D2A1, 1)
                (mX, mY) = midpoint((xA, yA), (xB, yB))
                cv2.putText(imagen, "Coord X= " "{:.1f} cm".format(D2A1), (int(xB + 10), int(yB + 95)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
                cv2.putText(imagen, "Coord Y= " "{:.1f} cm".format(D2O1), (int(xB + 10), int(yB + 65)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
            for ((xA2, yA2), (xB2, yB2)) in zip(refCoords2, objCoords2):
                # dibujar circulos correspondientes a los puntos actuales y conectarlos con una linea
                cv2.circle(imagen, (int(xA), int(yA)), 5, rojo, -1)
                cv2.circle(imagen, (int(xA2), int(yA2)), 5, rojo, -1)
                cv2.line(imagen, (int(xA), int(yA)), (int(xA2), int(yA2)), rojo, 2)
                # calcular la distancia euclidiana entre las coordenadas, y luego convertir la distancia en pixeles a distancia en unidades
                D2 = dist.euclidean((xA, yA), (xA2, yA2)) / refObj2[2] * 2.54
                D2O2 = dist.euclidean((xA, yA), (xA2, yA)) / refObj2[2] * 1.94
                D2A2 = dist.euclidean((xA2, yA2), (xA2, yA)) / refObj2[2] * 2.24
                Objeto2y = round(D2O2, 1)
                Objeto2x = round(D2A2, 1)
                (mX2, mY2) = midpoint((xA, yA), (xA2, yA2))
                cv2.putText(imagen, "Coord X= " "{:.1f} cm".format(D2A2), (int(xA2 + 10), int(yA2 + 95)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
                cv2.putText(imagen, "Coord Y= " "{:.1f} cm".format(D2O2), (int(xA2 + 10), int(yA2 + 65)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
                cv2.putText(imagen, "Coord X=0 cm ", (int(xA - 40), int(yA - 60)), cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro,2)
                cv2.putText(imagen, "Coord Y=0 cm ", (int(xA - 40), int(yA - 90)), cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro,2)
            for ((xA3, yA3), (xB3, yB3)) in zip(refCoords3, objCoords3):
                # dibujar circulos correspondientes a los puntos actuales y conectarlos con una linea
                cv2.circle(imagen, (int(xA), int(yA)), 5, azul, -1)
                cv2.circle(imagen, (int(xA3), int(yA3)), 5, azul, -1)
                cv2.line(imagen, (int(xA), int(yA)), (int(xA3), int(yA3)), azul, 2)
                # calcular la distancia euclidiana entre las coordenadas,y luego convertir la distancia en pixeles a distancia en unidades
                D3 = dist.euclidean((xA, yA), (xA3, yA3)) / refObj3[2] * 2.54
                D2O3 = dist.euclidean((xA, yA), (xA3, yA)) / refObj3[2] * 1.94
                D2A3 = dist.euclidean((xA3, yA3), (xA3, yA)) / refObj3[2] * 2.24
                Objeto3y = round(D2O3, 1)
                Objeto3x = round(D2A3, 1)
                (mX3, mY3) = midpoint((xA, yA), (xA3, yA3))
                cv2.putText(imagen, "Coord X= " "{:.1f} cm".format(D2A3), (int(xA3 + 10), int(yA3 + 95)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
                cv2.putText(imagen, "Coord Y= " "{:.1f} cm".format(D2O3), (int(xA3 + 10), int(yA3 + 65)),cv2.FONT_HERSHEY_SIMPLEX, 0.55, negro, 2)
        except UnboundLocalError:
            texto3 = tk.Label(ventana, text="No se detecta objetos", relief="flat", cursor="hand2", bg="#FF2213",
                              font=("Calisto MT", 12, "bold"))
            texto3.place(x=620, y=650)
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 45

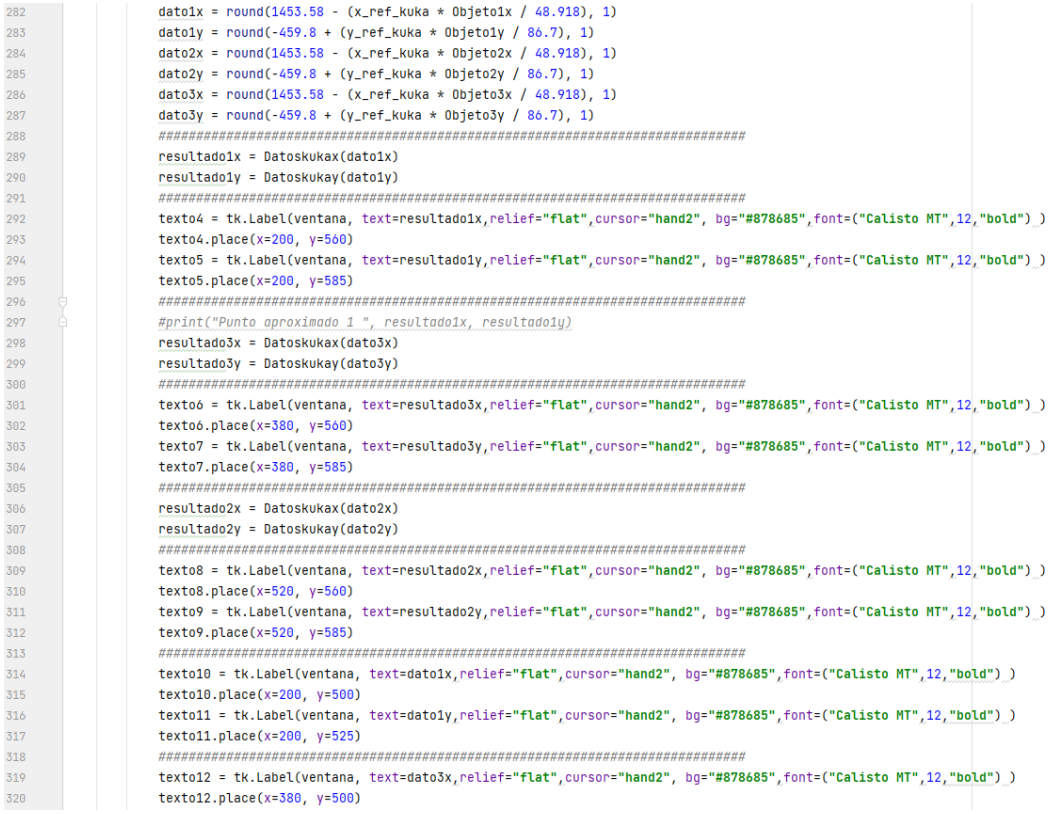
Programación principal de la herramienta tecnológica parte 7

```
    except ValueError:
        a=1
        if a == 1:
