2019

VISIÓN ARTIFICIAL ORIENTADA A LA DETECCIÓN DE PIEZAS EN MOVIMIENTO PARA SU CLASIFICACIÓN

MARTÍNEZ NAVARRO, FRANCISCO JAVIER

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE CONCEPCIÓN – REY BALDUINO DE BELGICA

VISIÓN ARTIFICIAL ORIENTADA A LA DETECCIÓN DE PIEZAS EN MOVIMIENTO PARA SU CLASIFICACIÓN.

Trabajo de Titulación para optar al Título de Ingeniero de Ejecución en Control e Instrumentación Industrial

Alumnos:

Francisco Javier Martínez Navarro

Juan Pablo Campos Robles

Profesor Guía:

Sr. Rodrigo Méndez

Profesor Co-referente:

Sr. Felipe Benavides

2019

RESUMEN

En la actualidad es común que los procesos industriales incorporen sistemas automatizados en sus procesos, ya sea para obtener productos de mayor calidad o como para obtener mayores volúmenes de producción a un menor costo.

La Universidad Técnica Federico Santa María sede Concepción, actualmente cuenta con un brazo robótico articulado, en el laboratorio de celda flexible, el cual está compuesto por diferentes estaciones de trabajo automatizadas, las cuales se asemejan bastante a procesos industriales existentes hoy en día. Por esta razón y debido a la gran cantidad de prestaciones en las que se puede desarrollar un robot en la actualidad, además del avance continuo de la tecnología en el tiempo, es posible sumarle a procesos ya establecidos mejoras significativas como lo es la visión artificial, la cual puede generar mejoras significativas de la calidad.

Por esta razón que el siguiente proyecto de título, muestra la integración de un sistema de visión artificial y un brazo robótico articulado, utilizando programación Python. Para llevar a cabo este trabajo de título se utilizó una Raspberry Pi 3 modelo B +, la cual mediante un código de programación permitirá realizar un reconocimiento de diferentes tipos de figuras y formas. Además consta con una cámara la cual se encarga de grabar y enviar la información a la Raspberry de las diferentes tipos de figuras que circulan por una cinta en movimiento, generando un reconocimiento y procesamiento de imágenes, que posteriormente permitirán interactuar con el brazo robótico Mitsubishi Melfa RV-2SDB, cuya función será el posicionamiento, la selección, el traslado y clasificación de piezas por colores y formas, conformando así un proceso industrial de clasificación de productos.

ÍNDICE

ÍNDICE DE FIGURAS v
ÍNDICE DE TABLAS vii
SIGLA viii
SIMBOLOGÍA ix
INTRODUCCIÓN 1
OBJETIVO GENERAL 2
OBJETIVOS ESPECÍFICOS 2
ALCANCES Y LIMITACIONES 2
CAPÍTULO 1: DEFINICIÓN Y CONCEPTOS GENERALES 3
1.1 VISIÓN ARTIFICIAL 4
1.2 OPENCV (THE OPEN COMPUTER VISION LIBRARY) 4
1.3 PYTHON 5
1.4 CIROS STUDIO 5
1.5 RASPBERRY PI 3 B+ 5
1.5.1 Características de la Raspberry Pi 3 B+ 6
1.6 CÁMARA RASPBERRY PI V2. 7
1.6.1 Especificaciones técnicas: 7
1.7 MÓDULOS DE RELÉ RASPBERRY 7
1.8 BRAZO ROBÓTICO 8
1.8.1 Robot "MELFA - RV-2SDB" 9
1.8.2 Aplicación en Procesos Productivos 9
1.8.3 Integración Sencilla en Sistemas 10
1.8.4 Herramientas de Programación 10
1.8.4.1 RT toolbox2 11
1.8.4.2 Melfa-visión 11
1.8.4.3 Melfa-trabajos 11
1.8.5 Controlador CR1DA-771 12
1.8.6 Consola Portátil R32TB 16
CAPÍTULO 2: PROGRAMACIÓN, INSTALACIÓN E INTEGRACION DE SOFTWARE 19
2.1 INSTALACIÓN DE OPENCV EN RASPBERRY PI 3 B+ 20
2.1.1 Paso 1: Expandir el sistema de archivos. 20
2.1.2 Paso 2: Instalar Dependencias. 21
2.1.3 Paso 3: Descargar el código fuente de OpenCV. 22
2.1.4 Paso 4: Python 3 22
2.1.5 Paso 5: Finalizar la Instalación de OpenCV para Python 3 en la Raspberry Pi. 26
2.2 APLICACIÓN DE VISION ARTIFICIAL: ETAPAS DE PROCESAMIENTO DE IMÁGENES 26
2.2.1 Etapas de una Aplicación de Visión por Computador 26
2.2.2 Procesamiento de Imágenes 27
2.2.3 Espacios de Color 28
2.2.3.1 Espacio HSV 28
2.2.3.1.1 HUE – Tono o Matiz 29
2.2.3.1.2 Saturation – Saturación 29
2.2.3.1.3 Value – Valor 30

2.2.4 Creación de máscara de color 30
2.2.5 Procesamiento de imágenes binarias..... 31
2.2.5.1 Transformaciones Morfológicas 32
2.2.5.2 Apertura 32
2.2.5.3 Cierre 33
2.2.6 Detección de bordes 33
2.2.7 Filtrado del ruido en una imagen 34
2.2.8 Aplicación de filtro Gaussiano para la eliminación de ruido en imágenes 35
2.2.9 Detector de bordes Canny 38
2.2.9.1 Detección de bordes con Sobel..... 39
2.2.9.2 Filtrado de bordes mediante la supresión non-maximun 40
2.2.9.3 Aplicar umbral por histéresis..... 40
2.2.9.4 Detector de bordes Canny con OpenCV 40
2.2.10 Buscar los contornos de una imagen 41
2.2.11 Dibujar los contornos en una imagen con OpenCV 43
2.3 PROGRAMACIÓN PYTHON 43
2.4 PUERTOS GPIO 44
2.4.1 Función de Pines por Color 45
2.5 PROGRAMACIÓN DE BRAZO ROBÓTICO..... 47
2.5.1 Programación con Melfa Basic V 47
2.5.2 Declaración de variables..... 48
2.5.3 Funciones de movimiento 48
2.5.4 Definición de Velocidades 50
2.5.5 Control de Gripper..... 51
2.5.6 Comandos de control de programa 51
2.5.7 Paro Incondicional y retardos..... 53
2.5.8 Entradas y salidas 53
2.6 PROGRAMACIÓN DEL BRAZO ROBÓTICO CON CIROS STUDIO..... 54
2.6.1 Creación de un nuevo proyecto 54
2.6.2 Configuración de comunicación 56
2.6.3 Manipulación del brazo Robótico desde Ciros Studio..... 58
2.6.4 Monitoreo Software Ciros Studio 59
2.6.5 Monitoreo de Programa 60
2.6.6 Encendido de la unidad de control manual 62
2.6.7 Encendido de Servos y Enclavamiento de Control 63
2.6.8 Opciones del Menú y/o Pantalla..... 64
2.6.9 Operación del Brazo Robótico en Modo JOG. 65
2.6.10 Ajustes de Velocidad..... 66
2.6.11 Crear un Nuevo Programa o Editar uno Existente. 67
2.6.12 Guardar Posiciones desde el Teaching Box. 68
2.6.13 Ejecución de un Programa. 69
2.7 INTEGRACIÓN RASPBERRY Y BRAZO ROBÓTICO 71
3 CONCLUSIONES Y RECOMENDACIONES 73
4 BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN 74

ÍNDICE DE FIGURAS

Figura 1-1: Raspberry Pi 3 B+ 6
Figura 1-2: Camara Raspberry Pi 7
Figura 1-3: Módulos de relé Raspberry 8
Figura 1-4: Brazo robótico Melfa RV-2SDB. 9
Figura 1-5: Integración de Sistemas 10
Figura 1-6: Figura: Diseño Estructural del Brazo Robótico. 11
Figura 1-7: Controlador CR1DA-771. 12
Figura 1-8: Consola Portátil y/o Teaching Box. 16
Figura 2-1: Seleccione el elemento " Opciones avanzadas del menú " 20
Figura 2-2: Expandiendo el sistema de archivos en Raspberry Pi 3. 20
Figura 2-3: Entorno Virtual. 24
Figura 2-4: Compilación Python 3. 24
Figura 2-5: Término de Compilación OpenCV 25
Figura 2-6: Etapas de una aplicación de visión artificial 27
Figura 2-7: Matriz G 27
Figura 2-8: Matriz R, B, G. 28
Figura 2-9: Espacio Color HSV 29
Figura 2-10: Variación del matiz en HSV con saturación y brillo máximo 29
Figura 2-11: Saturación en HSV con matiz en 0o y brillo máximo. 30
Figura 2-12: Brillo en HSV con saturación máxima y matiz de 90º 30
Figura 2-13: Ejemplo de aplicación de máscara de color 31
Figura 2-14: Imágenes de Proyecto realizado de dos mascaras para dos colores, verde y azul. 31
Figura 2-15: Ejemplo imagen binaria 32
Figura 2-16: Resultado apertura. 33
Figura 2-17: Resultado Cierre 33
Figura 2-18: Tipos de bordes descritos en una dimensión 34
Figura 2-19: Ejemplo mascara o kernel de 3x3 y 4x4 35
Figura 2-20: Función Gaussiana 36
Figura 2-21: Ejemplo Filtro Gaussiano 37
Figura 2-22: Detección de bordes de monedas. 38
Figura 2-23: Diferencia de cambio de luz en borde 39
Figura 2-24: Ejemplo de buscador de contornos. 41
Figura 2-25: Diagrama de flujo programación Python. 44
Figura 2-26: Pines GPIO en Raspberry 45
Figura 2-27: Pines GPIO para Raspberry Pi 3 Modelo B+ 46
Figura 2-28: Pines GPIO 47
Figura 2-29: Movimiento de interpolación de ejes 49
Figura 2-30: Movimiento en interpolación lineal 50
Figura 2-31: Crear un nuevo Proyecto 54
Figura 2-32: Nombre del Nuevo proyecto 55
Figura 2-33: Selección del tipo de robot y lenguaje de programación. 56
Figura 2-34: Configuración de máscara de red 56
Figura 2-35: Cambio de las propiedades de conexión con el robot 57
Figura 2-36: Configuración de la conexión con el robot 57

Figura 2-37: Ventana de información de conexión exitosa con el robot. 58
Figura 2-38: Ventana de Operación de Marcha a Impulsos. 59
Figura 2-39: Opciones de Monitoreo 59
Figura 2-40: Depurador de programa en línea y activación de servomotores del robot 60
Figura 2-41: Monitor de entradas y salidas 61
Figura 2-42: Encendido de la unidad de programación manual. 62
Figura 2-43: Selección de modo Automatico/Manual y Habilitación teaching box.. 63
Figura 2-44: Enclavamiento de control para encendido de servos. 64
Figura 2-45: Menú de la consola portátil. 65
Figura 2-46: Operación en modo JOG. 65
Figura 2-47: Ajustes de Velocidades de trabajo del Robot. 67
Figura 2-48: Crear un Nuevo Programa o Editar uno Existente. 67
Figura 2-49: Ingresar el nombre del programa. 68
Figura 2-50: Edición de Posiciones desde consola portátil. 68
Figura 2-51: Guardar posiciones desde consola portátil. 69
Figura 2-52: Cambio de modo de la unidad de control 69
Figura 2-53: Diagrama de flujo funcionamiento programa brazo robotico 70
Figura 3-1: Integracion Raspberry brazo robótico 71
Figura 3-2: Diagrama de flujo integracion Raspberry Brazo Robotico 72

ÍNDICE DE TABLAS

Tabla 1-1: Modelo y Características del Brazo..... 12
Tabla 1-2: Especificaciones del controlador. 15
Tabla 1-3: Especificaciones del Controlador..... 18
Tabla 2-1: Descripción de puertos GPIO por colores. 45
Tabla 2-2: Ejemplo de movimientos de interpolación de ejes..... 49
Tabla 2-3: Ejemplo de movimientos de interpolación lineal. 50
Tabla 2-4: Ejemplo de ajustes de velocidad de trabajo del robot..... 51
Tabla 2-5: Ejemplo de Abrir y Cerrar Gripper. 51
Tabla 2-6: Ejemplo On Goto (Salto Condicional a Linea Designada)..... 51
Tabla 2-7: Ejemplo On Goto (Salto Condicional a Linea Designada)..... 52
Tabla 2-8: Ejemplo ciclo If Then Else..... 52
Tabla 2-9: Ejemplo de Select Case (Salto Condicional). 52
Tabla 2-10: Ejemplo ciclo While..... 53
Tabla 2-11: Ejemplo de paro Incondicional 53
Tabla 2-12: Ejemplos de entradas y salidas 54

SIGLA

CSI	:	Camera Serial Interface (Interfaz Serie Para Cámaras)
DSI	:	Display Serial Interface (Interfaz Serial de Pantalla)
GOT	:	Terminal Grafica de Operaciones
GPIO	:	General Purpose Input/Output (Entradas y Salidas de Propósito General)
HDMI	:	High-Definition Multimedia Interface (Interfaz Multimedia de alta Definición)
HSV	:	Hue, Saturation, Value (Tono o matiz, Saturación, Brillo o Valor)
IMD	:	Introducción Manual de Datos
I2C	:	Inter Integrated Circuits (Circuito Inter-integrado)
LCD	:	Liquid Cristal Display (Pantalla de Cristal Líquido)
OPENCV	:	Open Computer Vision Library (Librería de Visión por Computador)
RGB	:	Red, Green, Blue (Rojo, Verde, Azul), Matriz de una imagen
RV-2SDB	:	RV: Robot de Brazo Articulado Vertical. 2: Capacidad de Carga en Kilos. SD: Serie D B: Break, todos los ejes provistos de frenos
RXD	:	Recibir Datos (Comunicación Serial)
SSCNET	:	Servo System Controller Network
TXD	:	Trasmitir Datos (Comunicación Serial)
UART	:	Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal)
USB	:	Universal Serial Bus (Bus Universal en Serie)
VAC	:	Voltage of Alternate Current (Voltage de Corriente Continua)
Vdc	:	Voltage of Continuos Current (Voltage de Corriente Continua)
%	:	Porcentaje

SIMBOLOGÍA

A	:	Ampere
kg	:	Kilo gramo
mA	:	Mili Amper
Mbps	:	Megabit por Segundo
mm	:	Milímetros
mm/s	:	Milímetros por segundo
º/seg	:	Grados por segundo
σ	:	Sigma

INTRODUCCIÓN

El presente trabajo de título está orientado a un proceso de automatización a través de visión artificial, en el cual se detalla cuáles son los campos de aplicación, que tipos de procesos podemos controlar y cuáles son sus ventajas y desventajas. Todo lo anterior está acompañado de una investigación, en la cual se realizó un estudio para lograr la interacción y/o comunicación de visión artificial con un brazo robótico utilizando diferentes tipos de software y controladores.

El Objetivo final de este proyecto está basado en la extracción automática de imágenes mediante una cámara conectada a una Raspberry pi 3, la que será la encargada de realizar el procesamiento de las imágenes a través del lenguaje de programación Python utilizando la librería de OpenCV. Una vez realizado el análisis el brazo robótico será el encargado de realizar la clasificación de cada una de las piezas detectadas, ya sea por color o forma en el lugar establecido para ello.

Dicho todo lo anterior podemos decir que visión artificial es una disciplina que comprende diversos ámbitos, pero principalmente está basada en la extracción, procesamiento y análisis de imágenes, cuyo objetivo final es mediante el uso de las nuevas tecnologías disponibles lograr que una máquina pueda ver, procesar y tomar decisiones de la misma forma que lo realizan los seres humanos. Para lograr que un robot realice movimientos similares a los de una persona debemos hacerlo a través de coordenadas las cuales deben ser ingresadas en su código de programación, incluyendo todas las condiciones de seguridad posibles, para evitar posibles colisiones, daños a personas y a estructuras.

Una vez realizado el ingreso de coordenadas y configurado las condiciones de seguridad del proceso, se debe realizar un análisis para determinar y desarrollar un algoritmo y/o código de programación en donde se indique lo que se desea realizar en el proceso de automatización, para ello se realizó un estudio en profundidad de los diferentes tipos de lenguajes de programación que contempla dicho proyecto.

OBJETIVO GENERAL

Automatizar el proceso de detección, ubicación y clasificación de objetos o piezas, por color y forma desde una correa transportadora en movimiento mediante visión artificial.

OBJETIVOS ESPECÍFICOS

- Estudiar las características principales de la Raspberry PI y el brazo robótico Melfa RV-2SDB
- Programar la Raspberry PI para hacer un reconocimiento de color y ubicación mediante visión artificial
- Establecer la comunicación entre la Raspberry PI y el brazo robótico
- Programar el sistema integrado para el almacenamiento selectivo.

ALCANCES Y LIMITACIONES

- La detección de los colores y formas se realiza mediante una cámara modelo PI Camera V2, conectada a una Raspberry PI Modelo 3 B.
- La programación se realiza mediante el lenguaje de programación Python.
- La detección y selección de piezas será sobre una correa en movimiento.
- La clasificación de las piezas se realizara mediante un brazo robótico modelo Melfa RV-2SDB
- Las pruebas con el brazo robótico solo se podrán realizar en laboratorio de la USM sede Concepción.
- Delay en las respuesta del controlador del brazo robótico
- Cambio de iluminación ambiental para reconocimiento de color

CAPÍTULO 1: DEFINICIÓN Y CONCEPTOS GENERALES

1.1 VISIÓN ARTIFICIAL

Visión artificial es una disciplina científica mediante la que se puede adquirir, procesar y analizar imágenes, capturadas mediante hardware para ser analizadas y procesadas mediante software.

El procesamiento se consigue descomponiendo la imagen en pixeles para su posterior estudio.

Algunos de sus usos más característicos son:

- Control de calidad.
- Clasificación.
- Contaje de productos.
- Posicionamiento
- Control de rotación.
- Pick and place.

1.2 OPENCV (THE OPEN COMPUTER VISION LIBRARY)

OpenCV es una librería de visión artificial creada por Intel, su publicación es bajo licencia BSD que permite que sea utilizada libremente para propósitos comerciales y de investigación.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, extracción de modelos 3D, comparar imágenes, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios.

Se usa en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos, etc.

OpenCV está escrito en C++, tiene interfaces en C++, C, Python, Java y MATLAB interfaces y funciona en Windows, Linux, Android y Mac OS.

1.3 PYTHON

Python es un lenguaje simple usado en varias áreas de la tecnología, procesamiento de datos, web, redes, visión artificial, etc.

Es un lenguaje sencillo de usar y de código abierto, Python es un lenguaje de programación de scripting. Los lenguajes scripting son aquellos lenguajes que usan un intérprete en vez de ser compilados. La mayoría de usuarios de Python le consideran como un lenguaje más limpio y elegante a la hora de programar.

1.4 CIROS STUDIO

Ciros Studio es una herramienta de trabajo profesional, utilizada para crear modelos de simulación de diferentes tipos de procesos. Es una potente plataforma de desarrollo para uso industrial, ya que unifica simulación, modelización y programación en una interfaz común.

1.5 RASPBERRY PI 3 B+

Raspberry Pi modelo 3 B+ es una placa computadora de tamaño reducido, computador de placa única u computador de placa simple de bajo costo y de libre utilización, desarrollado en el reino unido por la fundación Raspberry Pi en el 2011, con el objetivo de estimular la enseñanza de la informática en las escuelas. Su comercialización empezó en el año 2012.

El concepto es el de un computador desnudo de todos los accesorios que se pueden eliminar sin que afecte al funcionamiento básico. Está formada por una placa que soporta varios componentes necesarios en un computador común y es capaz de comportarse como tal.

A la Raspberry Pi la han definido como una maravilla en miniatura, que guarda en su interior un importante poder de cómputo en un tamaño muy reducido, ya que es capaz de realizar cosas extraordinarias.

Figura 1-1: Raspberry Pi 3 B+

Fuente: www.muycomputer.com

1.5.1 Características de la Raspberry Pi 3 B+

El modelo de la Raspberry Pi 3 B+ dispone de las siguientes características principales:

- Procesador **Broadcom BCM2837B0** de 1.4GHz quad-core ARM Cortex A53 de 64-bit.
- 1GB de memoria RAM soldada a la placa.
- Un puerto de Ethernet de 300 Mbps.
- Bluetooth 4.2.
- Conexión inalámbrica Wi-Fi 802.11 de doble banda (2,4 GHz y 5 GHz).
- Un Puerto HDMI.
- 4 puertos USB 2.0.
- 40 pines de entrada y salida general (Conector GPIO).
- Micro SD para memoria interna.
- Puerto y/o Conector CSI para conectar una cámara.
- Puerto DSI para conectar una pantalla táctil.
- Salida de audio estéreo y video compuesto.
- Power-over-Ethernet Poe.

1.6 CÁMARA RASPBERRY PI V2.

El módulo de cámara de alta definición ofrece una alta sensibilidad, baja diafonía y captura de imágenes de bajo ruido en un diseño ultra pequeño y liviano. El módulo de la cámara se conecta a la placa Raspberry Pi a través del conector CSI diseñado específicamente para la interfaz a las cámaras. El bus CSI es capaz de velocidades de datos extremadamente altas, y lleva exclusivamente datos de píxeles al procesador BCM2835.

El modulo Pi Camera V2 cuenta con un sensor 5 megapíxeles, lo sorprendente de los módulos de cámara Raspberry Pi es que no tiene que usar ningún algoritmo o software sofisticado para capturar y procesar imágenes. La Fundación Raspberry Pi creó una biblioteca de Python para manejar todo eso, e incluso incluye herramientas de procesamiento como filtros, rotación y recorte.

