



Escuela Técnica Superior de
INGENIERÍA DE SEVILLA



Universidad de Sevilla
Escuela Técnica Superior de Ingeniería
Laboratorio de Automatización y Robótica

Proyecto final de asignatura

Alumnos:

Araceli Baena Baena
Antonio Bustos López
Alfonso Boceta Lasarte
Juan Ariza Ramos

Índice

1. Esquema general del proyecto	3
2. Modo Cinta	4
2.1. Explicación y justificación	4
2.1.1. Transformación HSV: Explicación	5
2.2. Funciones <i>resaltadoras</i>	5
2.3. Clasificación en función del color de la pieza	5
2.3.1. Función detector de colores	7
3. Modo Reconocimiento	8
3.1. Explicación general	8
3.2. Código Principal	9
3.3. Problemas encontrados	10
3.3.1. Posicionamiento de la cuadrícula	10
3.3.2. Cambios de luz	10
4. Funciones específicas del Scrbot	11
4.1. Control de la cinta mediante sensor de movimiento	11
4.2. Transporte de piezas al depósito	11
4.2.1. Nombres de las posiciones	11
4.3. Modo bandera	12
4.3.1. Nombres de las posiciones	12
4.3.2. Instrucción por bandera	13
4.4. Problemas encontrados	15
4.4.1. Desajuste en las posiciones	15
4.4.2. Retraso en la colocación de piezas	15
4.4.3. Problemas con la cinta	15
5. Anexo: Funciones completas	16
5.1. Funciones <i>resaltadoras</i> de colores	16
5.1.1. <i>resaltarColorAzul.m</i>	16
5.1.2. <i>resaltarColorBlanco.m</i>	16
5.1.3. <i>resaltarColorRojo.m</i>	17
5.2. Función detectador de colores	17
5.3. Función para Modo Reconocimiento	18
5.4. Funciones específicas del Scrbot	20
5.4.1. Control de la cinta mediante sensor de movimiento	20
5.4.2. Función transporte de piezas al depósito	21
5.4.3. Modo bandera	22
5.5. Código Main	26

1. Esquema general del proyecto

Introducción

Este proyecto se ha pensado tratando de implementar los conocimientos obtenidos durante el curso. Para ello hemos diseñado un sistema que se divide en dos ramas principales.

Una primera parte del proyecto consiste en la elaboración de banderas en una cuadrícula y clasificación de piezas en base de su color, haciendo uso de recursos del laboratorio como el brazo robótico **Scorbot**, la cinta, una cámara, o sensores de proximidad.

La segunda parte del proyecto se basa en el reconocimiento de banderas mediante webcam, haciendo uso de programas que resaltan colores para su identificación.

A continuación se presenta un resumen del funcionamiento del proyecto desarrollado:

Modo Cinta

1. **Inicio del proceso:** El robot inicia en el **Modo Cinta** con la **Cámara 1** posicionada en el extremo izquierdo de la cinta transportadora. Una pieza de color (azul, rojo o blanco) se coloca sobre la cinta en dicho extremo.
2. **Captura y detección de color:** Desde la consola de MATLAB, se toma una fotografía de la pieza utilizando la **Cámara 1**. La imagen capturada se muestra en pantalla y el programa analiza el color de la pieza (rojo, azul o blanco), imprimiendo el resultado.
3. **Movimiento de la cinta:** Una vez identificado el color, la cinta se activa, desplazando la pieza hacia el extremo derecho. El sensor de proximidad detecta cuando la pieza alcanza su posición final y ordena detener la cinta.
4. **Clasificación de la pieza:** El **Scorbot** recoge la pieza y la transporta hacia el depósito correspondiente según su color:
 - **Zona roja:** para piezas rojas.
 - **Zona azul:** para piezas azules.
 - **Zona blanca:** para piezas blancas.Además, el programa identifica si la pieza es la **primera, segunda o tercera** de su color y la posiciona en el lugar correspondiente dentro de su depósito.
5. **Restricciones:** No pueden almacenarse más de tres piezas del mismo color.
6. **Repetición del ciclo:** Este proceso se repite nueve veces, asegurando que las nueve piezas (tres de cada color) queden correctamente clasificadas en sus depósitos.