            messagebox.showwarning("Advertencia","Cámara obstruida")
            a==0
            return quitar()
    try:
        imagen1 = Image.fromarray(imagen)
        image = ImageTk.PhotoImage(image=imagen1)
        etiqueta_video.configure(image=image)
        etiqueta_video.image = image
        etiqueta_video.after(10, iniciar)
        Area1 = ''.join(map(str, detecciones[1]))
        area1 = float(Area1)
        Area2 = ''.join(map(str, detecciones[2]))
        area2 = float(Area2)
        Area3 = ''.join(map(str, detecciones[3]))
        area3 = float(Area3)
        envio_tamaño1 = clasificacion(area3)
        texto1 = tk.Label(ventana, text=envio_tamaño1,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto1.place(x=200, y=700)
        envio_tamaño3 = clasificacion(area2)
        texto2 = tk.Label(ventana, text=envio_tamaño3,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto2.place(x=380, y=700)
        envio_tamaño2 = clasificacion(area1)
        texto3 = tk.Label(ventana, text=envio_tamaño2,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=520, y=700)
        texto3 = tk.Label(ventana, text=Objeto1x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=200, y=630)
        texto3 = tk.Label(ventana, text=Objeto3x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=380, y=630)
        texto3 = tk.Label(ventana, text=Objeto2x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=520, y=630)
        texto3 = tk.Label(ventana, text=Objeto1y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=200, y=650)
        texto3 = tk.Label(ventana, text=Objeto3y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=380, y=650)
        texto3 = tk.Label(ventana, text=Objeto2y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto3.place(x=520, y=650)
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 46

Programación principal de la herramienta tecnológica parte 8

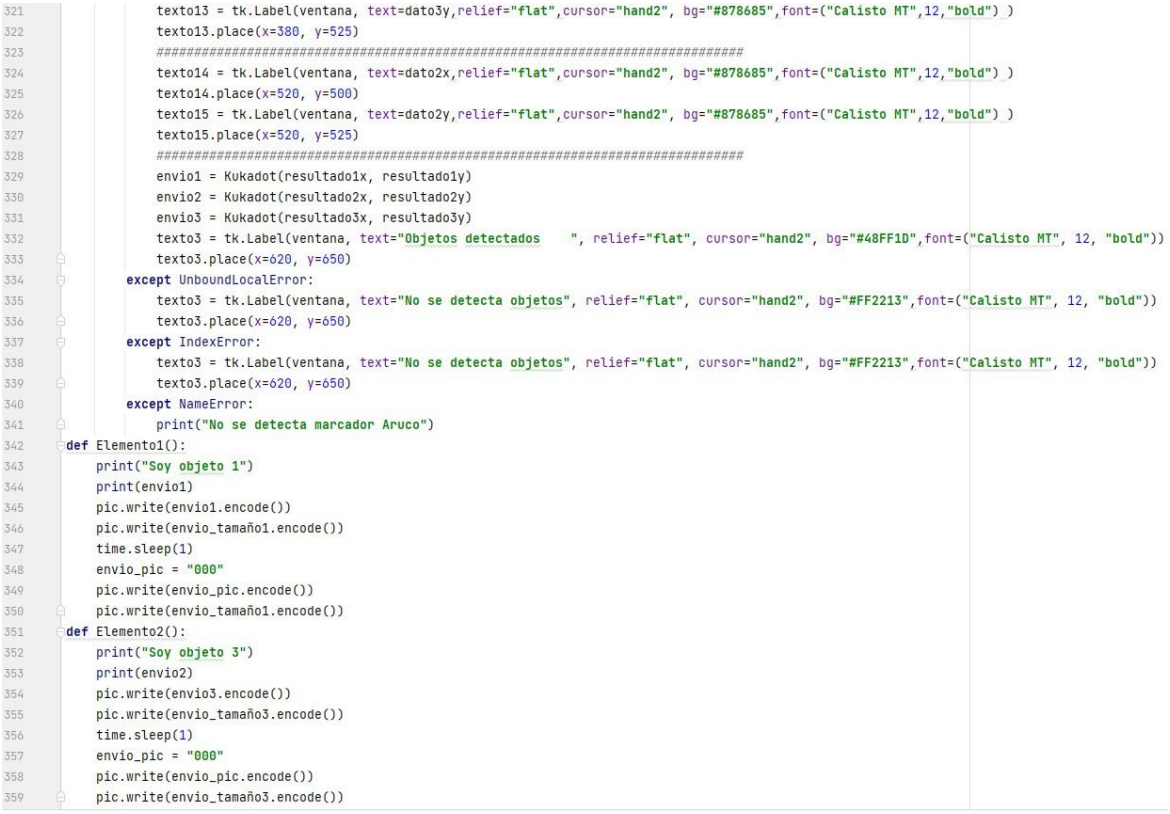


```
        dato1x = round(1453.58 - (x_ref_kuka * Objeto1x / 48.918), 1)
        dato1y = round(-459.8 + (y_ref_kuka * Objeto1y / 86.7), 1)
        dato2x = round(1453.58 - (x_ref_kuka * Objeto2x / 48.918), 1)
        dato2y = round(-459.8 + (y_ref_kuka * Objeto2y / 86.7), 1)
        dato3x = round(1453.58 - (x_ref_kuka * Objeto3x / 48.918), 1)
        dato3y = round(-459.8 + (y_ref_kuka * Objeto3y / 86.7), 1)
        ##########################################################################
        resultado1x = Datoskukax(dato1x)
        resultado1y = Datoskukay(dato1y)
        ##########################################################################
        texto4 = tk.Label(ventana, text=resultado1x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto4.place(x=200, y=560)
        texto5 = tk.Label(ventana, text=resultado1y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto5.place(x=200, y=585)
        ##########################################################################
        #print("Punto aproximado 1 ", resultado1x, resultado1y)
        resultado3x = Datoskukax(dato3x)
        resultado3y = Datoskukay(dato3y)
        ##########################################################################
        texto6 = tk.Label(ventana, text=resultado3x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto6.place(x=380, y=560)
        texto7 = tk.Label(ventana, text=resultado3y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto7.place(x=380, y=585)
        ##########################################################################
        resultado2x = Datoskukax(dato2x)
        resultado2y = Datoskukay(dato2y)
        ##########################################################################