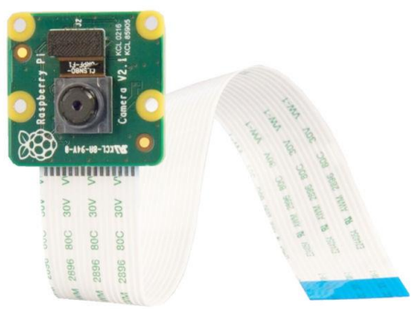

Figura 1-2: Camara Raspberry Pi

Fuente: www.pcfactory.cl

1.6.1 Especificaciones técnicas:

- Peso: 3g
- 25 mm x 23 mm x 9 mm
- Sensor de 5 megapíxeles

1.7 MÓDULOS DE RELÉ RASPBERRY

Estos módulos de relé están formados por una bobina que al paso de corriente crea un campo magnético que atrae una pieza metálica, generando que sus contactos se abran o se cierren. Esto permite la interacción de dos circuitos, de diferentes tensiones. En este caso la señal eléctrica que energiza la bobina del relé, es enviada

a través de los puertos GPIO de la Raspberry (3,3 Vdc), y la tensión que alimenta el común de los contactos del relé es 24 Vdc. Al cerrarse los contactos normalmente abiertos permiten el envío de una señal 24 volts a las entradas del controlador del brazo, seleccionando la programación y movimientos para los cuales fue configurado el robot.

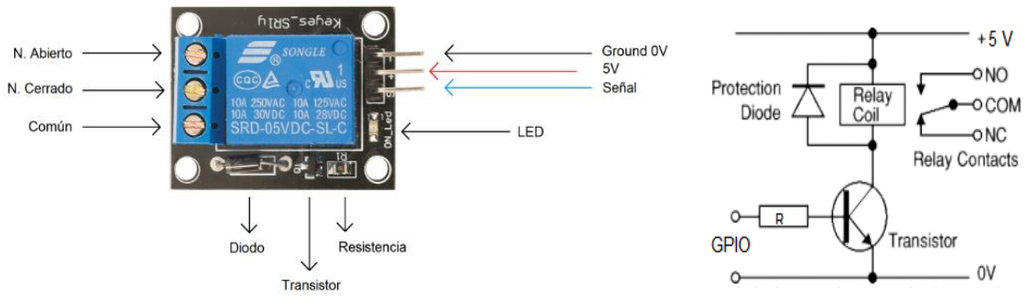

Figura 1-3: Módulos de relé Raspberry

Fuente: www.josehervas.es

El circuito del módulo de relés está diseñado para la conmutación de cargas de potencia. Los contactos del relé soportan cargas de hasta 10A y 250 Vac (30 Vdc), aunque es recomendable usar niveles de tensión por debajo de estos límites. Otra característica importante es que las entradas de control se encuentran aisladas con opto acopladores para minimizar el ruido percibido por el circuito de control mientras se realiza la conmutación de la carga.

1.8 BRAZO ROBÓTICO

El brazo robótico Mitsubishi es un tipo de brazo mecánico poliarticulado, compuesto y/o ensamblado por diferentes tipos de piezas mecánicas, las cuales permiten realizar todo tipo de movimientos, ya sean traslacionales, rotativos o desplazamientos lineales. Este robot cuenta con un potente controlador, el cual al ser programado puede manipular objetos y realizar trabajos que permitan la interacción con su entorno. Esto es posible debido al diseño constructivo de su brazo el cual tiene un ángulo de giro de ±240 grados, permitiendo aprovechar al máximo los accesos a su entorno. Además pueden realizar funciones de carácter repetitivo y mucho más rápido y preciso que los seres humanos, ya que cuentan con motores, los cuales crean y permiten los grados de libertad con la que se mueve esta máquina, simulando movimientos y funciones parecidas a las de un brazo humano.

Mientras más motores y articulaciones tengan el brazo, mayores serán los grados de libertad y movimientos.

1.8.1 Robot "MELFA - RV-2SDB"

El brazo robótico RV-2SDB, tiene una movilidad extraordinaria, debido a su diseño de construcción y ángulo de giro, de ±240 grados, permitiéndole realizar movimientos de diversa complejidad incluso en celdas y/o lugares de trabajo con espacios reducidos y apretados. Por esta razón es ideal en los lugares de procesos productivos en los que se cuenta con espacios muy pequeños, lo cual genera grandes ventajas en la parte económica, ya que se reducen considerablemente los costos, gracias al ahorro de material y al menor espacio necesario para la construcción de nuevos sistemas.

Figura 1-4: Brazo robótico Melfa RV-2SDB.

Fuente: www.esco.be

1.8.2 Aplicación en Procesos Productivos

Debido a sus 6 ejes de articulaciones, la movilidad constituye una característica destacada de este robot. Sin embargo, los procesos productivos requieren soluciones con altos grados de eficacia. Precisamente en este aspecto, el brazo puede realizar manipulaciones de objetos con un peso de 2 kg (3 kg con la muñeca apuntando hacia abajo), alcanzando una velocidad máxima de 4400 mm/s. Esto permite lograr ciclos de trabajo con tiempos mínimos y una productividad considerablemente superior. Además el robot posee una excelente repetitividad de los procesos, lo que permite posicionar componentes y piezas con una precisión de ±0,02 mm. Por lo tanto el brazo robótico, RV-2SDB ofrece cualidades de seguridad en la producción y calidad de los procesos.

1.8.3 Integración Sencilla en Sistemas

El robot RV-2SDB, diseñado por Mitsubishi, puede combinarse sin ninguna complejidad con un gran número de componentes de automatización. Esto debido a que la unidad de control del robot puede comunicarse vía un puerto Ethernet con un terminal de operación de la serie GOT (Terminal grafica de operaciones). De esta manera, es posible configurar varios paneles de control mediante un único terminal grafico de operaciones. Esto permite conseguir ahorro de tiempos de desarrollo y costos del sistema. La interfaz SSCNET III, disponible de serie en el robot, permite controlar hasta 8 servo ejes, mediante un cable de fibra óptica inmune a las interferencias. A ello se ha de añadir dos entradas de encoder en la unidad de control, permitiendo una sincronización sin complicaciones con correas y/o cintas transportadoras. Por otro lado, la interfaz Ethernet integrada permite conectar una cámara de un sistema de captación de imágenes.

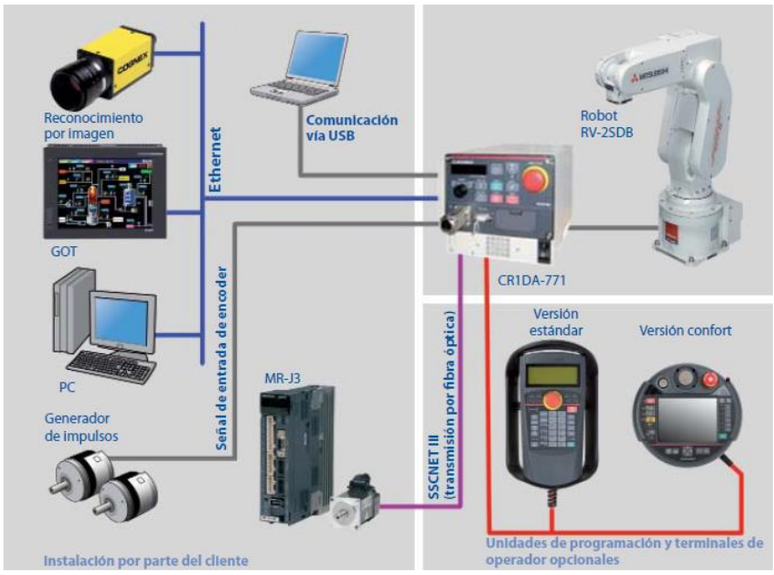

Figura 1-5: Integración de Sistemas

Fuente: www.esco.be

1.8.4 Herramientas de Programación

El robot RV-2SDB se beneficia de un extenso abanico de herramientas de programación para el desarrollo de sistemas de producción y/o pruebas. Con estas puede ahorrarse mucho tiempo a la hora de crear las rutinas de trabajo. El software multilingüe RT Toolbox 2 es la herramienta de programación estándar para el sistema del brazo robótico. Con la misma pueden verificarse los programas del robot y simularse los ciclos de trabajo. Con ayuda del software MELFA-Works es posible representar en 3D el robot, el lugar de trabajo, simular las operaciones y verificar la ejecución del programa del robot. Esta posibilidad permite localizar a tiempo y subsanar los problemas que podrían surgir antes de que ocasionen costos elevados en el sistema de producción definitivo.

1.8.4.1 RT toolbox2

Ventanas de software basado para programa de robot, eliminación de fallos, parámetros, simulación y diagnóstico.

1.8.4.2 Melfa-visión

Instrumento de configuración de software de visión de máquina para sensores de visión de Perspicacia Cognex.

1.8.4.3 Melfa-trabajos

La simulación avanzada de 3D y el software de diseño que trabaja como un accesorio a SolidWorks.

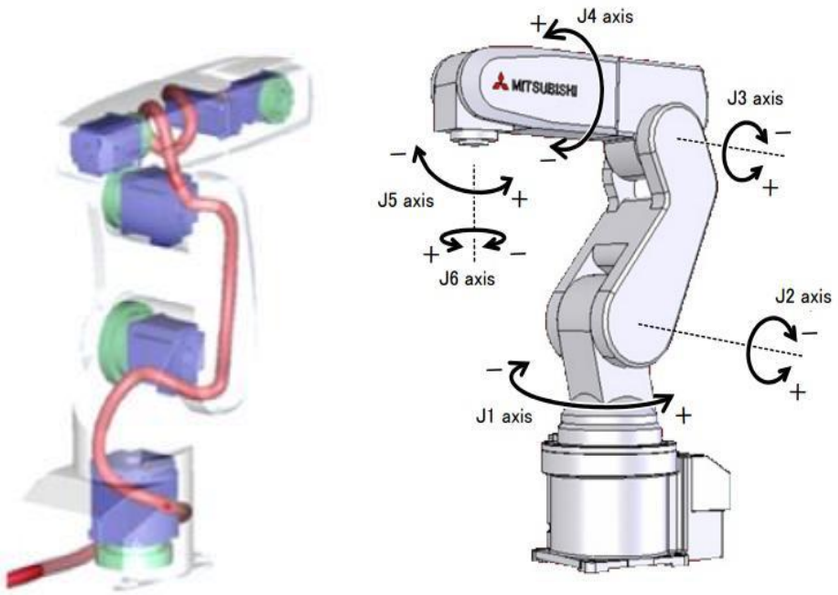

Figura 1-6: Figura: Diseño Estructural del Brazo Robótico.

Fuente: www.inducontrol.com.pe

Ítem		**Características**
Robot		RV-2SDB
Tipo Constructivo		Brazo Articulado Vertical
Montaje		Puede montarse en paredes, techos o suelos
Número de ejes		6
Longitud del Brazo (mm)		230+270
Esfuerzo de elevación	Valor Nominal (kg)	2 (kg)

	Máximo (kg)	3 (kg) Con la Muñeca Hacia Abajo
Velocidad Máxima Resultante (mm/s)		4.400 (mm/s)
Límites de trabajo máx. (mm)		504
Resolución (mm)		± 0.02
Peso (kg)		19 (kg)
Rango de Operación (Grados º)	J1	480º (-240º a +240º)
	J2	240º (-120º a +120º)
	J3	160º (0º a +160º)
	J4	400º (-200º a +200º)
	J5	240º (-120º a +120º)
	J6	720º (-360º a +360º)
Velocidad Máxima de Operación (Grados º/Segundos) (º/seg)	J1	225 (º/seg)
	J2	150 (º/seg)
	J3	275 (º/seg)
	J4	412 (º/seg)
	J5	450 (º/seg)
	J6	720 (º/seg)

Tabla 1-1: Modelo y Características del Brazo.

Fuente: Manual del Brazo Robótico

1.8.5 Controlador CR1DA-771

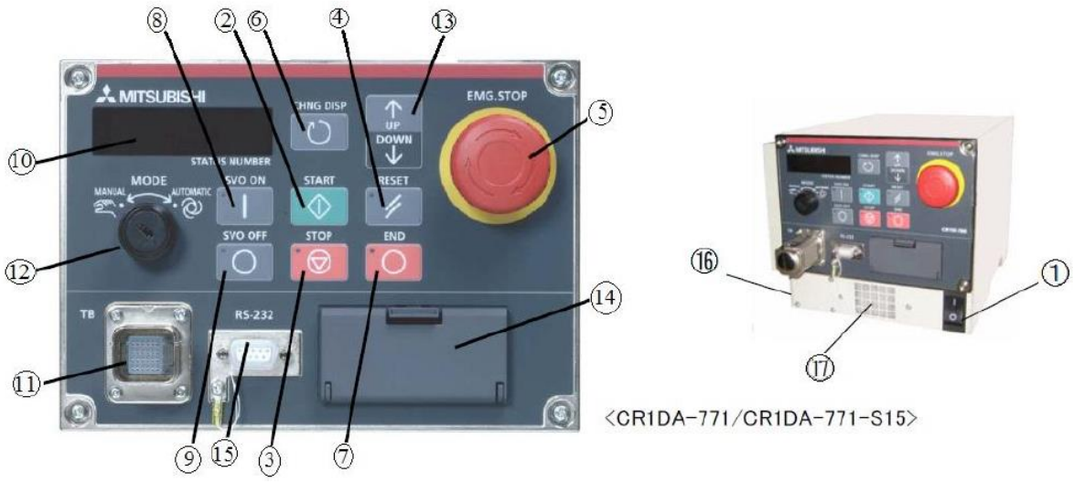

Figura 1-7: Controlador CR1DA-771.

Fuente: es.slideshare.net/robinsonsamuelgutierrezsanchez

1. **POWER:** Interruptor para encendido/apagado tanto del teaching box (T/B) como del controlador. ON/OFF.

2. **START:** Inicia el programa seleccionado y activa el robot. Una vez iniciado el programa éste corre de forma continua hasta pulsar nuevamente.

3. **STOP:** Detiene el programa seleccionado y el robot inmediatamente.

4. **RESET:** Restaura estado de error y reinicia el programa

5. **EMG.STOP:** Detiene el robot cuando existe un paro de emergencia. Cuando se activa este botón el paro es general en todas las estaciones involucradas.

6. **CHANGE DISP:** Cambia el contenido mostrado en el display, número de línea ejecutándose, Override, Error, número de programa.

7. **END:** Finaliza el programa que se está ejecutando en la última línea.

8. **SVO. ON:** Activa la alimentación de los servo motores del robot. El LED permanece encendido de color verde mientras el servo esté en funcionamiento.

9. **SVO. OFF:** Desactiva la alimentación de los servo motores del robot. El LED permanece encendido de color rojo mientras el servo esté desactivado.

10. **STATUS NUMBER:** Display donde se puede observar el número de error, el número de programa, y el porcentaje de velocidad.

11. **CONEXIÓN DE TEACHING BOX (TB):** Es un punto de conexión especializado que se utiliza para vincular el teaching box al controlador.

12. **MODE:** Activa los diferentes modos de operación del robot, ya sea de modo manual o automático:
    - AUTO (Op.): Seleccionando esta posición el controlador queda activado como único manipulador.
    - TEACH: Cuando este modo es seleccionado el T/B es activado, esto da a entender que las operaciones y los movimientos quedan restringidos para lo que se comande desde el teaching box.
    - AUTO (Ext.): Seleccionando este modo, queda el PC activado como dispositivo propio de control para el brazo robótico, deshabilitando las anteriores.

13. **UP/DOWN:** Moverse por los contenidos del display de estado, editar Override.

14. **CUBIERTA DE TERMINALES (CR1DA-700 SERIES):** Terminal que conecta el cable de alimentación.

15. **CONECTOR A PC:** Este es un emisor-receptor de datos de especificación RS-232 para hacer la conexión entre el controlador y el PC.

16. **ENCHUFE DE ALIMENTACION (COSTADO):** Transportador de energía, a partir del suministro eléctrico de alimentación central.

17. **FILTRO DE ENTRADA DE AIRE:** Permite la ventilación y recirculación de aire para la refrigeración interna.

Control del Robot		**CR1DA-771(RV-2SDB)**	**Observación**
Lenguaje de Programación		MELFA-BASIC V	
Determinación de la posición		Aprendizaje, introducción manual de datos (IMD)	
Entradas/Salidas Externas	Entradas/Salidas Generales	0 Entradas/ 0 Salidas (max. 256/256 como equipamiento especial)	
	Entradas/Salidas Especiales	Definido por Usuario	
	Entradas/Salidas para Pinza-Mano	4 Entradas/0 Salidas	Como opción pueden incorporarse 4 salidas adicionales.
	Parada de emergencia Externa	1	Con diseño basado en dos circuitos
	Contacto de cierre de puerta	1	Con diseño basado en dos circuitos
	Pulsador de Permiso	1	Con diseño basado en dos circuitos
	Sincronización para eje adicional	1	Con diseño basado en dos circuitos
	Estado operativo	1	Con diseño basado en dos circuitos

	Salida de errores		Con diseño basado en dos circuitos
Interfaces	RS-232	1	Aplicación para PC, sensor de captación de imagen, etc
	Ethernet	1	10BASE-T/100BASE-TX
	USB	1	Solo función de dispositivo, conector Mini-B
	Eje adicional	1	Para un total de hasta 8 servomotores
	Seguimiento de la cinta transportadora	1	Para dos encoder
	Conexión en la mano del robot	1	Para dos pinzas neumáticas
	Slot de ampliación	1	Para interfaces adicionales
Alimentación Eléctrica	Tensión de entrada	200-230V AC ±10 %, monofásica (180-253V)	
	Potencia absorbida kVA	0,5	Sin pico de corriente transitorio de conexión
	Frecuencia Hz	50/60	
Temperatura ambiente ºC		0-40	
Nivel de rendimiento (PL)		d	
Dimensiones (AnxAlxPr) mm		240x200x290	Sin piezas sobresalientes
Peso (kg)		Aproximado 9 kg	
Carcasa/Clase de protección		Carcasa cerrada para instalación sobre suelo / IP20	

Tabla 1-2: Especificaciones del controlador.

Fuente: Manual del controlador CR1DA-771

1.8.6 Consola Portátil R32TB

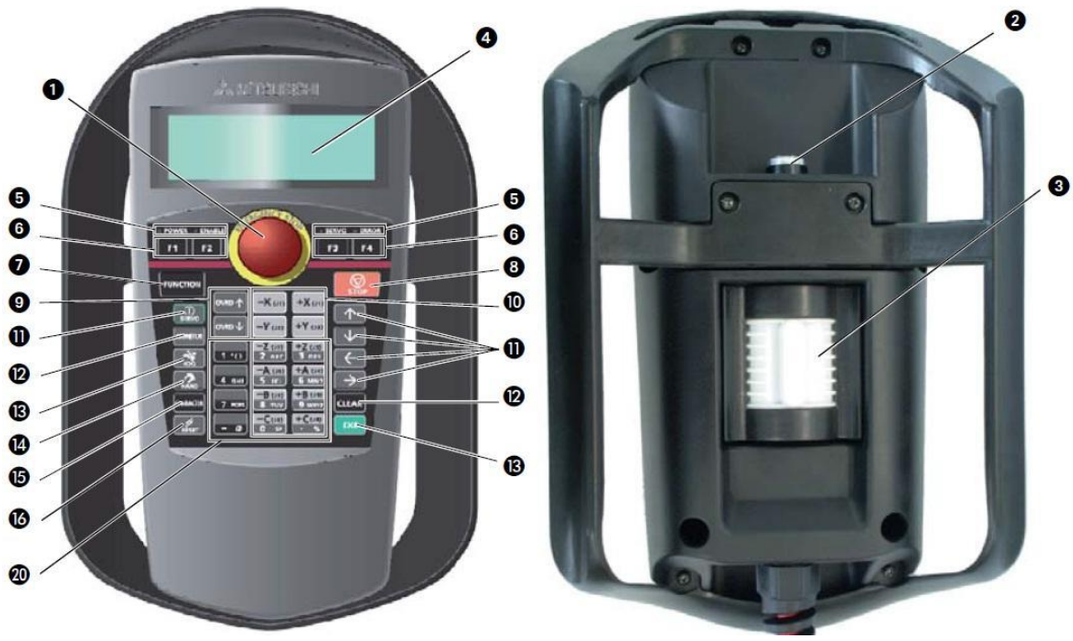

Figura 1-8: Consola Portátil y/o Teaching Box.

Fuente: Manual del Controlador

1. **PARADA DE EMERGENCIA:** Interruptor de parada de emergencia con funcionamiento de enclavamiento, al accionar el interruptor el brazo del robot se detiene inmediatamente, desconectando la tensión de alimentación de los servomotores.

2. **ENABLE / DISABLE:** Habilitación del control mediante la unidad de programación. Al Presionar el interruptor en la posición "Enable" permite controlar el brazo del robot mediante la unidad de programación. Si la unidad de programación está activa, no es posible intervenir en el control del brazo del robot mediante el panel de control y mediante dispositivos externos.

3. **DEADMAN SWITCH:** Este pulsador permite enclavar el control y activar los servo accionamientos, si están conectados a la unidad de programación.

4. **DISPLAY LCD:** En la pantalla se visualiza el estado del programa, coordenadas y configuraciones del brazo del robot.

5. **INDICADOR DE ESTADO:** El led indica el estado del robot o de la unidad de programación (T/B).

6. **TECLAS [F1], [F2], [F3], [F4]:** Permiten la ejecución de las funciones actualmente visualizadas en el display.

7. **TECLA [FUNCION]:** Conmutación de las funciones visualizadas en la parte inferior del display.

8. **TECLA [STOP]:** Interrupción del programa en ejecución y desaceleración del robot. La función coincide con la tecla [STOP] en el panel de la unidad de control la cual detiene el programa seleccionado y el robot inmediatamente.

9. **TECLAS [OVRD], [OVRD↓]:** Permite la variación de velocidad de los desplazamientos. Activando la tecla [OVRD↑] se aumenta la velocidad de desplazamiento, mientras que activando la tecla [OVRD↓] se reduce dicha velocidad.

10. **TECLAS PARA EL MODO JOG: [-X/ (J1)]… [+C/ (J6)]:** Tecla de función para el modo JOG. En el modo JOG de articulación es posible mover individualmente las articulaciones. En el modo JOG de XYZ, es posible mover el brazo del robot a lo largo de cada uno de los ejes de coordenadas. Con las teclas se realiza también la introducción de números de opción de menú o de números de opción de paso.