Modo Reconocimiento

1. **Selección de bandera:** Una vez completada la clasificación de las piezas, se puede seleccionar el **Modo Bandera** desde la consola de MATLAB. El usuario elige construir una de las siguientes banderas: **Chile, Polonia, Francia, Rusia o Holanda**.
2. **Construcción de la bandera:** El **Scorbot** extrae las piezas del depósito y las posiciona sobre una cuadrícula 3x3 para construir la bandera seleccionada. El robot sigue un patrón predefinido para cada bandera, colocando las piezas en las posiciones correctas de la cuadrícula.

Modo Reconocimiento

1. **Encendido de la Cámara 2:** El sistema activa la **Cámara 2**, situada sobre la cuadrícula 3x3.
2. **Reconocimiento manual de banderas:** El usuario puede construir manualmente cualquier bandera en la cuadrícula utilizando las piezas disponibles.
3. **Detección:** El programa captura la disposición de las piezas y detecta de qué bandera se trata. Las banderas reconocidas son:

- Chile
- Polonia
- Francia
- Rusia
- Holanda

4. **Validación:** Si la disposición no coincide con ninguna de las banderas definidas, el sistema indica que no se reconoce como una bandera válida.

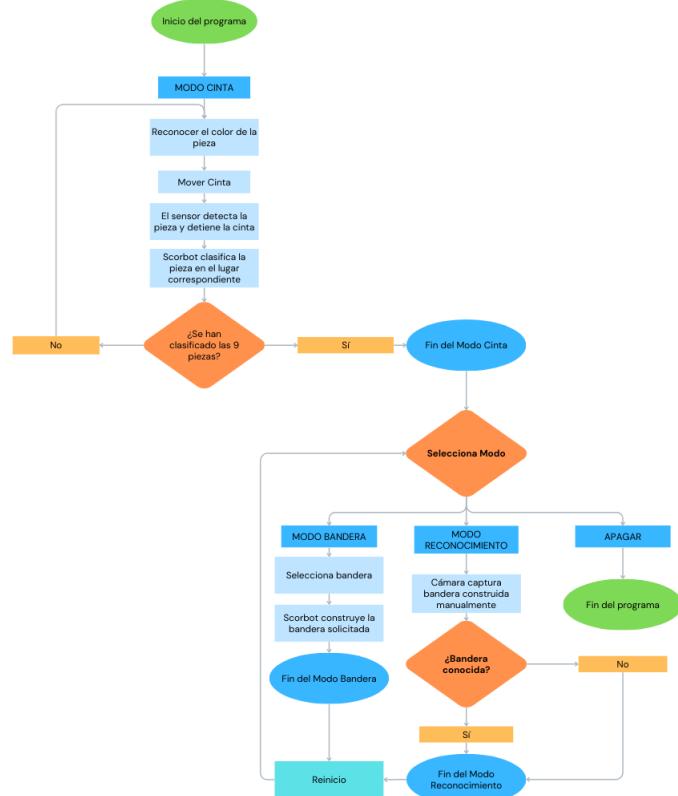


Figura 1: Diagrama de flujo del programa

2. Modo Cinta

Tal y como se ha introducido previamente, en este apartado se trabajará con la **cámara 1**, para esto, se empleará *Image Acquisition Toolbox* de MATLAB, el cual permite conectar la cámara USB con MATLAB y trabajar con ella desde los scripts. Esta cámara estará posicionada al inicio de la cinta, donde llegará la pieza para que se le tome una foto, la cuál será analizada para determinar el color de la pieza.

2.1. Explicación y justificación

Para el correcto análisis de las imágenes, es necesario hacer un preprocesamiento de las imágenes, incluyendo, entre otras cosas, la transformación de la imagen a un formato más sencillo que el RGB. Por este motivo, y tras probar con varios métodos, como comparar valores máximos de cada capa de la matriz RGB o analizar píxeles individuales, se ha determinado que el mejor método con el que podíamos trabajar era transformando la imagen a HSV.

2.1.1. Transformación HSV: Explicación

La transformación HSV presenta como principal ventaja, el que se aíslle el brillo (*value*) de las otras componentes que definen el color: matiz (*hue*) y saturación (*saturation*), facilitando el estudio del color de la imagen.