        texto8 = tk.Label(ventana, text=resultado2x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto8.place(x=520, y=560)
        texto9 = tk.Label(ventana, text=resultado2y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto9.place(x=520, y=585)
        ##########################################################################
        texto10 = tk.Label(ventana, text=dato1x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto10.place(x=200, y=500)
        texto11 = tk.Label(ventana, text=dato1y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto11.place(x=200, y=525)
        ##########################################################################
        texto12 = tk.Label(ventana, text=dato3x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
        texto12.place(x=380, y=500)
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 47

Programación principal de la herramienta tecnológica parte 9



```
            texto13 = tk.Label(ventana, text=dato3y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
            texto13.place(x=380, y=525)
            ##############################################################################
            texto14 = tk.Label(ventana, text=dato2x,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
            texto14.place(x=520, y=500)
            texto15 = tk.Label(ventana, text=dato2y,relief="flat",cursor="hand2", bg="#878685",font=("Calisto MT",12,"bold") )
            texto15.place(x=520, y=525)
            ##############################################################################
            envio1 = Kukadot(resultado1x, resultado1y)
            envio2 = Kukadot(resultado2x, resultado2y)
            envio3 = Kukadot(resultado3x, resultado3y)
            texto3 = tk.Label(ventana, text="Objetos detectados    ", relief="flat", cursor="hand2", bg="#48FF1D",font=("Calisto MT", 12, "bold"))
            texto3.place(x=620, y=650)
        except UnboundLocalError:
            texto3 = tk.Label(ventana, text="No se detecta objetos", relief="flat", cursor="hand2", bg="#FF2213",font=("Calisto MT", 12, "bold"))
            texto3.place(x=620, y=650)
        except IndexError:
            texto3 = tk.Label(ventana, text="No se detecta objetos", relief="flat", cursor="hand2", bg="#FF2213",font=("Calisto MT", 12, "bold"))
            texto3.place(x=620, y=650)
        except NameError:
            print("No se detecta marcador Aruco")
def Elemento1():
    print("Soy objeto 1")
    print(envio1)
    pic.write(envio1.encode())
    pic.write(envio_tamaño1.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño1.encode())
def Elemento2():
    print("Soy objeto 3")
    print(envio2)
    pic.write(envio3.encode())
    pic.write(envio_tamaño3.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño3.encode())
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 48

Programación principal de la herramienta tecnológica parte 10

```
def Elemento3():
    print("Soy objeto 2")
    print(envio3)
    pic.write(envio2.encode())
    pic.write(envio_tamaño2.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño2.encode())
def automatico():
    pic.write(envio1.encode())
    pic.write(envio_tamaño1.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño1.encode())
    time.sleep(9)
    pic.write(envio3.encode())
    pic.write(envio_tamaño3.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño3.encode())
    time.sleep(9)
    pic.write(envio2.encode())
    pic.write(envio_tamaño2.encode())
    time.sleep(1)
    envio_pic = "000"
    pic.write(envio_pic.encode())