11. **TECLA [SERVO]:** Al pulsar la tecla [SERVO] con la tecla [ENABLE] accionada hasta la mitad para conectar la tensión de alimentación del servo accionamiento.

12. **TECLA [MONITOR]:** Cambia al modo Monitor y muestra el menú Monitor.

13. **TECLA [JOG]:** Cambia al modo JOG y muestra el menú JOG.

14. **TECLA [HAND]:** Cambia al modo Hand y muestra el menú Hand.

15. **TECLA [CHARACTER]:** Llama al menú Edición y alterna entre números y letras.
16. **TECLA [RESET]:** Reposición de un código de error. Junto con la tecla [EXE] se repone el programa.
17. **TECLA [], [↓] [←] [→]:** Mueve el cursor en la dirección correspondiente.

18. **TECLA [CLEAR]:** Borra el carácter situado en la posición del cursor.

19. **TECLA [EXE]:** Introducción de datos o movimiento del robot en el modo Directo.
20. **TECLA DE CARÁCTER:** Sobrescribe el carácter situado en la posición del cursor.

Alcance		Uso, programación y el seguimiento de todas las funciones del robot.
Funciones		La lectura de la información, incluso durante la actual operación; programa de edición utilizando el teclado virtual en t9- estándar; supervisión de una en una de sus salidas; indicación de error; movimiento recto.36 teclas de conmutación para el control operacional.
Programación y seguimiento		Lineal.
Pantalla Visualización	Tipo (Físicamente)	Pantalla LCD gráfica, monocromática, con luz de fondo
	Ejecución	LCD-pantalla (24 caracteres x 8 líneas).
Método de conexión		Conexión con el controlador y conector cuadrado de 24 pines.
Interfaz		Rss422 para la conexión de controladores robóticos.
Protección		IP 65
Peso (kg.)		0,9 (excluyendo el cable).

Tabla 1-3: Especificaciones del Controlador.

Fuente: Manual del controlador CR1DA-771

CAPÍTULO 2: PROGRAMACIÓN, INSTALACIÓN E INTEGRACION DE SOFTWARE

2.1 INSTALACIÓN DE OPENCV EN RASPBERRY PI 3 B+

2.1.1 Paso 1: Expandir el sistema de archivos.

Lo primero que se debe realizar es expandir el sistema de archivos para incluir todo el espacio disponible en la tarjeta micro-SD, para esto se utilizan el comando:

- **Sudo raspi-config**

Luego se debe seleccionar Opciones avanzadas en el menú, a continuación, seleccionar expandir sistema de archivos y luego debe presionar finalizar y reiniciar la Raspberry utilizando el comando **sudo reboot**.

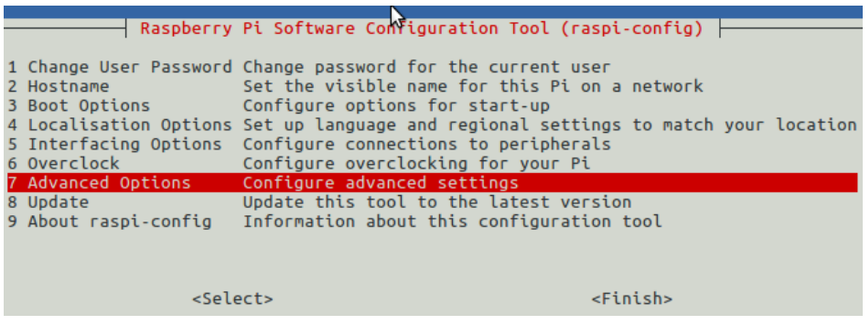

Figura 2-1: Seleccione el elemento " Opciones avanzadas del menú "

Fuente: www.pyimagesearch.com

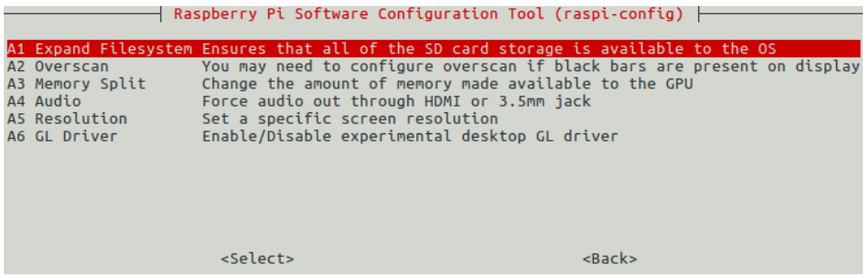

Figura 2-2: Expandiendo el sistema de archivos en Raspberry Pi 3.

Fuente: www.pyimagesearch.com

Después de reiniciar la Raspberry, el sistema de archivos debería haberse expandido para incluir todo el espacio disponible en la tarjeta micro-SD. Para verificar que el disco se haya expandido se debe ejecutar **df - h** y examinar la salida.

2.1.2 Paso 2: Instalar Dependencias.

El primer paso, antes de instalar cualquier dependencia y/o librería es actualizar cualquier paquete existente, para esto se utilizan los comandos:

- **sudo apt-get update && sudo apt-get upgrade**

Luego es necesario instalar algunas herramientas del desarrollador como CMake, que nos permite configurar el proceso de construcción de OpenCV, para esto utilizamos el siguiente comando:

- **sudo apt-get install build-essential cmake pkg-config**

Luego de esto, se debe instalar algunos paquetes de entrada y salida de imágenes que permiten cargar diferentes tipos de formatos de imágenes desde la memoria, tales como formatos jpeg, png, tiff, etc. Para esto se utiliza el comando:

- **sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev**

También necesitamos paquetes de entrada y salida de video. Estas bibliotecas permiten leer varios formatos de archivo de video desde la memoria, así como trabajar directamente con secuencias de video. Para esto se utilizan los comandos:

- **sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev**
- **sudo apt-get install libxvidcore-dev libx264-dev**

La biblioteca OpenCV viene con un sub-módulo llamado highgui que se usa para mostrar imágenes en la pantalla de la Raspberry y crear interfaces gráficas de usuario básicas. Para compilar el módulo highgui, es necesario instalar la biblioteca de desarrollo de gtk, para esto utilizamos:

- **sudo apt-get install libgtk2.0-dev libgtk-3-dev**

Muchas operaciones dentro de OpenCV (es decir, operaciones matriciales) pueden optimizarse aún más instalando algunas dependencias adicionales, ya que son de gran importancia para dispositivos con recursos limitados como la Raspberry Pi, para esto se utiliza:

- **sudo apt-get install libatlas-base-dev gfortran**

Por último, se instalan los archivos de encabezado de Python 3 para poder compilar OpenCV con enlaces de Python, para esto se utiliza:

- **sudo apt-get install python2.7-dev python3-dev**

2.1.3 Paso 3: Descargar el código fuente de OpenCV.

Ahora que están las dependencias y/o librerías instaladas, se debe tomar la versión 3.3.0 de OpenCV, desde el sitio oficial, para esto se utiliza:

- **cd ~**
- **$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip**
- **$ unzip opencv.zip**

Para obtener la instalación completa de OpenCV, y tener acceso a todos los tipos de funciones debemos realizar la siguiente instalación.

- **wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip**
- **unzip opencv_contrib.zip**

2.1.4 Paso 4: Python 3

Antes de comenzar a compilar OpenCV en la Raspberry Pi 3, primero necesitamos instalar pip, un administrador de paquetes de Python, para realizar esta instalación se debe realizar lo siguiente en la pantalla de comandos:

- **wget https://bootstrap.pypa.io/get-pip.py**
- **sudo python get-pip.py**
- **sudo python3 get-pip.py**

Instalar virtualenv y virtualenvwrapper

Es importante comprender que un entorno virtual es una herramienta especial que se utiliza para mantener las dependencias requeridas por los diferentes proyectos en lugares separados mediante la creación de entornos Python aislados e independientes para cada uno de ellos (Ejemplo: Resolver el proyecto X depende de la versión 1.x, pero el Proyecto Y necesita la versión 4.x).
También mantiene los paquetes ordenados y libres de desorden, por lo tanto para instalar un entorno virtual debemos ejecutar lo siguiente en la pantalla de comandos:

- **sudo pip install virtualenv virtualenvwrapper**
- **sudo rm -rf ~/.cache/pip**

Una vez instalados virtualenv y virtualenvwrapper es necesario actualizar los archivos de perfil para incluir las siguientes líneas en la parte inferior del archivo:

- **virtualenv and virtualenvwrapper**
- **export WORKON_HOME=$HOME/.virtualenvs**
- **export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3**
- **source /usr/local/bin/virtualenvwrapper.sh**
- **echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile**
- **echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile**
- **echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile**
- **echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile**

Una vez que el perfil se encuentra actualizado, es necesario volver a cargarlo para asegurar que los cambios tengan efecto.

Para esto es necesario realizar lo siguiente:

- **Cerrar sesión y luego volver a iniciar sesión.**
- **Cerrar una instancia de terminal y abrir una nueva.**
- **Otra alternativa es, simplemente usar el comando source ~/.profile**

Creando el entorno Virtual de Python.

Es importante destacar que el entorno virtual cv de Python es completamente independiente y está aislado de la versión predeterminada de Python incluida en la descarga de Raspbian Stretch.

En este paso se crea el entorno virtual de Python que se utiliza para el desarrollo de visión artificial. En este caso el siguiente comando crea un entorno virtual llamado Python 3:

- **mkvirtualenv cv -p python3**

Comprobar que se está en el entorno virtual CV

Al reiniciar y/o cerrar sesión en la Raspberry Pi y luego volver a iniciar o abrir un nuevo terminal, se debe utilizar los siguientes comandos para volver a acceder al entorno virtual CV.

- **source ~/.profile**
- **workon cv**

Figura 2-3: Entorno Virtual.

Fuente: www.pyimagesearch.com

Instalación de Numpy en Raspberry Pi.

Python cuenta con una única dependencia que es NumPy, el cual es un paquete de Python utilizado para el procesamiento numérico y/o matemático.
Es importante destacar que esta instalación debe realizarse dentro del entorno virtual de CV.

- **pip install numpy**

Compilar e Instalar OpenCV.

Para realizar la compilación e instalación de OpenCV es necesario estar en el entorno virtual CV, ya que desde este entorno podemos configurar la compilación utilizando Cmake, a través de los siguientes comandos:

- cd ~/opencv-3.3.0/
- $ mkdir build
- $ cd build
- $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
- -D CMAKE_INSTALL_PREFIX=/usr/local \
- -D INSTALL_PYTHON_EXAMPLES=ON \
- -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
- -D BUILD_EXAMPLES=ON ..

Otro punto importante es asegurarse de que la salida Cmake, esté compilando OpenCV para Python 3, para esto se debe chequear que las rutas donde se está instalando sean válidas (Ruta de interprete, librerías, números y paquetes).

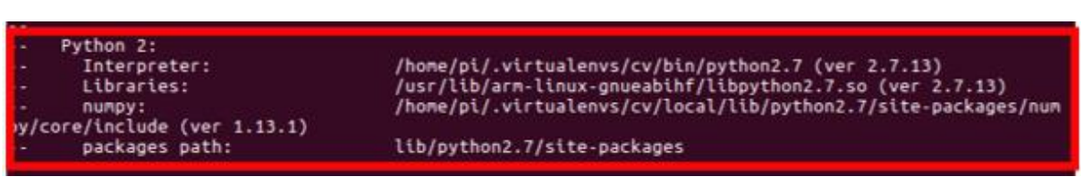

Figura 2-4: Compilación Python 3.

Fuente: www.pyimagesearch.com

Configurar el tamaño del espacio de intercambio antes de compilar

Antes de comenzar el proceso de compilación, se debe aumentar el tamaño del espacio de intercambio. Esto permite que OpenCV compile con los cuatro núcleos de la Raspberry Pi, sin que la compilación se bloquee debido a problemas de memoria. Una vez finalizado el proceso de compilación se debe volver a la normalidad el tamaño de compilación.

- **# CONF_SWAPSIZE=100**
- **CONF_SWAPSIZE=1024**

Para activar el nuevo espacio de intercambio, se debe reiniciar el servicio de intercambio, utilizando los siguientes comandos:

- **sudo /etc/init.d/dphys-swapfile stop**
- **sudo /etc/init.d/dphys-swapfile start**

Compilar OpenCV.

Para realizar la compilación de OpenCV, se debe ejecutar el siguiente comando:

- **make –j4**

Una vez que OpenCV haya terminado de compilar, su salida debería ser similar a la siguiente imagen:

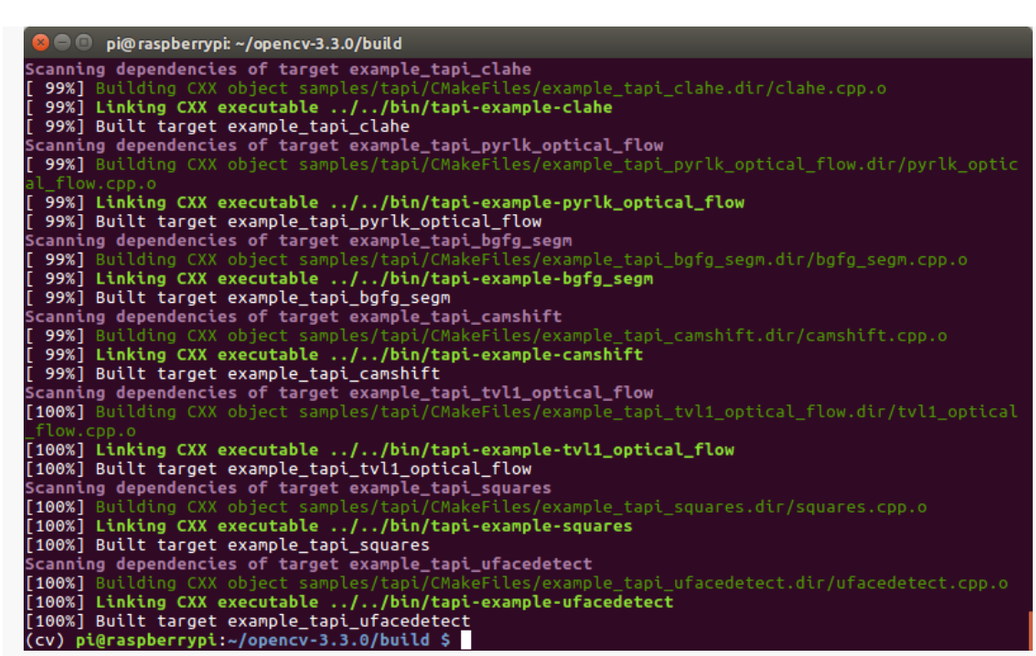

Figura 2-5: Término de Compilación OpenCV

Fuente: www.pyimagesearch.com

Una vez finalizado el proceso anterior debemos instalar OpenCV, utilizando los siguientes comando:

- sudo make install
- sudo ldconfig

2.1.5 Paso 5: Finalizar la Instalación de OpenCV para Python 3 en la Raspberry Pi.

Después de ejecutar make install, los enlaces de OpenCV y Python deben instalarse en /usr / local / lib / python3.5 / sitio - paquetes. Esto se puede verificar con el comando ls, el cual indicara lo siguiente:

- ls -l /usr/local/lib/python3.5/site-packages/
- total 1852
- -rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so

Finalmente debemos vincular simétricamente nuestros enlaces OpenCV en el entorno virtual cv para Python 3.5, para esto debemos ejecutar los siguientes comandos:

- **cd ~/.virtualenvs/cv/lib/python3.5/site-packages/**
- **ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so**

2.2 APLICACIÓN DE VISION ARTIFICIAL: ETAPAS DE PROCESAMIENTO DE IMÁGENES

2.2.1 Etapas de una Aplicación de Visión por Computador

Las etapas de programación y procesamiento de imágenes de visión artificial son variadas y dependen del resultado o finalidad a ejecutar, para el siguiente proyecto de reconocimiento y selección de imágenes por color y forma se requirió de distintos procesos y tanto de Visión de bajo y alto nivel.

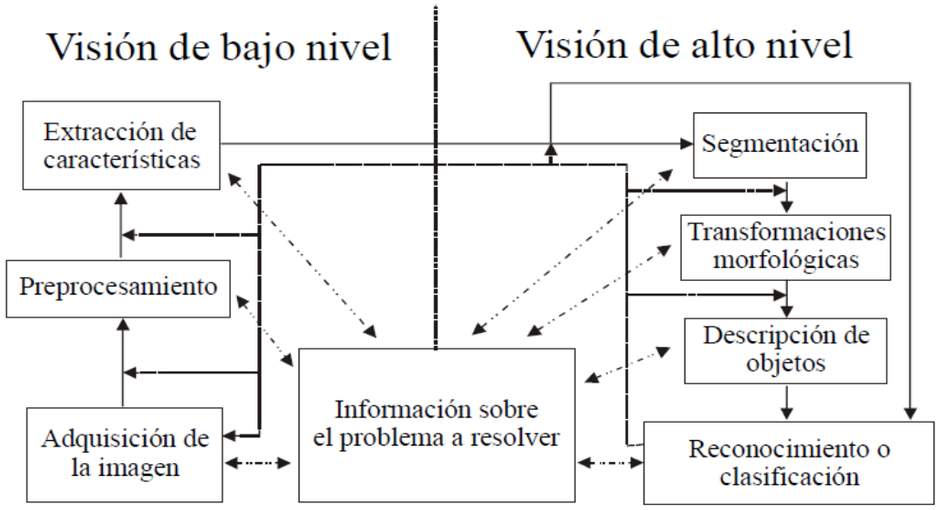

Figura 2-6: Etapas de una aplicación de visión artificial

Fuente: programarfacil.com

2.2.2 Procesamiento de Imágenes

Una imagen se conforma digitalmente por una serie de valores en una matriz, por ejemplo una matriz como la de la ecuación (Fig. 2-7), podría representar una imagen de escala de grises, donde cada valor dentro de esta es considerado un pixel y donde axy toma valores entre un rango de 0 y 255, además cada pixel de la matriz se considera de 1 byte (8 bits). Teniendo en cuenta esta información, se puede decir que si un valor axy se acerca a 0, significa que ese pixel es más oscuro, mientras que si se acerca a 255, el pixel es más claro. Cuando solo existen esos dos valores (0 y 255), se considera una imagen binaria, es decir de solo blanco y negro. La resolución de una imagen es conocida como los valores de m y n, es decir, las dimensiones de la matriz G.

$$G = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Figura 2-7: Matriz G

Fuente: Fuente: es.wikipedia.org

Una imagen de color está compuesta por tres matrices con la forma de G (Fig. 2-8), en donde cada matriz R, G y B, representan la intensidad en la imagen de los colores rojo, verde y azul respectivamente, que al combinarse estos colores, se puede generar toda una gama de colores que percibe el ser humano. En este caso, un pixel es representado por tres valores (rxy; gxy; bxy) indicando la intensidad de cada color, tal como se muestra en la ecuación (Fig. 2-8).

$$RGB = \left[R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ r_{21} & r_{22} & r_{23} & \cdots & r_{2n} \\ r_{31} & r_{32} & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & r_{m3} & \cdots & r_{mn} \end{bmatrix} G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & g_{m3} & \cdots & g_{mn} \end{bmatrix} B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \cdots & b_{mn} \end{bmatrix} \right]$$

Figura 2-8: Matriz R, B, G.

Fuente: es.wikipedia.org

Bajo este concepto de estructura de una imagen podemos entender más conceptos de procesamiento de imágenes.

2.2.3 Espacios de Color

Hay varias maneras en que se puede describir una imagen, una de ellas es la que se describió anteriormente y es el espacio RGB, el cual se utiliza bastante debido a que físicamente cada pixel en una pantalla es representado por la cantidad de intensidad que se le aplique para cada color, y la pantalla completa se describe como la matriz de la ecuación (Fig. 2-8). Sin embargo, esta matriz puede transformarse de varias maneras digitalmente, modificando los valores RGB y transformando las matrices R, G y B, para obtener otros espacios de color y así representar la imagen con otra matriz que contenga los valores del nuevo espacio de color. A continuación se describirá el espacio de color con el que trabajamos en este proyecto.

2.2.3.1 Espacio HSV

El modelo HSV es un espacio de color que al igual que el espacio RGB descrito en la ecuación (Fig. 2-8), es representado con tres componentes que son H, S y V, que por sus siglas en ingles corresponden al tono o matiz (Hue), saturación (Saturation) y brillo o valor (Value). Estas tres magnitudes se representan por medio de un diagrama cilíndrico que contiene la información necesaria para hallar el color deseado (Fig.2-9). El tono o matiz está representado por un ángulo que recorre el cilindro sobre su eje, los valores de brillo está representados el eje vertical del cilindro y la saturación por el eje horizontal.

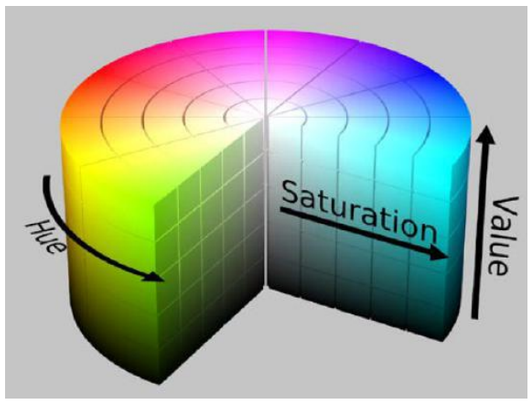

Figura 2-9: Espacio Color HSV

Fuente: Fuente: es.wikipedia.org

2.2.3.1.1 HUE – Tono o Matiz

La gama cromática se representa por medio del valor del ángulo del matiz respecto al eje del cilindro, donde cada ángulo representa un color diferente. En el caso de ser programado en Python con la biblioteca de OpenCV este valor varía entre 0° a 180° puesto que si se trabajara con 360°, este valor supera el valor de 255, que es el valor máximo de un pixel en el formato llamado uint8 que utiliza únicamente 8 bits. Se puede observar en la figura 4, como varía el color al variar el ángulo, manteniendo constante la saturación y el brillo en 255 (100 %).