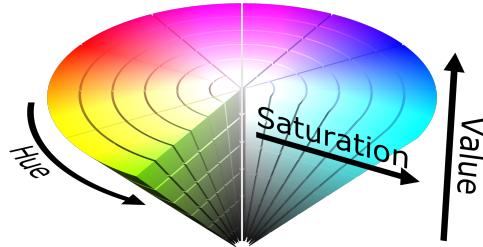


Figura 2: Cono HSV.

Se muestra aquí una comparación entre una imagen RGB y la misma imagen tras ser transformada a HSV:

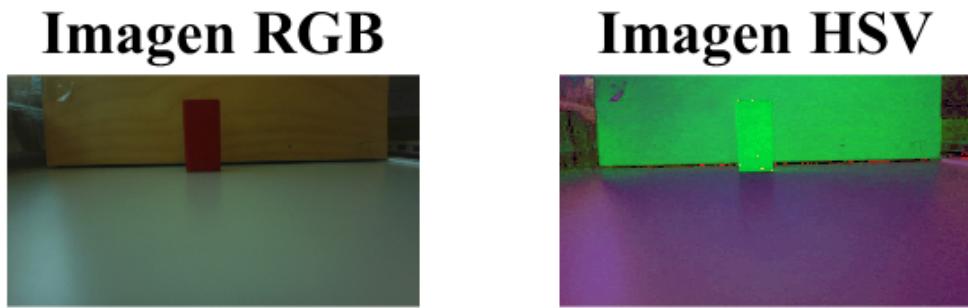


Figura 3: Comparación RGB-HSV

2.2. Funciones *resaltadoras*

Se les ha llamado funciones resaltadoras a aquellas funciones encargadas de resaltar uno de los tres colores relevantes (rojo, azul y blanco) de las imágenes, tras ser estas procesadas. Estas funciones son:

- *resaltarColorRojo.m*
- *resaltarColorAzul.m*
- *resaltarColorBlanco.m*

El funcionamiento de estas imágenes es el siguiente:

Estas funciones son iguales para los tres colores, cambiando únicamente los umbrales que definen el color. Todas devuelven como argumento de salida la máscara creada.

2.3. Clasificación en función del color de la pieza

Una vez definidas las funciones resaltadoras, se crea una función que, empleando estas funciones, determine el color de la imagen. Para ello emplea contadores de píxeles y compara los tres valores para encontrar el mayoritario. Un ejemplo del funcionamiento con una imagen sencilla es el siguiente:

Algorithm 1 Resaltar colores

Transformar imágenes a hsv.

Separar en los distintos canales $HSV \leftarrow H, S, V$

Definir rangos de saturación.

Definir saturación y brillo mínimo (*preferiblemente altos*).

Crear la máscara en función de los umbrales ya definidos:

$mask = rangoColor \ \& \ saturación \ \& \ brilloAlto$

Poner todos los valores no acordes con la máscara definida a 0:

$imagenResaltada(repmat(mask, [1, 1, 3])) = 0;$

$=0$

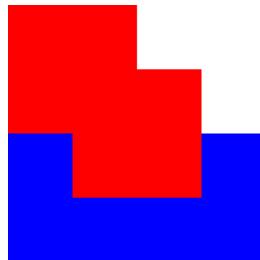


Figura 4: Imagen de prueba.

Imagen resaltada Azul

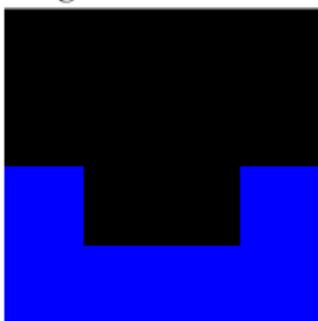


Imagen resaltada Roja

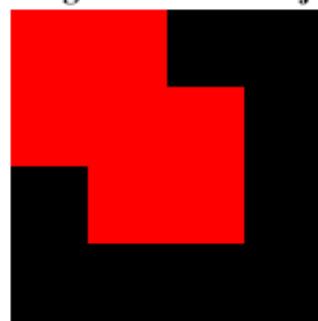


Imagen resaltada Blanca



Contador color: 15000

Contador color: 17500

Contador color: 7500

Figura 5: Resultados de detección de colores.