    pic.write(envio_tamaño2.encode())
    quitar()
def quitar():
    global cap
    texto3 = tk.Label(ventana, text="No se detecta objetos", relief="flat", cursor="hand2", bg="#FF2213",font=("Calisto MT", 12, "bold"))
    texto3.place(x=620, y=650)
    texto1 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto1.place(x=200, y=700)
    texto2 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 49

Programación principal de la herramienta tecnológica parte 11

```
    texto2.place(x=380, y=700)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=520, y=700)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=200, y=630)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=380, y=630)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=520, y=630)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=200, y=650)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=380, y=650)
    texto3 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto3.place(x=520, y=650)
    texto4 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto4.place(x=200, y=560)
    texto5 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto5.place(x=200, y=585)
    texto6 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto6.place(x=380, y=560)
    texto7 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto7.place(x=380, y=585)
    texto8 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto8.place(x=520, y=560)
    texto9 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto9.place(x=520, y=585)
    texto10 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto10.place(x=200, y=500)
    texto11 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto11.place(x=200, y=525)
    texto12 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto12.place(x=380, y=500)
    texto13 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto13.place(x=380, y=525)
    texto14 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto14.place(x=520, y=500)
    texto15 = tk.Label(ventana, text="               ", relief="flat", cursor="hand2", bg="#878685",font=("Calisto MT", 12, "bold"))
    texto15.place(x=520, y=525)
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.

Figura 50

Programación principal de la herramienta tecnológica parte 12

```
    etiqueta_video.place_forget()
    cap.release()
boton=tk.Button(ventana, text="Iniciar video", bg="#497174", relief="flat",cursor="hand2",command=iniciar_video,width=13,height=2,font=("Calisto MT",12,"bold"))
boton.place(x=815, y=536)
botont1 = tk.Button(ventana, text="Objeto 1",bg="#497174", relief="flat", cursor="hand2", command=Elemento1,width=13, height=2, font=("Calisto MT", 12, "bold"))
botont1.place(x=1057, y=46)
botont2 = tk.Button(ventana,text="Objeto 2", bg="#497174", relief="flat", cursor="hand2", command=Elemento2,width=13, height=2, font=("Calisto MT", 12, "bold"))
botont2.place(x=1057, y=143)
botont3 = tk.Button(ventana,text="Objeto 3", bg="#497174", relief="flat", cursor="hand2", command=Elemento3,width=13, height=2, font=("Calisto MT", 12, "bold"))
botont3.place(x=1057, y=240)
botont3 = tk.Button(ventana,text="Automatico", bg="#497174", relief="flat", cursor="hand2", command=automatico,width=13, height=2, font=("Calisto MT", 12, "bold"))
botont3.place(x=1057, y=373)
boton2=tk.Button(ventana, text="Quitar video", bg="#497174", relief="flat",cursor="hand2",command=quitar,width=13,height=2,font=("Calisto MT",12,"bold"))
boton2.place(x=815, y=647)
etiqueta_video=tk.Label(ventana, bg="black")
etiqueta_video.place(x=30, y=24)
ventana.mainloop()
```

Nota: Se aprecia la programación realizada para la identificación y clasificación de objetos.