Figura 2-10: Variación del matiz en HSV con saturación y brillo máximo

Fuente: Fuente: es.wikipedia.org

2.2.3.1.2 Saturation – Saturación

La saturación indica la escala de grises en el espacio de color. Los valores varían de 0% a 100 %. Al aumentar este parámetro, se obtiene un color más vivo, mientras que al disminuirlo, el color tiende al blanco. En la figura 2-11 se puede apreciar cómo se comportan los cambios de este componente, tomando como ejemplo

un tono rojo de matiz cero (0) y un brillo (value) máximo (255) y variando la componente de saturación.

Figura 2-11: Saturación en HSV con matiz en 0o y brillo máximo.

Fuente: Fuente: es.wikipedia.org

Estos valores en OpenCV varían entre 0 y 255, para representar el rango mencionado anteriormente de 0% a 100% de saturación.

2.2.3.1.3 Value – Valor

El valor o brillo es el grado de luminosidad del color que al igual que la saturación, varía entre 0% y 100 %. Cuando lo valores están cercanos a 0 el espacio de color estará en la tonalidad oscura en cambio, mientras más grande sea el valor, va disminuyendo la oscuridad. En la figura 2-12 se puede apreciar cómo aumenta el brillo del azul, que corresponde a un ángulo de matiz de 90% y utilizando una saturación del 100 %.

Figura 2-12: Brillo en HSV con saturación máxima y matiz de 90º

Fuente: Fuente: es.wikipedia.org

2.2.4 Creación de máscara de color

Una Capa Máscara, es un tipo de capa usada para esconder o "enmascarar" parte del contenido de otra capa, en el caso de este proyecto se realiza para acotar el color a seleccionar separándolo del entorno donde se encuentran objetos de diferente color y forma que debemos descartar.

El comando utilizado es "mask = cv2.inRange(hsv, bajos, altos)"

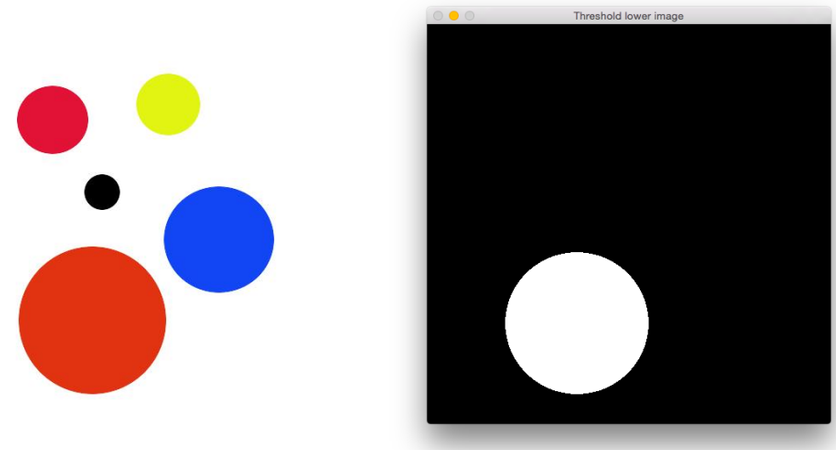

Figura 2-13: Ejemplo de aplicación de máscara de color

Fuente: solarianprogrammer.com

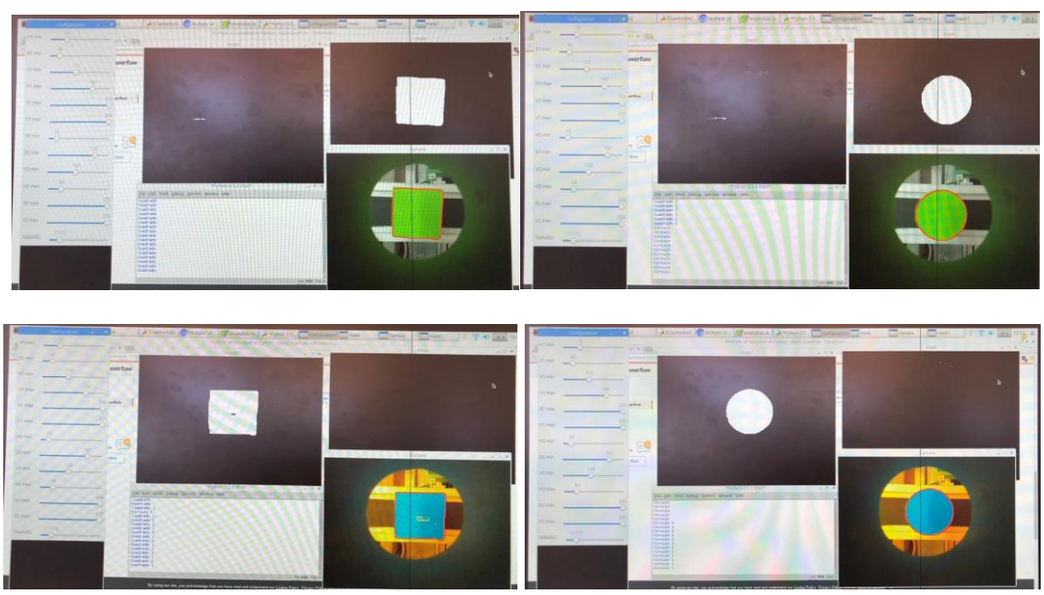

Figura 2-14: Imágenes de Proyecto realizado de dos mascaras para dos colores, verde y azul.

Fuente: Imágenes elaboración Propia

2.2.5 Procesamiento de imágenes binarias

Al aplicar un filtro de color, o cualquier otro tipo de segmentación de una imagen, generalmente se obtiene una imagen binaria (en blanco y negro), donde la región de interés son los pixeles blancos. Sin embargo, no siempre esos pixeles logran conformar toda la región de interés y quedan espacios en negro que deberían ser

blancos o en caso contrario, quedan espacios en blanco que no corresponden a la región de interés, por lo que el proceso que hay que realizar es primero eliminar esas regiones blancas que deberían ser negras y después, agregar las partes blancas que si corresponden a la región de interés. Para este proyecto y la detección del color a seleccionar debemos aplicar estos operadores con el fin de limpiar la superficie de color para buscar la forma de la superficie y acotar el tamaño del objeto o región a seleccionar.

2.2.5.1 Transformaciones Morfológicas

Las transformaciones morfológicas son algunas operaciones simples basadas en la forma de la imagen, que normalmente se aplican a imágenes binarias. Necesita dos entradas, una es nuestra imagen original, la segunda se llama elemento estructurante o núcleo (kernel) que decide la naturaleza de la operación. Dos operadores morfológicos básicos son Erosión y Dilatación. Luego, sus formas variantes como Apertura, Cierre, Gradiente, etc, también entran en juego. A continuación se verán algunas de estas transformaciones, apoyándonos en la siguiente imagen binaria:

Figura 2-15: Ejemplo imagen binaria

Fuente: unipython.com

2.2.5.2 Apertura

La apertura es simplemente la eliminación de los puntos blancos no relevantes de nuestra imagen que se encuentran en la imagen binaria. Es útil para eliminar el ruido. En este caso se utiliza la función, cv2.morphologyEx() y dentro de los valores requeridos por la función se le indicara que se trata de una apertura.

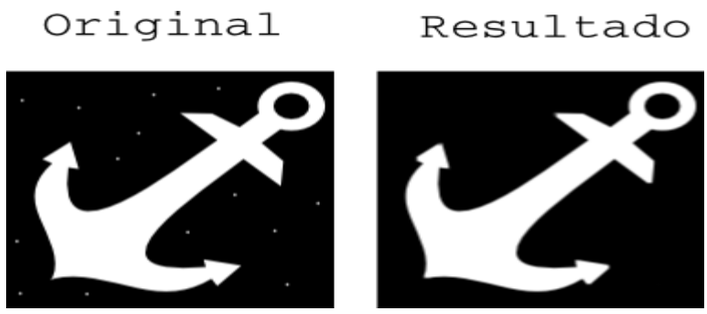

Figura 2-16: Resultado apertura.

Fuente: robologs.net

2.2.5.3 Cierre

El Cierre es el opuesto de Apertura, es decir, dilatación seguida de erosión. Este comando elimina de la imagen binaria los puntos negros no relevantes. Como en el caso anterior también se utiliza la función, cv2.morphologyEx() y dentro de los valores requeridos por la función se le indicara que se trata de un cierre.

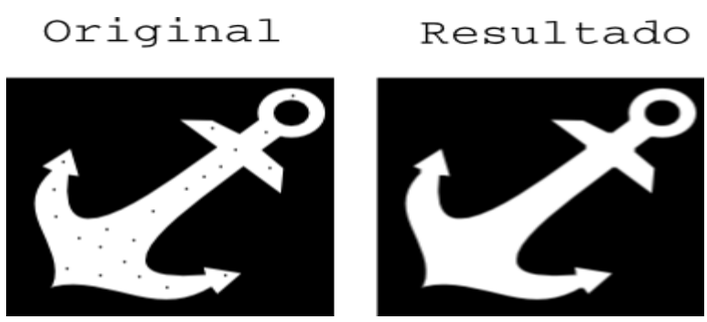

Figura 2-17: Resultado Cierre

Fuente: robologs.net

2.2.6 Detección de bordes

Para detectar mejor los bordes lo mejor es realizar un preprocesamiento a la imagen para reducir el ruido que pueda haber en la imagen y así no detectar falsos bordes, para ello es recomendable filtrar la imagen para homogeneizarla, como la creación de la máscara explicada anteriormente, sin embargo a esta se le deben aplicar transformaciones morfológicas para limpiar la imagen y luego mediante otro filtro definirla.

Una vez limpia la imagen binaria mediante el proceso de transformación morfológica se procede a detectar los bordes.

La detección de bordes consiste en tomar los valores máximos o blancos que representan el borde de la imagen, mientras que los valores mínimos o negros representan el fondo de la imagen.

En una imagen existen varios tipos de bordes que pueden ser nombrados como tipo rampa, línea, techo y paso como se aprecian en la figura 2-18, Estos bordes son generalmente un cambio abrupto de los valores de los pixeles de la imagen y son justamente los pixeles de dicho cambio los que se desean detectar y marcar del color escogido o seleccionado para resaltar el borde.

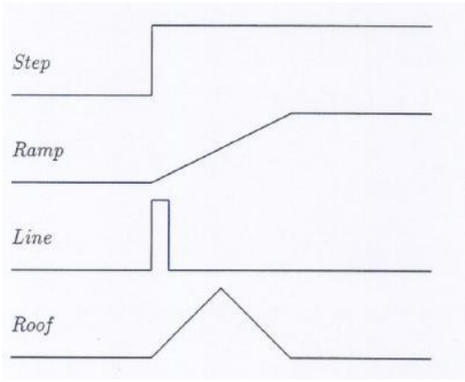

Figura 2-18: Tipos de bordes descritos en una dimensión

Fuente: unipython.com

2.2.7 Filtrado del ruido en una imagen

Las imágenes digitales no son perfectas. El ruido inherente de los propios dispositivos o los efectos contraproducentes por iluminación alteran la realidad.

De acuerdo a los diferentes entornos los cambios de luz, reflejos y brillos alteran la imagen, por lo consiguiente se deben tener en cuenta soluciones a estos problemas.

Es algo que no solo sucede en los entornos visuales. Cualquier señal digital o analógica está expuesta a diferentes fuentes de ruido.

En el tratamiento digital de imágenes hay diferentes métodos para eliminar el ruido (promediado, mediana, Gaussiano o bilateral).

Para realizar los filtros hemos utilizado un método llamado convolución. Este consiste en recorrer píxel a píxel una imagen con una máscara o kernel de $N \times N$.

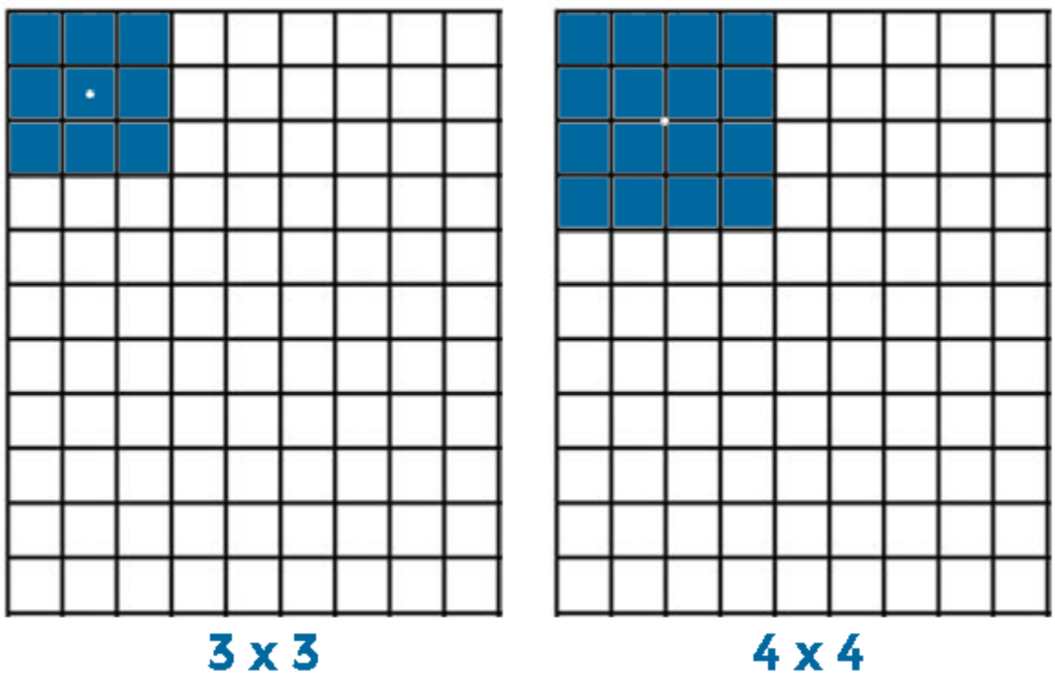

Figura 2-19: Ejemplo mascara o kernel de 3x3 y 4x4

Fuente: programarfacil.com

Es importante que el tamaño del kernel sea impar para tener un pixel central que será el pixel a tratar, en al caso de nuestro proyecto el Kernel es de 5x5.

Gracias al kernel podemos suavizar la imagen y eliminar los detalles, se puede comparar con el desenfoque de una cámara fotográfica. Para poder detectar el borde en nuestra imagen aplicaremos el filtro Gaussiano.

2.2.8 Aplicación de filtro Gaussiano para la eliminación de ruido en imágenes

El filtrado Gaussiano es igual que un promediado pero ponderado. Esta ponderación se hace siguiendo la Campana de Gauss. Con esto se consigue dar más importancia a los píxeles que están más cerca del centro de los que están más alejados.

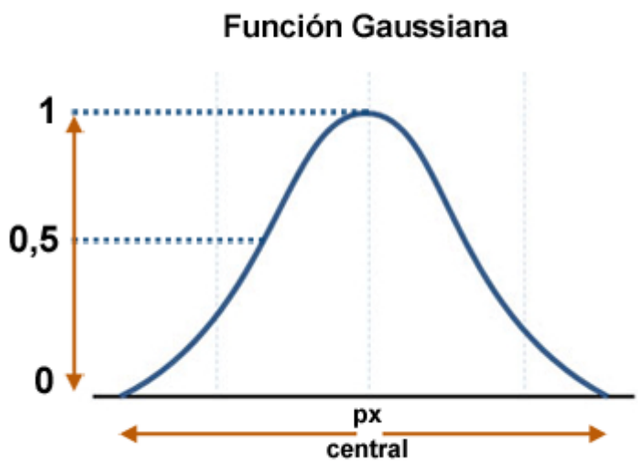

Figura 2-20: Función Gaussiana

Fuente: programarfacil.com

En términos simples la campana de Gauss es una aproximación de cómo ve el ojo humano, si lo vemos en comparación a algo natural.

La función Gaussiana o Distribución Normal se utiliza en muchas áreas de la ciencia, la altura de la población o el coeficiente intelectual se puede representar a través de la Distribución Normal. Gracias a su forma acampanada y simétrica, hace que los elementos que están más cerca del centro sean los más comunes. Por el contrario, los elementos más lejanos serán los más raros o diferentes.

Para poder aplicarlo en una imagen debemos hacerlo en dos dimensiones. Esto lo hacemos a través de una máscara o kernel de convolución.

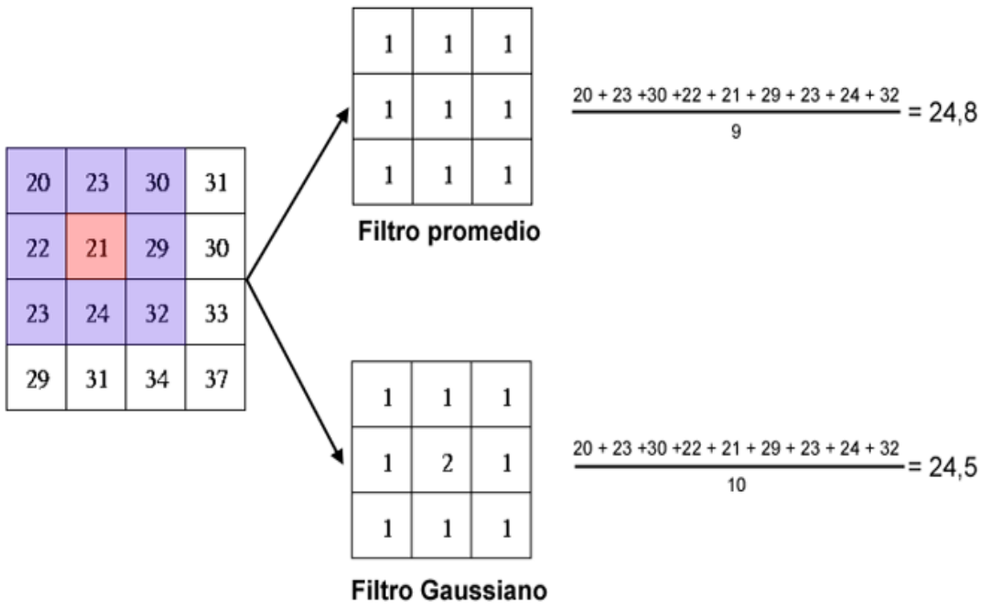

Figura 2-21: Ejemplo Filtro Gaussiano

Fuente: programarfacil.com

Como se puede comprobar en la imagen anterior, se da más importancia al píxel central que a los píxeles que están alrededor de éste.

La configuración en OpenCV para el filtro Gaussiano es como sigue:

gaussiana = cv2.GaussianBlur(imagen, (n, n), σ)

Donde

- gaussiana: es la imagen desenfocada resultante.
- imagen: es la imagen original, la que queremos suavizar o desenfocar.
- n X n: es el tamaño del kernel o máscara de convolución. Recuerda que debe ser impar.
- σ: sigma (σ) representa la desviación estándar en el eje X es decir, la anchura de la campana de Gauss. Si se pone un 0, OpenCV se encarga automáticamente de calcular ese valor para el kernel o máscara que se ha elegido. Esta es la opción aconsejable.

En nuestro caso la aplicación del filtro Gaussiano con un kernel de 5×5 es de la siguiente manera.

blur1 = cv2.GaussianBlur(mask1, (5, 5), 0)

2.2.9 Detector de bordes Canny

El proceso para detectar bordes con Canny se divide en 3 pasos.

- Detección de bordes con Sobel
- Supresión de píxeles fuera del borde
- Aplicar umbral por histéresis

Uno de los grandes inconvenientes de los algoritmos de la visión artificial es la parametrización. Muchas técnicas requieren establecer parámetros o condiciones iniciales según cada situación. En muchas ocasiones esos parámetros son exclusivos para una determinada iluminación o perspectiva.

Para nuestro proyecto se resolvió usando un rango color que incluye el brillo, luego se creó la máscara que es una imagen binaria en blanco y negro facilitando la detección de bordes.

Figura 2-22: Detección de bordes de monedas.

Fuente: programarfacil.com

Normalmente, este tipo de algoritmos se entrenan para buscar la mejor solución para múltiples casos. A todo esto se le llama Machine Learning o aprendizaje automático, creando diferentes casos de resolución de acuerdo a las condiciones.

2.2.9.1 Detección de bordes con Sobel

El detector de bordes Sobel se basa en el cálculo de la primera derivada. Esta operación matemática mide las evoluciones y los cambios de una variable. Básicamente se centra en detectar cambios de intensidad.

Cuando hablamos de bordes en una imagen, hablamos de los píxeles donde hay un cambio de intensidad. Como en las imágenes que estamos tratando donde el color a reconocer termina.

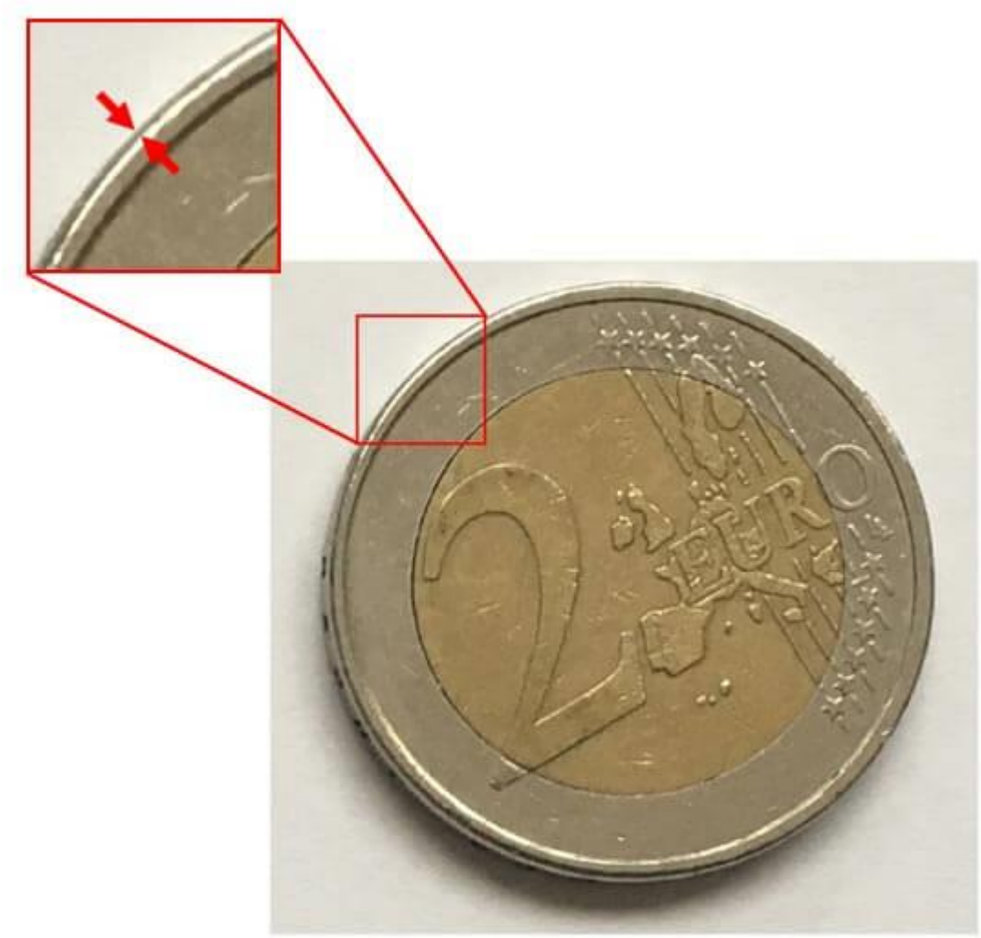

Figura 2-23: Diferencia de cambio de luz en borde

Fuente: programarfacil.com

El objetivo es poder detectar este borde a través de la primera derivada.

2.2.9.2 Filtrado de bordes mediante la supresión non-maximun

El objetivo en esta fase es que el algoritmo pueda detectar los bordes que cumplan cierta condición. El detector de bordes Canny detecta aquellos que tengan como grosor 1 (un pixel).

La supresión non-maximun es una técnica que permite adelgazar los bordes basándose en el gradiente.

2.2.9.3 Aplicar umbral por histéresis

La umbralización de imágenes nos permite también segmentar una imagen en sus diferentes objetos. Básicamente consiste en determinar un umbral por el cual se decide si un píxel forma parte del fondo o forma parte de un objeto.

En el detector de bordes Canny este es el último paso. Al contrario que el umbral simple, el umbral por histéresis se centra en establecer dos umbrales, uno máximo y otro mínimo.

Esto ayuda a determinar si un píxel forma parte de un borde o no. Pueden darse 3 casos:

- Si el valor del píxel es mayor que el umbral máximo, el píxel se considera parte del borde.
- Un píxel se considera que no es borde si su valor es menor que el umbral mínimo,
- Si está entre el máximo y el mínimo, será parte del borde si está conectado con un píxel que forma ya parte del borde.

2.2.9.4 Detector de bordes Canny con OpenCV

Todo este proceso puede ser extremadamente difícil de implementar. Sin embargo escogiendo en OpenCV el algoritmo correcto se puede realizar de manera sencilla y efectiva.

A través de la librería de OpenCV operando desde Python podemos ingresar la imagen con la queremos trabajar a un comando que detecta el borde. El valor del umbral depende exclusivamente de las condiciones y el resultado buscado.

El método para calcular los bordes con el método Canny es el siguiente.

canny = cv2.Canny(imagen, umbral_minimo, umbral_maximo)

En nuestro programa.

edges1 = cv2.Canny(mask1,10,100)

Donde:

- canny: es la imagen resultante. Aparecerán los bordes detectados tras el proceso.
- imagen: es la imagen original.
- umbral_minimo: es el umbral mínimo en la umbralización por histéresis
- umbral_maximo: es el umbral máximo en la umbralización por histéresis

2.2.10 Buscar los contornos de una imagen

Hay una diferencia entre un borde y un contorno. Los bordes son cambios de intensidad pronunciados. Pero un contorno es una curva de puntos sin huecos ni saltos, tiene un principio y el final de la curva termina en ese principio.

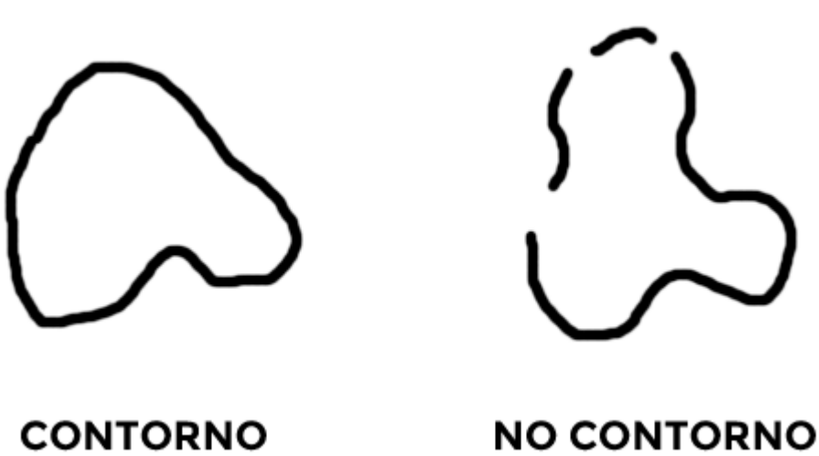

Figura 2-24: Ejemplo de buscador de contornos.

Fuente: programarfacil.com

El objetivo de esta fase es analizar todos los bordes detectados y comprobar si son contornos o no. En OpenCV lo podemos hacer con el siguiente método.

(contornos, jerarquia) = cv2.findContours(imagenbinarizada, modo_contorno, metodo_aproximacion)

En nuestro programa.

(_,contours,hier)=
cv2.findContours(edges1,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

Donde:

- Resultado: obtenemos 3 valores como resultados, el interesante para nosotros es contornos.
    - contornos: es una lista de Python con todos los contornos que ha encontrado.
    - jerarquía: la jerarquía de contornos. Estos van ordenados de acuerdo a su posición en la imagen.

- imagenbinarizada: es la imagen donde hemos detectado los bordes o umbralizado. Como este método modifica la imagen trabajaremos con una copia. Como norma general, la imagen binarizada debe ser con el fondo negro y los objetos a ser buscados deben ser blancos.

- modo_contorno: es un parámetro interno del algoritmo que indica el tipo de contorno que queremos. Puede tomar diferentes valores RETR_EXTERNAL, RETR_LIST, RETR_COMP y RETR_TREE. Nos centraremos en el primero, RETR_EXTERNAL que obtiene el contorno externo de un objeto y solamente este, simplificando la resolución y procesamiento de la imagen.

- metodo_aproximacion: este parámetro indica cómo aproximar el contorno. Básicamente le decimos si queremos almacenar todos los puntos del contorno. Si se tiene una línea recta ¿para qué almacenar todos los puntos? con dos es suficiente. En el caso que se tengan imágenes muy grandes, toma más tiempo el procesamiento de la imagen además de más memoria. Podemos tomar dos valores CHAIN_APPROX_NONE que toma todos los puntos y CHAIN_APPROX_SIMPLE, que elimina todos los puntos redundantes.

2.2.11 Dibujar los contornos en una imagen con OpenCV

Por último queda dibujar los contornos que hemos encontrado en la imagen. Esto lo haremos a través de otro método de OpenCV.

cv2.drawContours (imagen, lista_contornos, numero_contornos, color_RGB, grosor)

En nuestro programa.

cv2.drawContours(mask1,[actual],0,(0,0,255),2)

Donde:

- imagen: es la imagen donde vamos a dibujar los contornos.
- lista_contornos: es la lista de Python con los contornos.
- numero_contornos: el número de contornos que queremos dibujar si son todos pasar -1.
- color_RGB: es un array o tupla con el color RGB del contorno.
- grosor: grosor de la línea a dibujar.

Y con esto ya tendríamos el algoritmo completo. Ahora vamos a unir todas las piezas y ver cómo quedaría el código completo.

2.3 PROGRAMACIÓN PYTHON

Para la realización del programa de reconocimiento de figuras por color y forma se utilizaron varias etapas de programación, tanto básicas como específicas de la librería de OpenCV.

En primera instancia se realiza un llamado a las librerías, después se utilizaron sentencias básicas como while e if así también como barras de seguimiento para fijar valores.

El siguiente diagrama de flujo muestra el funcionamiento las etapas y comparaciones que realiza el programa en Python

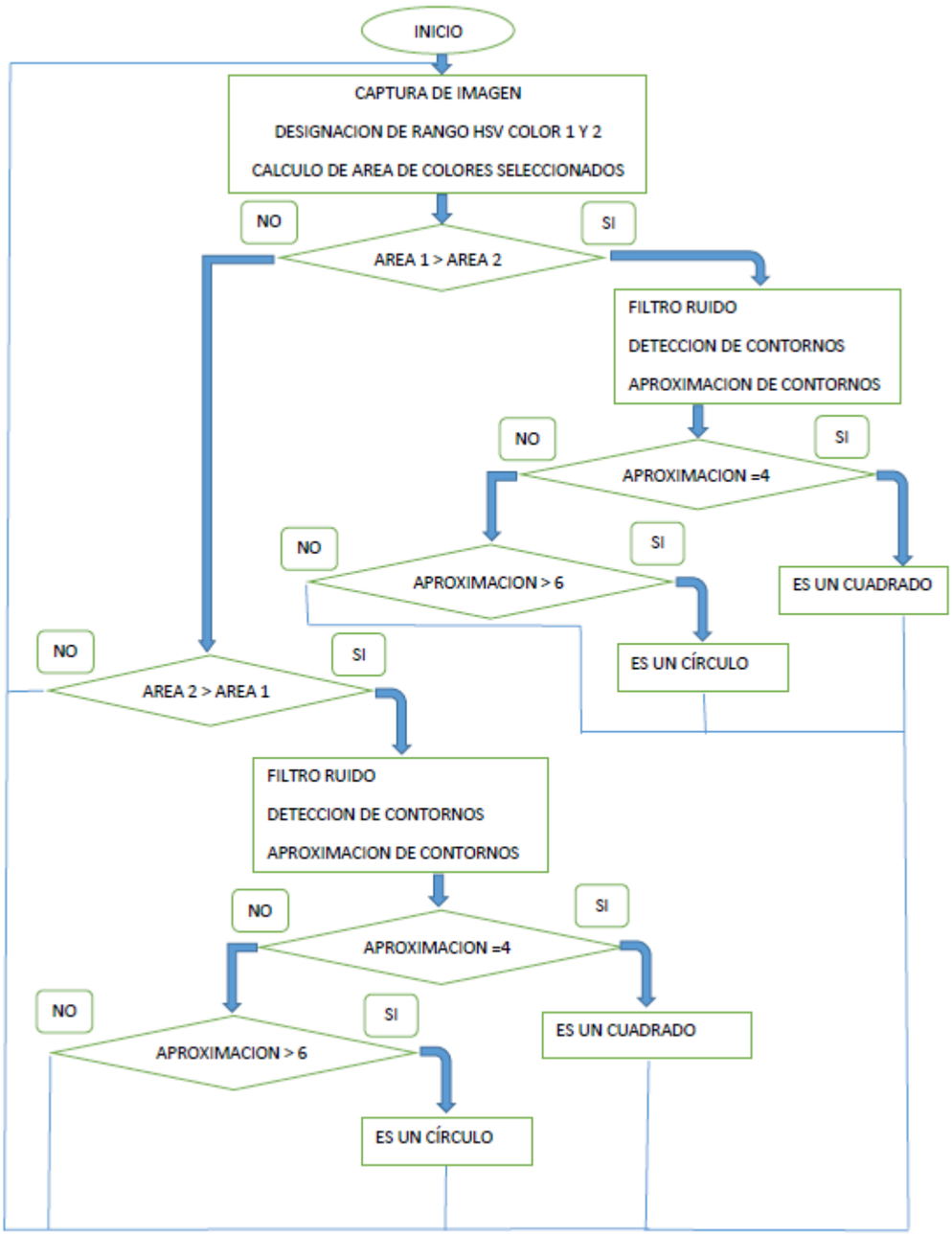

Figura 2-25: Diagrama de flujo programación Python.

Fuente: imagen elaboración propia

2.4 PUERTOS GPIO

GPIO (General Purpose Input/Output) es como su nombre lo indica, un sistema de entradas y salidas de propósito general, es decir, consta de una serie de pines o conexiones que se pueden usar como entradas o salidas para múltiples usos. Estos puertos representan una interfaz entre la Raspberry Pi y el mundo exterior, con ellos es posible realizar una gran multitud de proyectos, desde los más simples a unos con mayor grado de sofisticación.

Otro de los puntos importantes a tener en cuenta es que todos los pines son de tipo "unbuffered", es decir, no disponen de buffers de protección, así que se debe tener cuidado con las magnitudes de voltajes e intensidades cuando se conecten componentes a ellos para no dañar la placa, ya que no todos los pines tienen la misma función.

2.4.1 Función de Pines por Color

Los pines GPIO tienen funciones específicas, aunque algunos comparten funciones y se pueden agrupar de la siguiente manera:

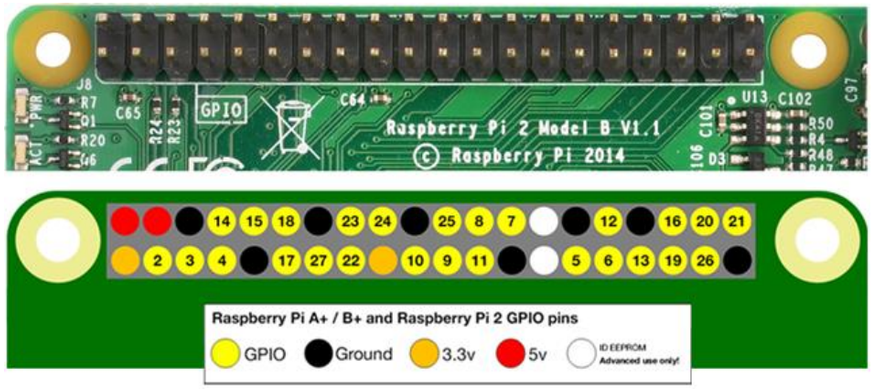

Figura 2-26: Pines GPIO en Raspberry

Fuente: www.programoergosum.com

Color	Cantidad	Descripcion
Amarillo	02	Alimentación a 3.3Vdc.
Rojo	02	Alimentación a 5Vdc.
Naranja	26	Entradas y salidas de propósito general. Pueden configurarse como entradas o salidas. Se debe tener presente que el nivel alto de tensión es de 3.3Vdc y no son tolerantes a tensiones de 5Vdc.
Gris	02	Reservados.
Negro	08	Conexión a GND o masa.
Azul	02	Comunicación mediante el protocolo I2C para comunicarse con periféricos que siguen este protocolo.
Verde	02	Destinados a conexión para UART para puerto serie convencional.
Morado	05	Comunicación mediante el protocolo SPI para comunicarse con periféricos que siguen este protocolo.

Tabla 2-1: Descripción de puertos GPIO por colores.

Fuente: www.programoergosum.com

- **Pines de alimentación:** La placa de la Raspberry tiene pines de 5 Vdc y 3,3 Vdc (limitados a 50mA) y tierra (GND o Ground), que aportan alimentación para los circuitos. Estos pueden servir como una fuente de alimentación, aunque también se pueden utilizar otras fuentes (pilas, fuentes de alimentación externas, etc).

- **GPIO Normales:** Son conexiones configurables que se pueden programar de acuerdo al tipo de proyecto.

- **GPIO Especiales:** Dentro de éstos se encuentran algunos pines destinados a una interfaz UART, con conexiones TXD y RXD que sirven para comunicaciones en serie, como por ejemplo, conectar con una placa arduino.

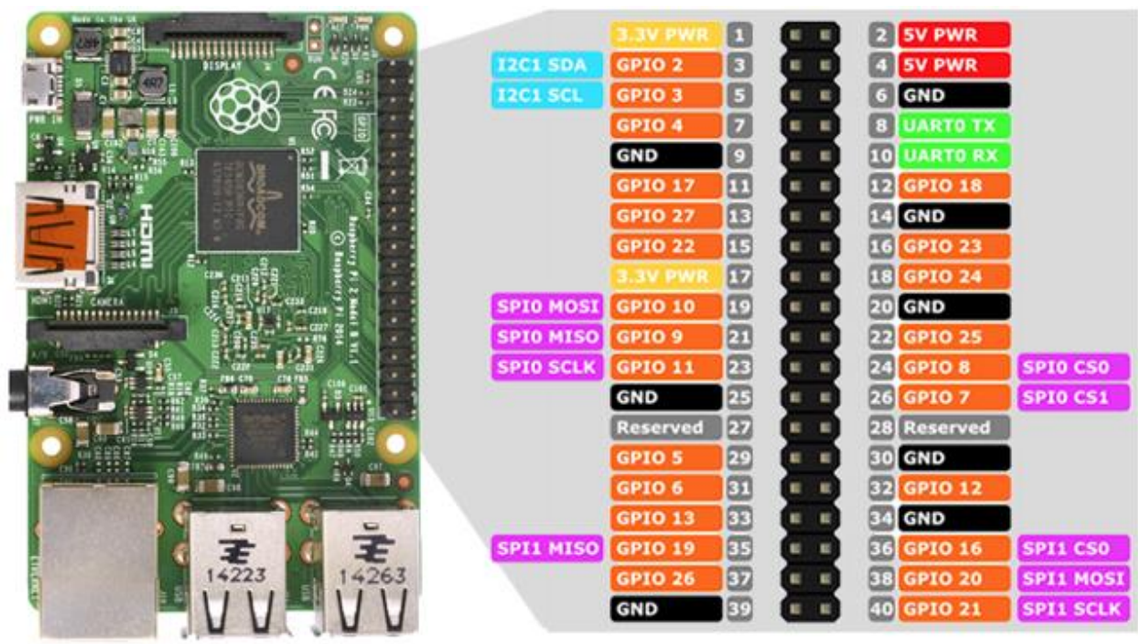

Figura 2-27: Pines GPIO para Raspberry Pi 3 Modelo B+

Fuente: www.programoergosum.com

Existen 2 formas de numerar los pines de la Raspberry Pi, en modo GPIO o en modo BCM.

- En el **modo GPIO**, los pines se numeran de forma física por el lugar que ocupan en la placa (representados por el color gris) viene siendo igual para todas las versiones (comenzamos a contar desde arriba a la izquierda y finalizamos abajo a la derecha).

- En el **modo BCM**, los pines se numeran por la correspondencia en el chip Broadcom (que es la CPU de la Raspberry Pi).

Por este mismo motivo se puede encontrar 2 nomenclaturas a la hora de realizar las prácticas de electrónica con Raspberry Pi, esto puede ser el modo GPIO o el modo BCM. A continuación se muestra una tabla de equivalencias.

BOARD	GPIO		GPIO	BOARD
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figura 2-28: Pines GPIO

Fuente: www.programoergosum.com

2.5 PROGRAMACIÓN DE BRAZO ROBÓTICO

2.5.1 Programación con Melfa Basic V

En este lenguaje la programación se estructura como un conjunto de instrucciones cuyo flujo de proceso se realiza en un lenguaje BASIC estándar.
El aspecto de un programa es un conjunto de instrucciones propias del sistema de Robot entre sentencias ya conocidas de BASIC. Se obtiene así una forma intuitiva de programación, sencilla incluso para aquellos usuarios con pocos conocimientos de BASIC.

Caracteres con significado especial

Apóstrofe (´)

Las líneas de comentarios están indicadas con apóstrofes, y serán transferidas también a la drive unit.

Ejemplo: 100 ´ posición de inicio

Asterisco (*)

El asterisco define marcas de salto (etiquetas). No serán transferidas a la drive unit

Ejemplo: 110 *TABLA1

Coma (,)

La coma sirve de separador cuando se especifican muchos parámetros consecutivos.

Ejemplo: 100 P50 = (450,100,300, ...)

Punto (.)

Para datos múltiples, como los datos posicionales, el punto sirve como separador de cada componente singular.

Ejemplo: 110 M10 = P10.X

2.5.2 Declaración de variables

Los nombres de variables del tipo de posición, articulación (joint), aritmética y cadena de caracteres, empiezan con un carácter particular.

La norma es:

P = Positional (variable de posición)
J = Joint (articulaciones)
M = Aritméticas
C = Character string (cadena de caracteres)

2.5.3 Funciones de movimiento

MOV: Movimiento interpolación de ejes. Esta instrucción mueve a un punto determinado mediante interpolación de ejes. La trayectoria de un punto a otro no es lineal, es decir no describe una línea recta en el espacio, sino que la CPU procesa y mueve los ejes a su conveniencia, por su camino más sencillo. Por lo tanto la trayectoria no es 100% predecible por el usuario y puede producir colisiones.

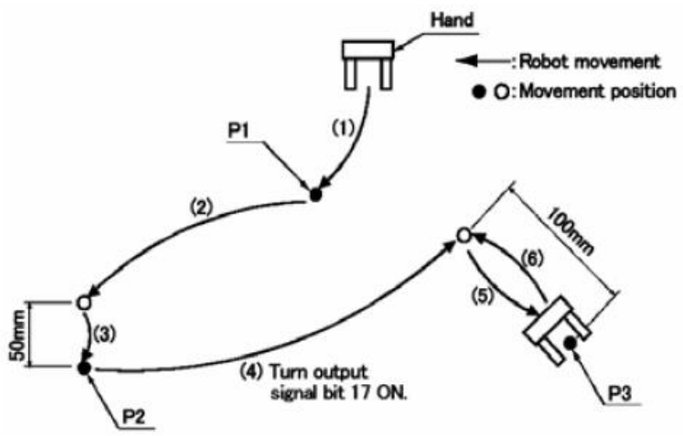

Figura 2-29: Movimiento de interpolación de ejes

Fuente: Manual de Brazo Robótico.

10 MOV P1	Mueve hacia P1
20 MOV P2,-50	Mueve respecto a P2, 50mm atrás de la posición de la mano.
30 MOV P2	Mueve hacia P2
40 MOV P3,-100 WTH M_OUT(17)=1	Mueve respecto a P3, 100mm atrás mientras activa la salida bit nº 17
50 MOV P3	Mueve hacia P3
60 MOV P3,-100	Mueve respecto a P3, 100mm atrás
70 END	Fin de programa

Tabla 2-2: Ejemplo de movimientos de interpolación de ejes.

Fuente: Manual de Brazo Robótico.

MVS: Movimiento en interpolación lineal. Esta instrucción mueve a un punto determinado mediante interpolación lineal. La trayectoria de un punto a otro es lineal, es decir, describe una línea recta en el espacio.

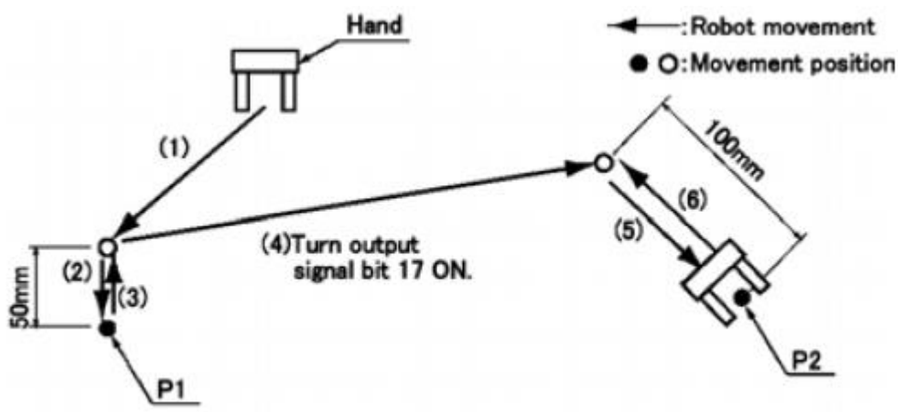

Figura 2-30: Movimiento en interpolación lineal

Fuente: Manual de Brazo Robótico.

10 MVS P1,-50	Mueve respecto a P2, 50mm atrás en línea recta
20 MVS P1	Mueve hacia P1 en línea recta
30 MVS,-50	Mueve 50mm atrás desde la posición actual en línea recta
40 MVS P2,-100 WTH M_OUT(17)=1	Mueve respecto a P2, en línea recta 100mm atrás, mientras activa la salida bit nº 17
50 MVS P2	Mueve hacia P2, en línea recta
60 MVS,-50	Mueve respecto a P2, en línea recta 50mm atrás.
70 END	Fin de programa

Tabla 2-3: Ejemplo de movimientos de interpolación lineal.

Fuente: Manual de Brazo Robótico.

2.5.4 Definición de Velocidades

ACCEL: Designa aceleración y deceleración en % respecto a la máxima permitida

OVRD: Designa la velocidad de trabajo del robot en %.

1 OVRD 100	fija la velocidad de trabajo al 100%
2 MOV P1	Mueve a P1 en línea recta
3 OVRD 50	fija la velocidad de trabajo al 50%
4 MVS P2	Mueve a P2 en línea recta
5 OVRD 100	velocidad general al 100%
6 ACCEL 70,70	Aceleración y desaceleración a 70%
7 MOV P1	Mueve a P1

8 ACCEL	fija la aceleración y deceleración al 100%
9 END	Fin de Programa

Tabla 2-4: Ejemplo de ajustes de velocidad de trabajo del robot.

Fuente: Manual de Brazo Robótico.

2.5.5 Control de Gripper

HOPEN: Abre la pinza designada

HCLOSE: Cierra la pinza designada

Ejemplo:

1 HCLOSE 1	Cierra la pinza y/o gripper (atrapa pieza)
2 HOPEN 1	Abre la pinza y/o gripper (suelta pieza)

Tabla 2-5: Ejemplo de Abrir y Cerrar Gripper.

Fuente: Manual de Brazo Robótico.

2.5.6 Comandos de control de programa

Estos comandos realizan las mismas funciones que el BASIC estándar, y sirve para transferir el control del programa a líneas determinadas de éste, condicionalmente a un caso particular o incondicionalmente.

GOTO: Salto incondicional a línea

Ejemplo:

GOTO <línea o label>

GOTO 200	Salta a línea numero 200
GOTO *FINAL	Salta a línea marcada como * FINAL

Tabla 2-6: Ejemplo On Goto (Salto Condicional a Linea Designada)

Fuente: Manual de Brazo Robótico.

ON GOTO: Salto condicional a línea designada por una variable entera. El programa seguirá el valor de orden de esta variable (0, 1, 2, 3, 4…)

Ejemplo:

ON <Variable entera> GOTO<destino><destino><destino>

ON M1 GOTO 100,200,300	SI M1=1 salta a línea 100, Si M1=2 salta a línea 200 y si no corresponde salta a la línea siguiente

Tabla 2-7: Ejemplo On Goto (Salto Condicional a Linea Designada)

Fuente: Manual de Brazo Robótico.

IF THEN ELSE: Salto condicionado, si no se da la circunstancia se ejecuta el salto designado en **ELSE**.

Ejemplo:

IF <condición> THEN <línea> ELSE <línea o label>

IF M1=1 THEN 130	Salta a 130 si M1=1
IF M1=1 THEN 130 ELSE 150	Salta a 130 si M1=1, si no salta a 150

Tabla 2-8: Ejemplo ciclo If Then Else.

Fuente: Manual de Brazo Robótico.

SELECT CASE: Salto condicional, según la condición se ejecuta lo designado en **CASE**.

Ejemplo:

SELECT M1	
CASE 10	Si M1=10 ejecuta sólo las líneas entre CASE 10 y CASE IS 11
CASE IS 11	Si M1=11 ejecuta sólo las líneas entre CASE IS 11 y CASE IS <5
CASE IS < 5	
CASE 6 TO 8	Ejecuta si está entre 6 y 8
DEFAULT	corresponde al grupo de instrucciones que se ejecuta cuando ninguno de los casos anteriores se ha cumplido
END SELECT	

Tabla 2-9: Ejemplo de Select Case (Salto Condicional).

Fuente: Manual de Brazo Robótico.

WAIT: Espera en esta línea hasta que la condición ha sido alcanzada

WHILE-WEND: Repite las sentencias comprendidas entre WHILE y WEND hasta que se cumpla una condición determinada.

Ejemplo:

WHILE <variable condición>
(Sentencias)
WEND

WHILE (M1>=1)AND(M1<=10)	Las líneas entre 10 y 60 se repetirán HASTA que M1 esté entre 1 y 10
60 WEND	

Tabla 2-10: Ejemplo ciclo While.

Fuente: Manual de Brazo Robótico.

2.5.7 Paro Incondicional y retardos

HLT: Para el programa en aquel punto

10 IF M_IN(20)THEN HLT	Detiene el programa si la entrada 20 es ON

Tabla 2-11: Ejemplo de paro Incondicional

Fuente: Manual de Brazo Robótico.

DLY: Establece un retardo

2.5.8 Entradas y salidas

Entradas

<variable>=M_IN(<bit>)
<variable>=M_INB(<byte>)
<variable>=M_INW(<word>)

Salidas

M_OUT(<bit>)=<1/0>
M_OUT(<byte>)=<byte>
M_OUT(<word>)=<word>

M1=M_INB(20)	Bits OUT 20 a 27 pasan a M1
WAIT M_IN(3)=1	Espera hasta que bit IN 3 es ON
M_OUT(1)=1 DLY 0.5	conmuta bit OUT 1 a ON durante 0.5S

Tabla 2-12: Ejemplos de entradas y salidas

Fuente: Manual de Brazo Robótico.

2.6 PROGRAMACIÓN DEL BRAZO ROBÓTICO CON CIROS STUDIO

Para realizar la programación del brazo robótico Melfa RV-2SDB, a través de Ciros Studio, se deben ejecutar una serie de pasos fundamentales antes de iniciar dicho proceso, por lo cual a continuación se utiliza una serie de imágenes las cuales permitirán explicar y comprender de mejor forma dicho proceso.

2.6.1 Creación de un nuevo proyecto

Para crear un nuevo proyecto lo primero que se debe realizar es abrir la aplicación de Ciros Studio y realizar la creación de este, para lo cual se debe ir a File_New_Project Wizard.

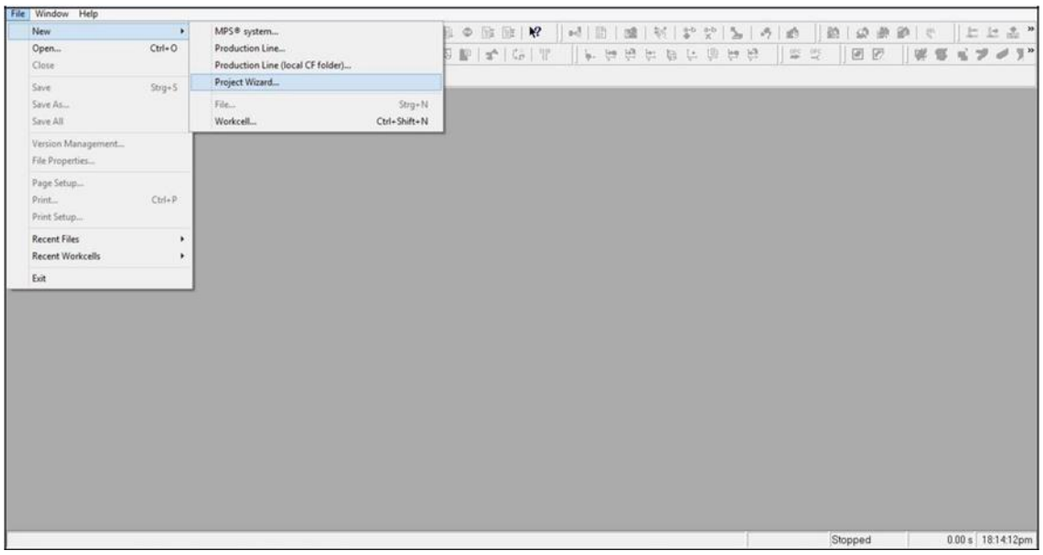

Figura 2-31: Crear un nuevo Proyecto

Fuente: Manual de Ciros Robotics

Al seleccionar la creación de un nuevo proyecto se debe indicar el título de este, el cual puede llevar cualquier nombre definido por el usuario, el que también

indicara el nombre con el cual se guarda en el directorio. Otras de las opciones que permite este cuadro es introducir el nombre del creador del proyecto, una identificación de usuario con iniciales y una pequeña descripción del nuevo proyecto que se realizara.

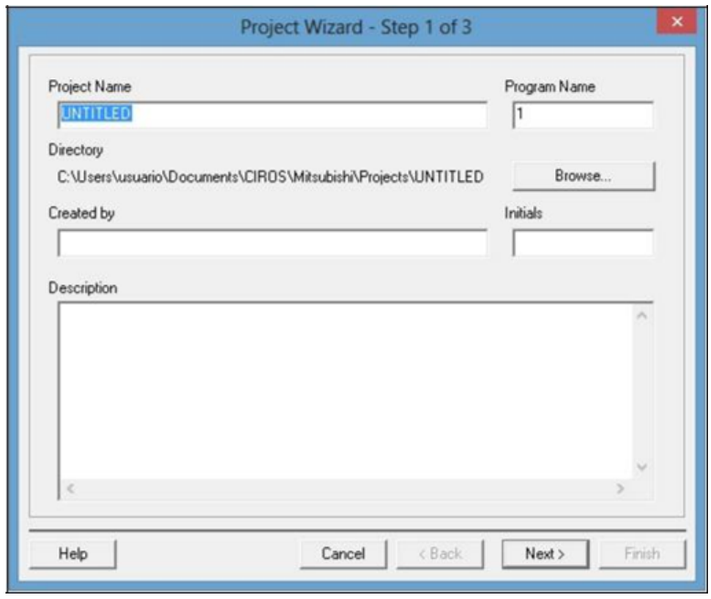

Figura 2-32: Nombre del Nuevo proyecto

Fuente: Manual de Ciros Robotics

Una vez establecido el nombre del proyecto se avanza al siguiente paso, el cual permite seleccionar el tipo de robot que se requiere programar, en este caso es el robot RV-2SD/SDB/SQ de seis ejes articulados. Otro de los puntos relevantes, es la selección del lenguaje de programación a utilizar, el cual es MELFA V.

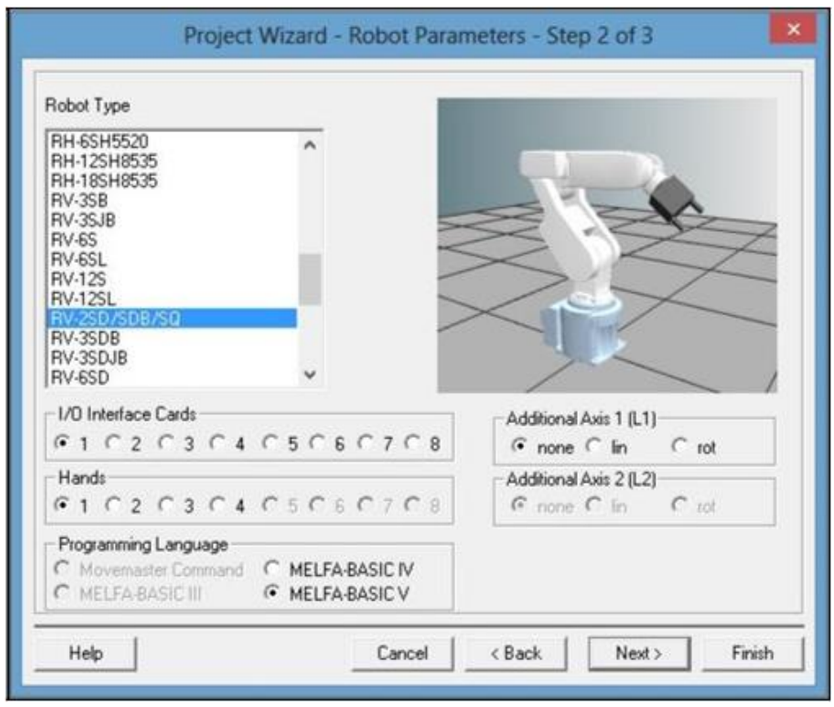

Figura 2-33: Selección del tipo de robot y lenguaje de programación.

Fuente: Manual de Ciros Robotics

2.6.2 Configuración de comunicación

Para establecer comunicación entre el brazo y el Pc, se debe verificar en conexión de área local y propiedades que este configurado con una dirección IP diferente a la del robot, pero que se encuentre dentro del rango de la misma mascara de red.

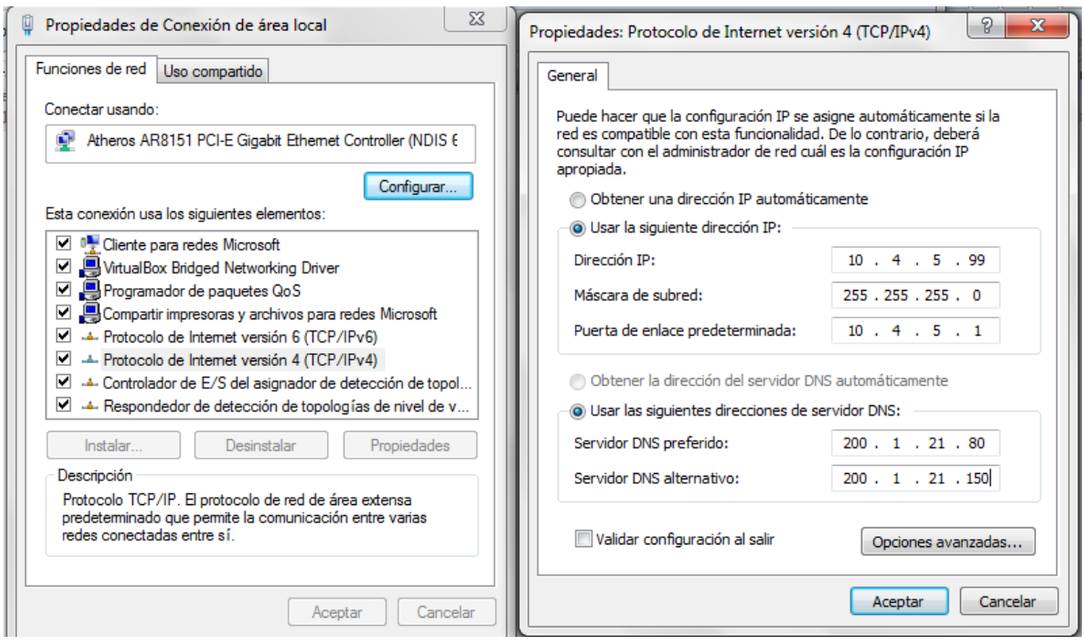

Figura 2-34: Configuración de máscara de red

Fuente: Imagen de elaboración propia

Para poder iniciar la conexión con el robot es necesario abrir RCI Explorer y con el botón derecho del mouse dar clic en Connection y posteriormente en properties.

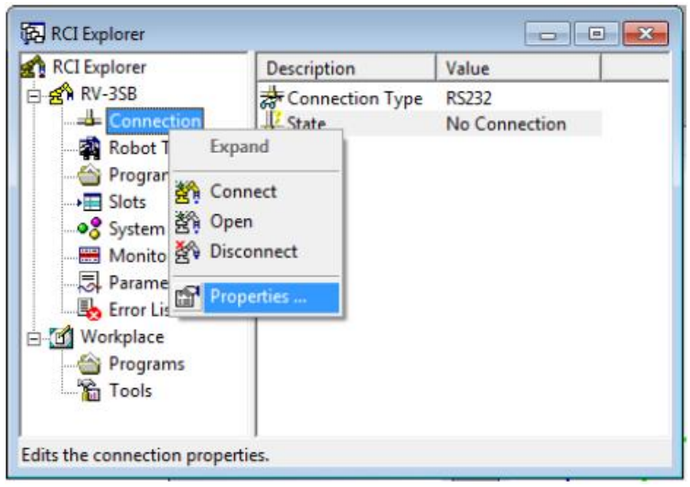

Figura 2-35: Cambio de las propiedades de conexión con el robot

Fuente: www.udb.edu.sv

Una vez realizado lo anterior se abrirá una ventana que permite configurar todos los parámetros de conexión con el robot. En donde se debe especificar la interfaz de comunicación y la dirección del brazo.

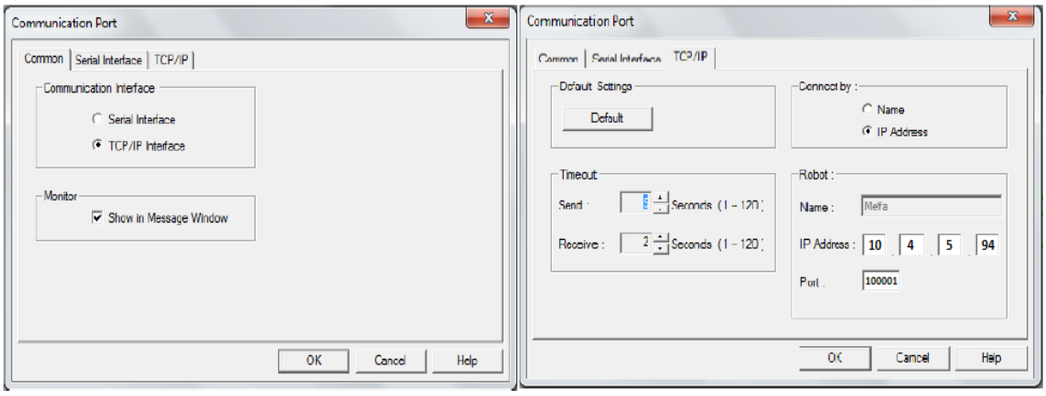

Figura 2-36: Configuración de la conexión con el robot

Fuente: www.udb.edu.sv

Una vez configurada la conexión con el robot se debe regresar a RCI explorer y nuevamente hacer clic derecho en Connection y seleccionar la opción Connect. Se abre un cuadro de dialogo el cual indica si la conexión fue exitosa.

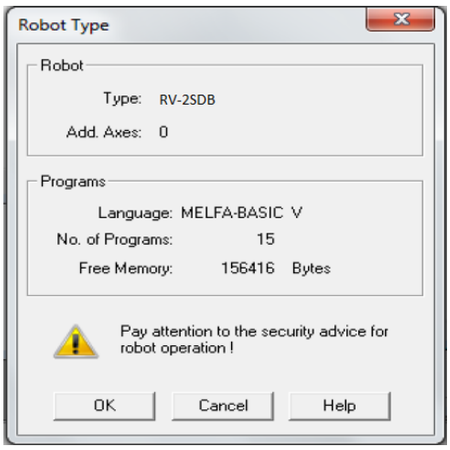

Figura 2-37: Ventana de información de conexión exitosa con el robot.

Fuente: www.udb.edu.sv

2.6.3 Manipulación del brazo Robótico desde Ciros Studio

Una vez establecida la conexión con el brazo robótico, está la opción de realizar movimientos desde el programa Ciros Studio. Para esto es necesario que el controlador se encuentre en modo automático y se debe seleccionar la ventana de operación de marcha por impulsos, Jog Operation. Al utilizar esta opción se debe tener precaución, ya que los movimientos no son tan precisos como cuando se realizan operaciones y programación desde el teachpendant.

Las funciones XYZ, JOINT y TOOL, trabajan de la misma manera que el robot real, por lo que para moverlo virtualmente se hará con la opción que más acomode. Para crear una posición es necesario primero posicionar el robot en la ubicación deseada. Esto se puede ejecutar ajustando el porcentaje de velocidad y los movimientos entre los distintos tipos de ejes del robot, dependiendo de las condiciones y precisión que se requiera. Una vez hecho esto se debe seleccionar Current Position, la cual guarda automáticamente la posición en orden correlativo.
Todo lo anterior debe ser complementado realizando un código de programación que permita interactuar las diferentes posiciones guardadas, con una lista de instrucciones que le indican al controlador cuales son los pasos y/o secuencias de los movimientos que debe realizar el brazo. Una vez hecho todo lo anterior es fundamental cargar las posiciones y el código de programación al controlador del robot, de lo contrario el programa se encuentra vacío y no genera movimientos.

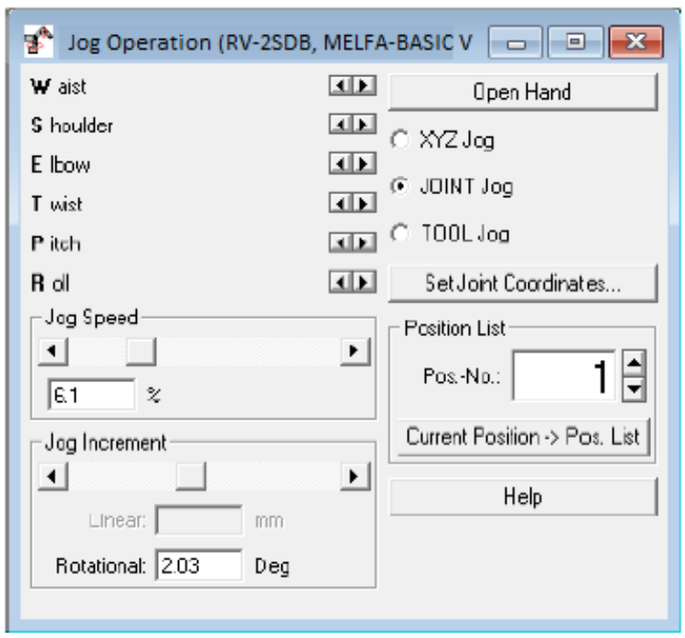

Figura 2-38: Ventana de Operación de Marcha a Impulsos.

Fuente: www.udb.edu.sv

2.6.4 Monitoreo Software Ciros Studio

Las opciones de monitorización del software ciros studio sirven para la supervisión de los diferentes elementos y características del brazo robótico, mientras éste está ejecutando una tarea.

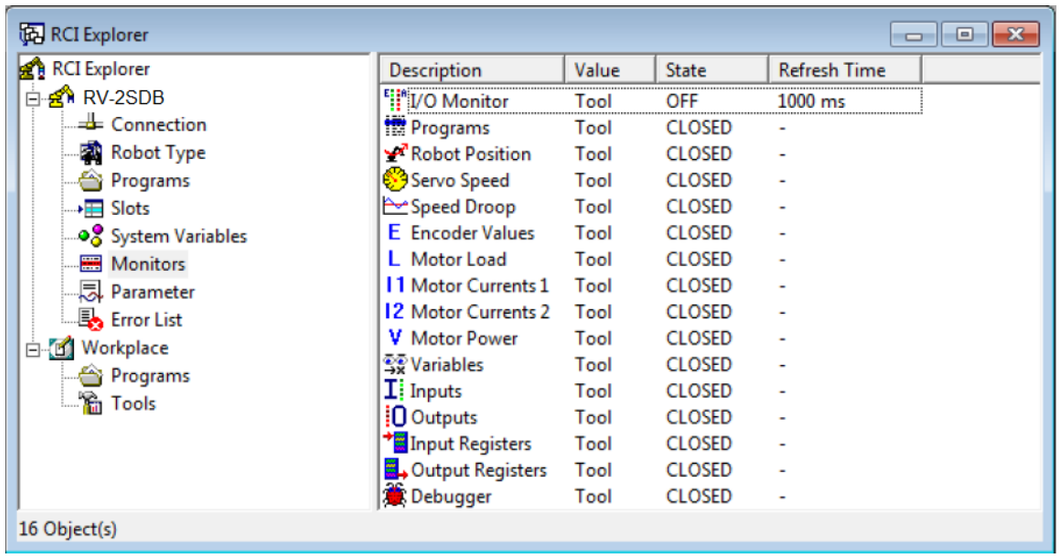

Figura 2-39: Opciones de Monitoreo

Fuente: www.udb.edu.sv

2.6.5 Monitoreo de Programa

Una de las opciones de monitoreo que incluye el software es ejecutar y/o comprobar un código de programa, ya sea de forma secuencial o paso a paso, lo que permite verificar si los movimientos están acordes a las instrucciones ingresadas al controlador. Para realizar esta tarea es necesario abrir la ventana del RCI EXPLORER, seleccionar el programa deseado y mediante clic derecho ir a la opción DEBUG, adicionalmente a esto se deben activar y/o encender los servos motores del brazo robótico. Luego de esto se puede visualizar paso a paso las líneas que se están ejecutando en el programa.

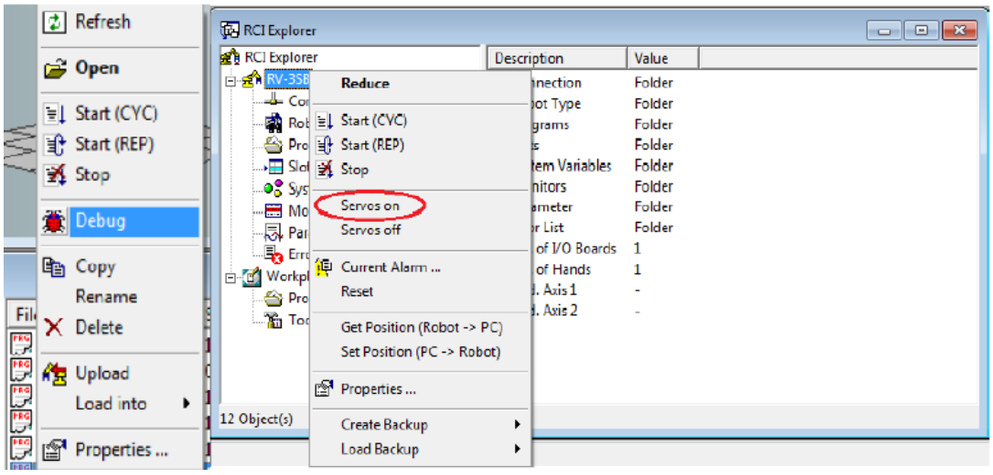

Figura 2-40: Depurador de programa en línea y activación de servomotores del robot

Fuente: www.udb.edu.sv

- **I/O Monitor:** Realiza el Monitoreo de manera visual de las entradas y salidas externas, indicando el estado y dirección de cada uno de los sensores conectados al controlador.

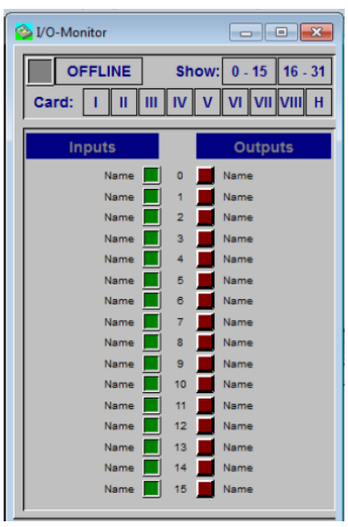

Figura 2-41: Monitor de entradas y salidas

Fuente: www.udb.edu.sv

- **Programs:** Esta opción permite monitorizar el programa online y observar la línea de programa que se está ejecutando en ese momento.

- **Robot Position:** Monitorea las posiciones angulares de cada eje y las posición respecto al eje de coordenadas del robot.

- **Servo Speed:** Con esta opción se puede observar las velocidades de los diferentes servomotores en cada uno de los movimientos del programa mientras se ejecuta.

- **Speed Droop:** Monitorea de manera gráfica la relación entre la medida actual de posición del encoder y el valor máximo que puede llegar a alcanzar.

- **Encoder Values:** Monitorea el valor de los encoders en valor absoluto y relativo de cada eje del robot.

- **Motor Load:** Monitorea de manera gráfica la relación entre la medida actual de carga y el valor máximo de carga de cada eje que puede llegar a alcanzar antes de dar un error.

- **Motor Currents 1:** Monitorea de manera gráfica la relación entre la medida actual de corriente de algunos lazos de corriente 1 de los diferentes servomotores y el valor máximo en cada eje.

- **Motor Currents 2:** Monitorea de manera gráfica la relación entre la medida actual de corriente de algunos lazos de corriente 2 de los diferentes servomotores y el valor máximo en cada eje.

- **Motor Power:** Monitorea de manera gráfica la relación entre la medida actual de tensión de cada motor y el valor máximo antes de generar un error.

- **Variables:** Se pueden monitorear las diferentes variables que se han definido en el programa.

- **Inputs:** Cada Monitor de Entrada puede monitorear 16 valores de entrada binaria sucesivos al mismo tiempo.

- **Outputs:** Cada Monitor de Salida puede monitorear 16 valores de salida binaria sucesivos al mismo tiempo.

- **Inputs Registers:** Cada Monitor de Registros de Entrada puede monitorear 16 registros de entrada sucesivos al mismo tiempo.

- **Outputs Registers:** Cada Monitor de Registros de Salida puede monitorear 16 registros de salida sucesivos al mismo tiempo.

- **Debugger:** El Depurador le permite directamente revisar la funcionalidad de programas individuales.

2.6.6 Encendido de la unidad de control manual

Al encender la unidad de control manual del robot (Controlador), se debe realizar antes, la inspección de cada una de sus conexiones, principalmente de la unidad de programación portátil con la cual se opera el brazo robótico de modo manual, ya sea para guardar posiciones de manera más precisa y no cometer errores, como para familiarizarse con las funciones del robot y sus capacidades.

Si la conexión de la consola portátil se realiza con el controlador del robot encendido, éste arroja un mensaje de error y un sonido de alerta, indicando que no está permitido la conexión de dispositivos con el controlador ya encendido.

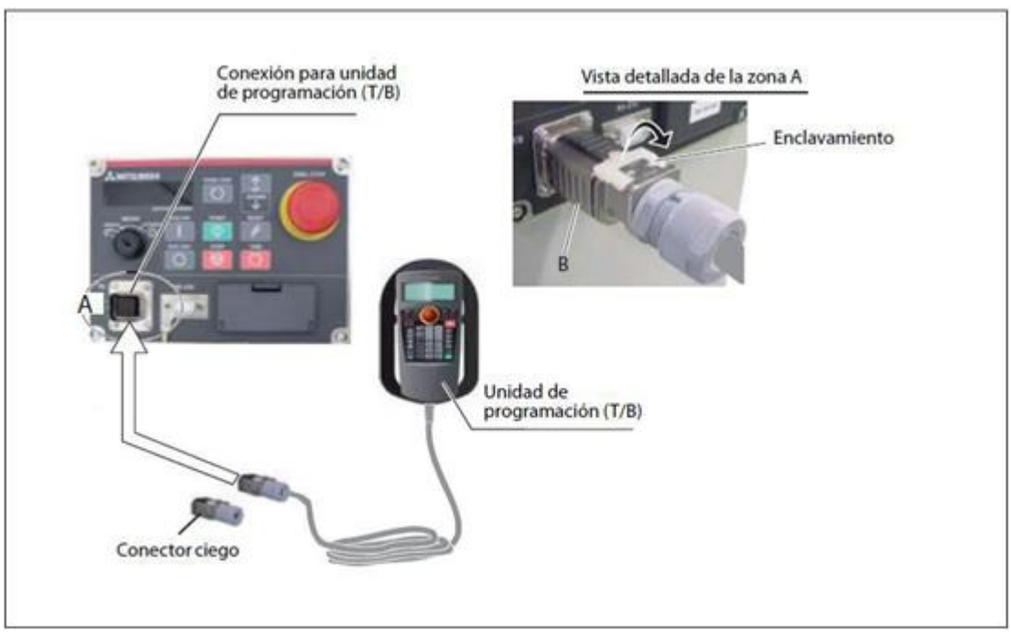

Figura 2-42: Encendido de la unidad de programación manual.

Fuente: meltrade.hu

Una vez encendido el controlador se debe tener precaución con las paradas de emergencia con las que este cuenta y también con las de la unidad de programación portátil, ya que si se encuentra alguna pulsada y/o presionada, arroja un error y comienza a sonar la alarma del controlador. Por esta razón, antes de encender el controlador se debe verificar el modo en el que este inicializara, ya que si se encuentra en modo manual con el teaching box conectado, pero no habilitado,

se encenderá una alarma sonora, y en el caso que el control manual se encuentre habilitado no generara ningún problema.

Una vez encendido el controlador y el teaching box conectado correctamente, éste se puede habilitar o deshabilitar desde la unidad de programación del robot, dependiendo de cómo requiera trabajar el usuario, para esto en la parte posterior del teaching box, se encuentra el botón "ENABLE/DISABLE", el cual al estar habilitado encenderá una luz de estado, permitiendo mover el robot de manera manual, realizar diferentes tipos de funciones y/o trabajos, guardar posiciones y realizar programas a partir de las posiciones guardadas, entre otras. En el caso contrario al estar deshabilitado, la luz de estado se encuentra apagada.

Si se requiere trabajar con el controlador en modo "AUTOMÁTICO", es posible realizarlo directamente desde el computador utilizando el programa ciros studio o utilizando la memoria del controlador con programas almacenados anteriormente desde el teaching box o simplemente desde el computador.

Para corregir algún error y/o alarma del controlador, se debe presionar el botón "RESET" de la unidad, ya sea un error ocasionado en el controlador o un error desde el teaching box.

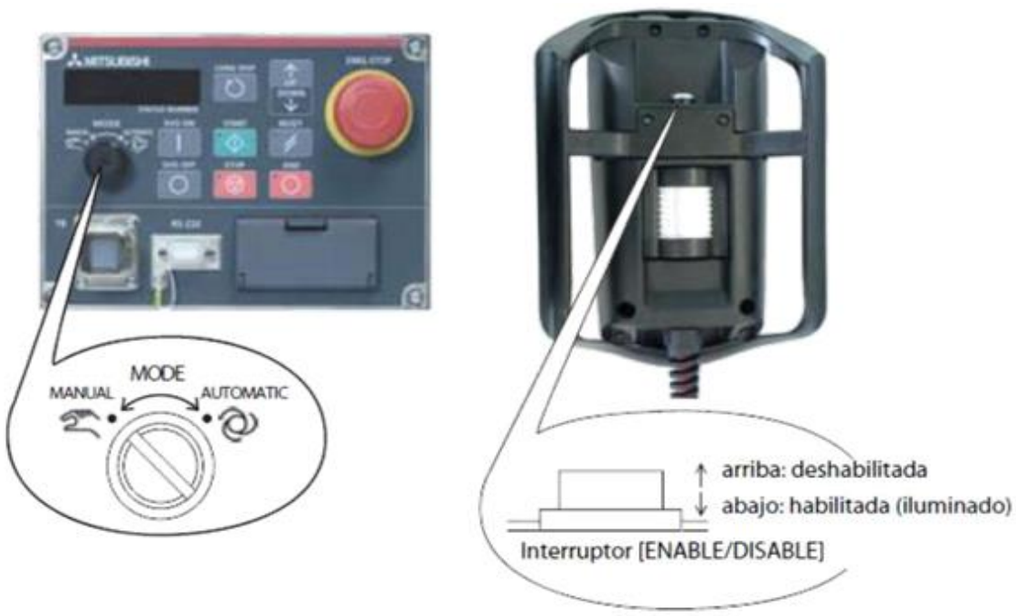

Figura 2-43: Selección de modo Automatico/Manual y Habilitación teaching box.

Fuente: meltrade.hu

2.6.7 Encendido de Servos y Enclavamiento de Control

Una vez habilitado el control manual, a través del botón Enable/Disable de la consola portátil del controlador, es necesario realizar el encendido independiente de

los servo motores, para esto existe un switch llamado "DEADMAN SWITCH" el cual se encuentra ubicado en la parte trasera del teaching box y permite realizar un enclavamiento manual de control. Para esto es necesario llevar el switch a la posición central y luego encender los servos motores en la parte frontal de la consola portátil y/o teaching box.

Figura 2-44: Enclavamiento de control para encendido de servos.

Fuente: meltrade.hu

2.6.8 Opciones del Menú y/o Pantalla

Una vez realizado todos los pasos anteriores, en la pantalla del teaching box se visualiza un mensaje con las características del software, el modelo del brazo robótico y las características del fabricante.

Para abrir el menú principal en la consola portátil, se debe presionar la tecla [EXE] (tecla verde del teaching box), la cual direcciona al menú e indica diferentes tipos de opciones al usuario.

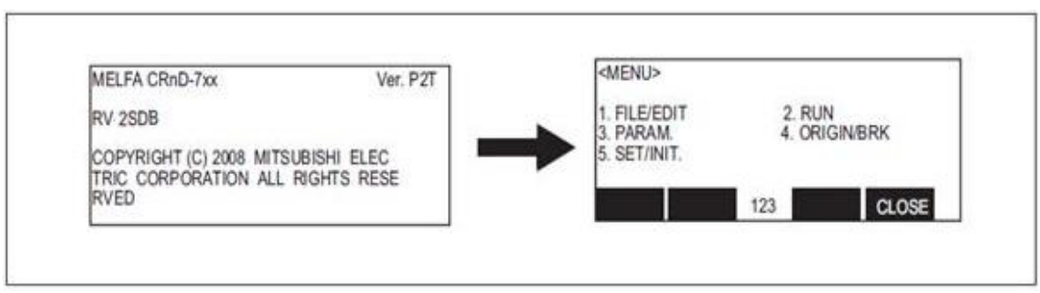

Figura 2-45: Menú de la consola portátil.

Fuente: meltrade.hu

2.6.9 Operación del Brazo Robótico en Modo JOG.

Para comenzar a mover el brazo robótico, se debe presionar la tecla "JOG", que se encuentra en la primera columna de la unidad de programación manual teaching box. Al seleccionar esta opción se visualiza en pantalla la función en la que se encuentra la consola portátil y cuál es el modo de operación seleccionado para mover las articulaciones.

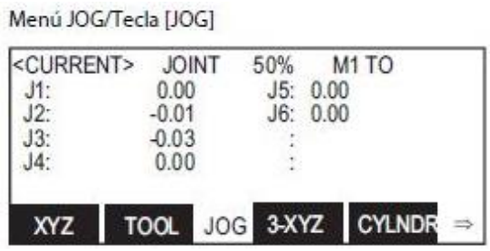

Figura 2-46: Operación en modo JOG.

Fuente: meltrade.hu

Si por alguna razón y/o condición se requiere cambiar de modo de operación de las articulaciones, existen 5 modos en total, pero con diferentes perspectivas y principios de funcionamiento distintos.

- **Modo JOG de Articulación**
  En el modo Jog de articulación, es posible mover individualmente los ejes del robot. Para tal movimiento es posible realizar un ajuste independiente de los ejes J1 hasta J6 y de los ejes adicionales J7 y J8. El número de ejes depende del modelo del robot y el control de los ejes adicionales se realiza mediante las teclas J1 y J2.

- **Modo JOG de Herramienta**
  En el modo Jog de herramienta es posible mover la posición de la punta de la pinza manipuladora a lo largo de los ejes en el sistema de coordenadas de herramienta.

  La punta de la pinza manipuladora se mueve linealmente. La posición del robot puede girarse en torno a los ejes X, Y y Z del sistema de coordenadas de herramientas, sin modificar la posición de la punta de la pinza manipuladora. El centro de la herramienta debe definirse mediante el parámetro MEXTL.

En el robot de brazo articulado vertical, la dirección de la brida de la pinza manipuladora hacia la punta de la pinza manipuladora está definida como +Z.

Modo JOG de XYZ.

En el modo Jog de XYZ es posible mover la posición de la punta de la pinza manipuladora a lo largo de los ejes en el sistema de coordenadas XYZ.

La posición del robot puede girarse en torno a los ejes X, Y y Z del sistema de coordenadas XYZ, sin modificar la posición de la punta de la pinza manipuladora. El centro de la herramienta debe definirse mediante el parámetro MEXTL.

Modo JOG de XYZ en 3 Ejes.

En el modo Jog de XYZ en 3 ejes es posible mover la posición de la punta de la pinza manipuladora a lo largo de los ejes en el sistema de coordenadas XYZ.

A diferencia del modo JOG de XYZ, la posición del robot, al igual que en modo JOG de articulación, se modifica girando los ejes J4, J5 y J6. Si la posición de la punta de la pinza manipuladora está definida a un valor fijo, la posición del robot se interpola mediante los ejes X, Y, Z, es decir la posición del robot no es constante. El centro de la herramienta debe definirse mediante el parámetro MEXTL.

Modo JOG de Círculo.

En el modo Jog de círculo es posible mover la posición de la punta de la pinza manipuladora en círculo en torno al origen.

Una variación de la coordenada del eje X provoca un movimiento radial de la punta de la pinza manipuladora a partir del centro del robot. Una variación de la coordenada del eje Y provoca el mismo movimiento que el control del eje J1 en el modo Jog de articulación.

Una variación de la coordenada del eje Z provoca un movimiento de la pinza manipuladora en la dirección Z como en el modo JOG de XYZ. En el caso de variación de las coordenadas del eje A, B o C, el giro de la pinza manipuladora se realiza como en el modo JOG de XYZ.

2.6.10 Ajustes de Velocidad

En el display de la consola portátil y/o teaching box se puede observar una unidad numérica indicando un porcentaje [100%]. Este valor se refiere a la velocidad de trabajo del robot, basándose en la rapidez con la que ejecutara sus movimientos con respecto a la máxima velocidad que se le pueda emplear, pudiéndose modificar

con las teclas llamadas [OVRD], [OVRD↓] encontradas en la unidad de programación del robot.

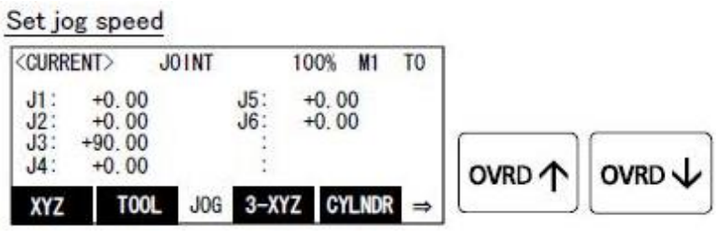

Figura 2-47: Ajustes de Velocidades de trabajo del Robot.

Fuente: meltrade.hu

2.6.11 Crear un Nuevo Programa o Editar uno Existente.

Para crear o modificar un programa de debe seleccionar la opción File/Edit en el menú principal, esta direccionara a otro menú, en el cual es posible comenzar a editar un programa ya existente o crear uno nuevo seleccionando la opción New del display y/o consola portátil.

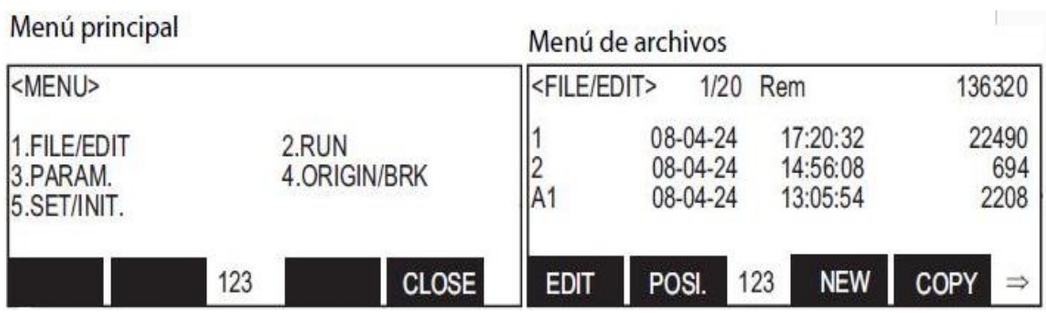

Figura 2-48: Crear un Nuevo Programa o Editar uno Existente.

Fuente: meltrade.hu

En el caso que se requiera crear un nuevo programa, esto se debe realizar seleccionando la opción [New] en la consola de programación, la cual direccionara a la opción de ingresar el nombre del programa a realizar.

Una vez nombrado el nuevo proyecto de programación, hay que ingresarle algún movimiento (puede ser "END") para que el teaching pueda reconocer algún contenido y no lo borre de manera automática.

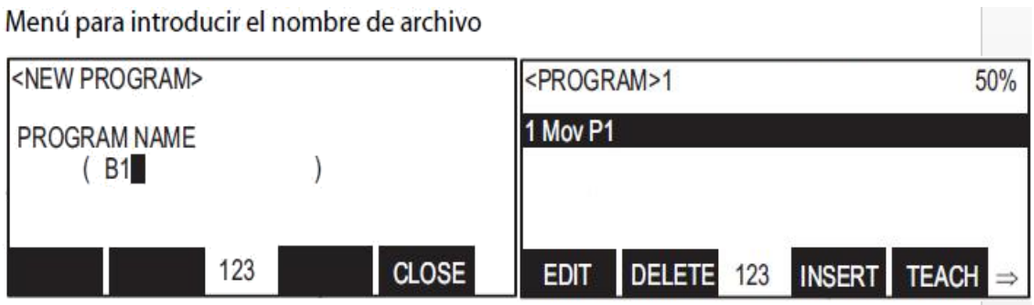

Figura 2-49: Ingresar el nombre del programa.

Fuente: meltrade.hu

2.6.12 Guardar Posiciones desde el Teaching Box.

Para crear y guardar nuevas posiciones en el controlador se debe ingresar al menú de archivos y seleccionar el programa a modificar, una vez hecho esto se debe seleccionar la opción [POSI], la que direcciona al menú para edición de posiciones. Desde esta pantalla se debe ingresar al menú JOG (Presionando la tecla Jog desde la consola portátil), el cual permite realizar movimientos con el brazo robótico y de esta manera seleccionar las diferentes posiciones de trabajo requeridas en el proceso a implementar.

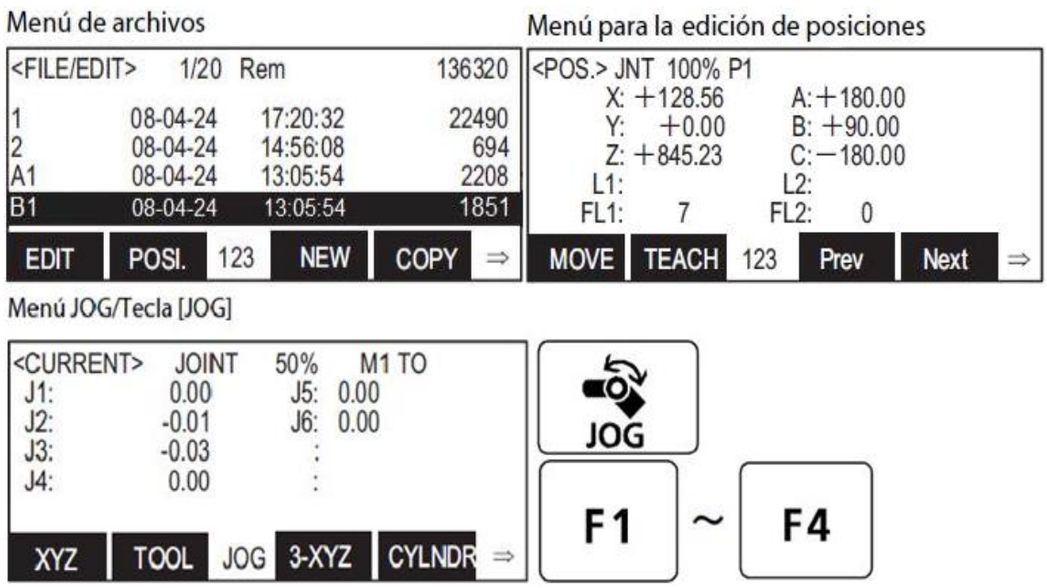

Figura 2-50: Edición de Posiciones desde consola portátil.

Fuente: meltrade.hu

Una vez seleccionada la posición de trabajo que se utiliza en el programa, se debe ingresar a [TEACH] y confirmar el almacenamiento de la posición, la cual más

adelante se convertirá en un movimiento, posterior a esto se debe cerrar el menú para la edición de posición con la opción close.

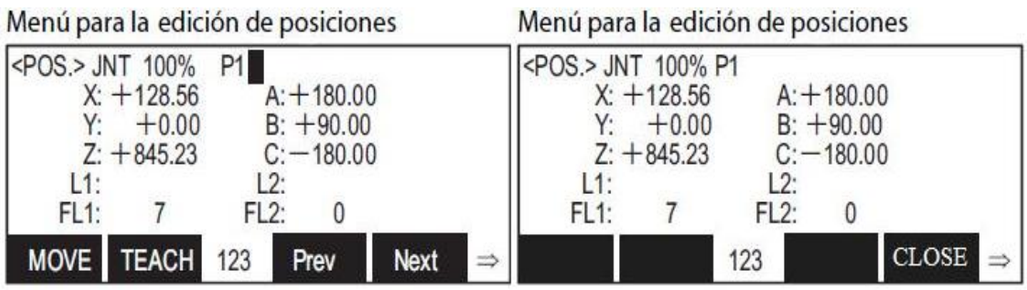

Figura 2-51: Guardar posiciones desde consola portátil.

Fuente: meltrade.hu

2.6.13 Ejecución de un Programa.

Para poder ejecutar un proyecto se debe deshabilitar la unidad de programación con el botón [ENABLE/DISABLE], para así luego pasar a la unidad de control a cambiar la modalidad de trabajo a [AUTOMATIC].

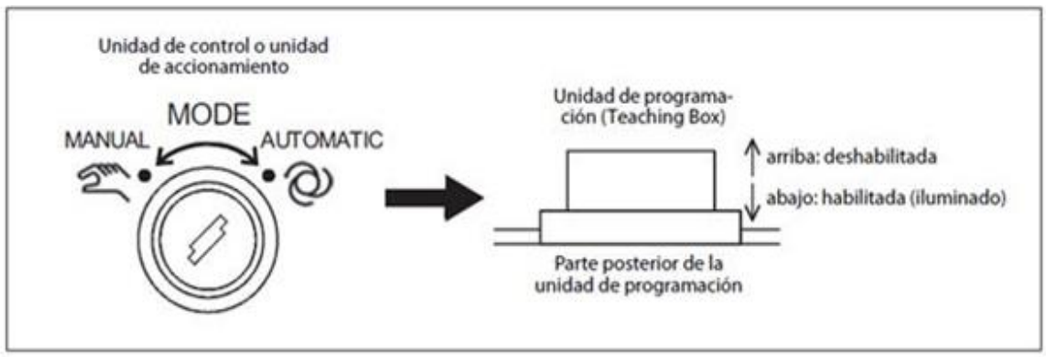

Figura 2-52: Cambio de modo de la unidad de control

Fuente: meltrade.hu

En el modo automático se debe trabajar directamente en el controlador del robot, apoyándose de la pantalla de éste, en la que se visualiza el menú. Con la tecla del panel frontal [CHNG DSP] se busca la opción PROGRAM del menú, posterior a esto se debe buscar el programa que se requiere ejecutar, utilizando las teclas [UP/DOWN]. Luego de haber encontrado el programa se presiona nuevamente la tecla [CHNG DSP], para poder buscar y ajustar la velocidad de trabajo, ya que debe

ser segura, tanto como para el operador, como para que el robot no sufra golpes que le produzca problemas de funcionamiento. El siguiente paso consiste en la activación de los servos, lo cual se realiza en el panel frontal de la unidad de control con la tecla [SERVO]. Habiendo tomado todas las precauciones de operación, para el usuario y el área de trabajo del robot, se presiona la tecla [START] y así dará inicio instantáneamente al proyecto seleccionado.

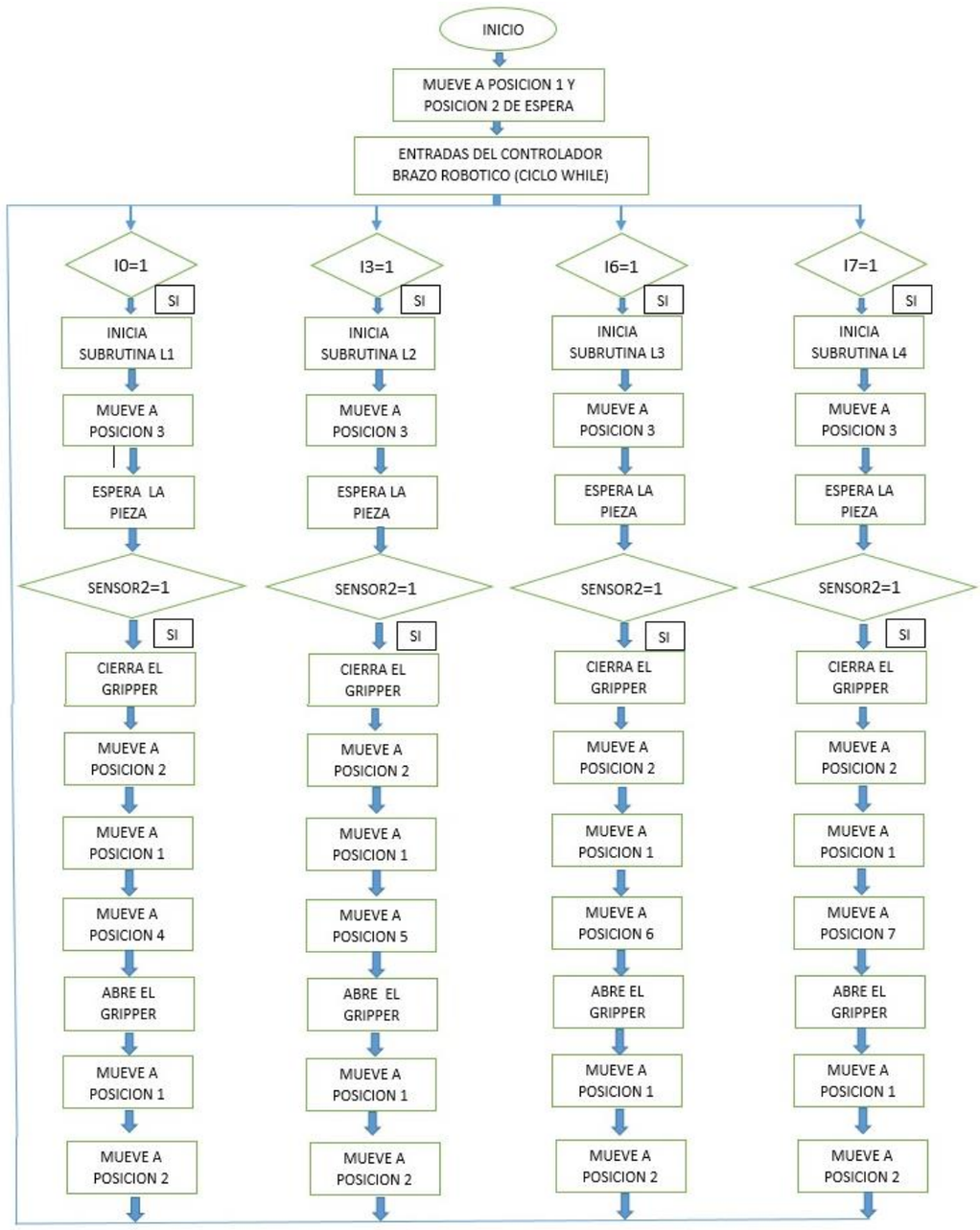

Figura 2-53: Diagrama de flujo funcionamiento programa brazo robotico

Fuente: Imagen de elaboración propia

2.7 INTEGRACIÓN RASPBERRY Y BRAZO ROBÓTICO

Para hacer funcionar en conjunto la Raspberry y el brazo robótico, se realizó a través de las salidas de la Raspberry Pi (Puertos GPIO) y las entradas del controlador del brazo robótico, donde el programa en lenguaje Python funciona de tal manera que al entrar en la sentencia de identificación de las piezas, este activa un puerto GPIO, que a su vez acciona un módulo de relés y estos activan las entradas del controlador del brazo por medio de 24 Vdc, permitiendo la interacción de los circuitos de forma aislada.

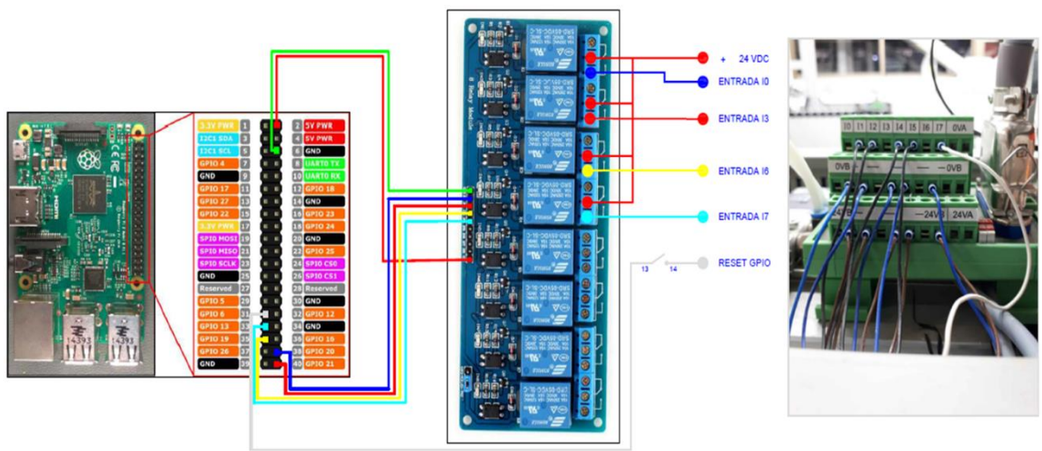

Figura 2-54: Integracion Raspberry brazo robótico

Fuente: Imagen de elaboración propia

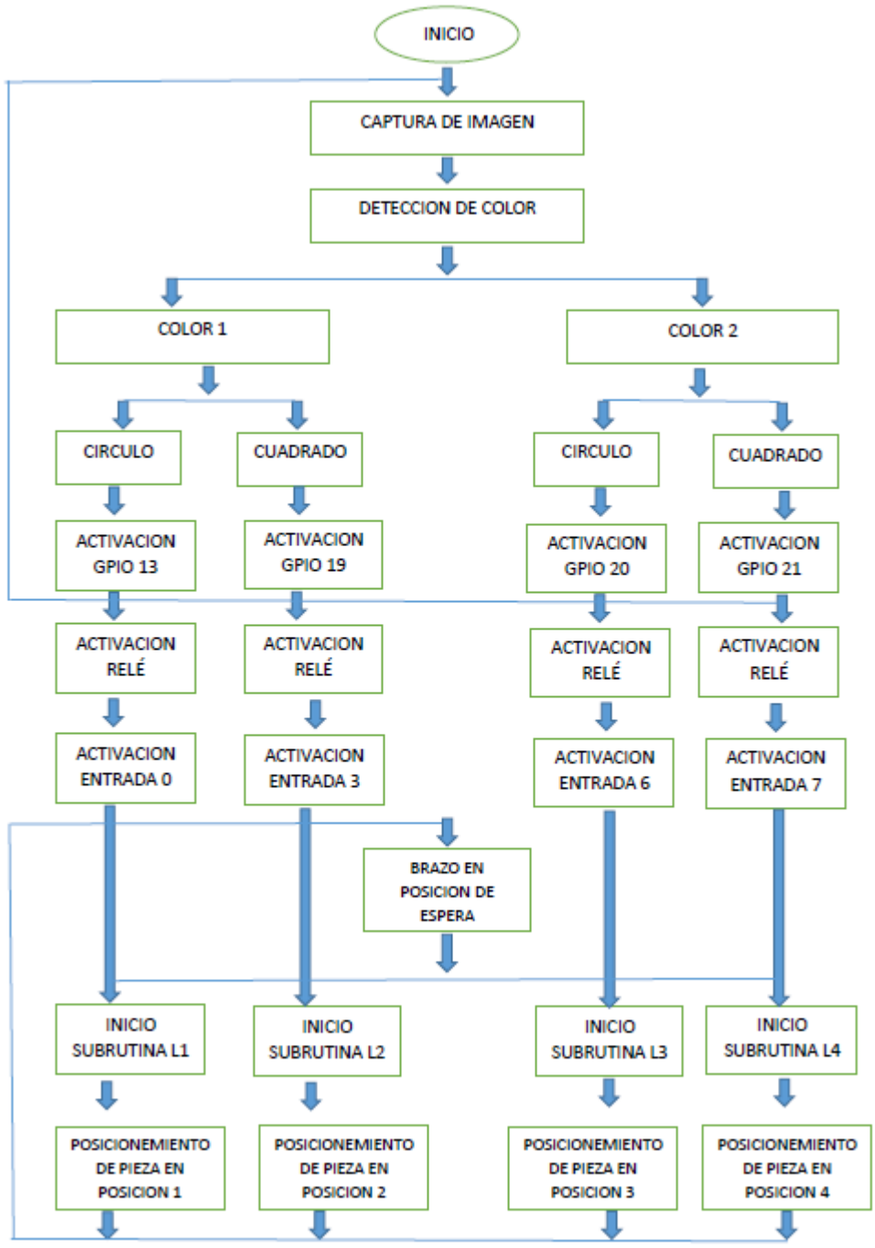

Figura 2-55: Diagrama de flujo integracion Raspberry Brazo Robotico

Fuente: Imagen de elaboración propia

CONCLUSIONES Y RECOMENDACIONES

- La aplicación de Visión Artificial es bastante amplia, y últimamente es muy utilizada como una tecnología irremplazable para aplicaciones específicas.
- Este proyecto es de bajo costo, lo elementos base del hardware de visión artificial son económicos, 100 dólares máximo aproximado, esto contempla Raspberry, cámara, fuente y módulo de relés.
- Ya que las condiciones de luz son variables, de acuerdo al lugar, se creó una máscara de color con un rango que permite que si las condiciones varían dentro de este todavía funcione el algoritmo, estos valores son modificables en tiempo real desde la pantalla.
- Para implementar este proyecto de visión artificial, no es necesario una cámara de alto costo, se utilizó la más económica para el puerto de la Raspberry.
- Este proyecto de visión artificial está diseñado para trabajar con objetos que poseen características conocidas, con una altura, tamaño, color y forma predeterminada. El cambio de tamaño de los objetos y la distancia se deben configurar de acuerdo a las condiciones del lugar a implementar, sin embargo estos pueden ser modificados en tiempo real desde la pantalla.
- Este sistema es muy versátil y se puede seguir desarrollando y perfeccionando para detectar mayor número de colores y formas. Así también desarrollar una interfaz para su utilización.

BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN

1. FESTO.COM. [en línea]. <https://www.festo-didactic.com/es-es/productos/software-e-learning/ciros/ciros-studio-creacion-de-entornos-virtuales-de-aprendizaje.htm?fbid=ZXMuZXMuNTQ3LjE0LjE4LjExMTAuODE4Ng> [Consulta: 03 Abril 2019].
2. GEEKYTHEORY.COM. Cursos de programación online [en línea]. < https://geekytheory.com/tutorial-raspberry-pi-uso-de-picamera-con-python> [Consulta: 28 Octubre 2018].
3. MELTRADE.HU. [en línea]. < http://meltrade.hu/open_pdf.php?f=9641-rv-2sd-instruction-manual- > [Consulta: 07 Marzo 2019].
4. MITSUBISHIELECTRIC.COM. [en línea]. <https://www.mitsubishielectric.com/fa/assist/e-learning/pdf/spa/5-MELFA_Basics_FD_fod_spa.pdf> [Consulta: 29 Noviembre 2018].
5. PROGRAMARFACIL.COM. [en línea]. <https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/> [Consulta: 03 Noviembre 2018].
6. PROGRAMOERGOSUM.COM. [en línea]. <https://www.programoergosum.com/cursos-online/raspberry-pi/238-control-de-gpio-con-python-en-raspberry-pi/que-es-gpio> [Consulta: 30 Junio 2019].
7. ROBOLOGS.NET. [en línea]. <https://robologs.net/2015/07/26/como-filtrar-el-ruido-de-una-mascara-con-opencv/> [Consulta: 07 Noviembre 2018].
8. SCRIBD.COM. [en línea]. <https://es.scribd.com/document/151109625/Ciros-Robotics-Manual-Es> [Consulta: 21 Marzo 2019].
9. UDB.EDU.SV. Universidad don Bosco. Puesta en funcionamiento del robot Mitsubishi [en línea]. http://www.udb.edu.sv/udb/archivo/guia/electronica-ingenieria/fundamentos-de-robotica/2017/i/guia-8.pdf > [Consulta: 19 Marzo 2019].
10. UNIPYTHON.COM. [en línea]. <https://unipython.com/transformaciones-morfologicas/> [Consulta: 06 Noviembre 2018].