Para la imagen se obtiene que el color de la pieza es el rojo al tener un contador mayoritario (17500 píxeles rojos frente a 15000 píxeles azules y 7500 píxeles blancos).

Para una foto más compleja se obtienen los siguientes resultados:



Figura 6: Imagen de prueba compleja.

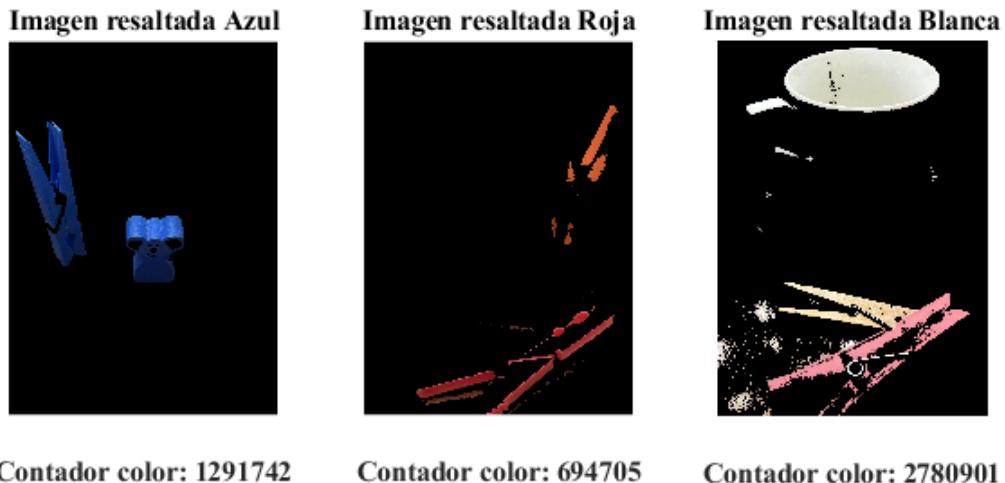


Figura 7: Resultados de imagen de prueba compleja.

Para esta imagen se obtiene que el color mayoritario es blanco, sin embargo, se observan los efectos de destellos de la cámara debido a la iluminación empleada, así como fallos en la detección del color blanco. Esto es un problema recurrente, del cual se hablará en mayor detalle más adelante.

2.3.1. Función detector de colores

La función encargada de la detección de los colores, tiene la siguiente lógica:

Nota: El recorte de la imagen se ha obviado en las imágenes de prueba, con el fin de mostrar los errores debido a brillos y reflejos en la imagen. En el proyecto se había recortado de forma que se estudiase un trozo de la pieza en lugar de la totalidad, esto se hizo en pro de la robustez del programa, ya que los cambios en la iluminación hacían que ciertos colores del fondo alterasen de forma significativa el resultado en función de la hora del día.

Esta función devuelve en minúsculas el color de la pieza:

- "azul"
- "rojo"
- "blanco"

Estos valores se emplearán en funciones futuras para la toma de decisión de la posición objetivo del robot.

Algorithm 2 Detectar colores

- 1: **Recortar imagen.**
 - 2: Obtener máscaras con las funciones resaltadoras de color.
 - 3: Crear tres versiones de la imagen, cada una resaltando uno de los tres colores de interés.
 - 4: Definir saturación y brillo mínimo (*preferiblemente altos*).
 - 5: Convertir imágenes resaltadas en vectores.
 - 6: **for** cada píxel de la imagen resaltada **do**
 - 7: **if** el píxel $\neq 0$ **then**
 - 8: $contadorColor = contadorColor + 1$
 - 9: **end if**
 - 10: **end for**
 - 11: Comparar valores de los contadores.
 - 12: Mostrar imágenes resaltadas.
 =0
-

3. Modo Reconocimiento

3.1. Explicación general

El tercer y último modo de nuestro proyecto consiste en un algoritmo de reconocimiento de banderas. Para ello, se tomará como marco de referencia una cuadrícula 3x3 sobre la que se situará la cámara. Sobre cada una de las casillas de la cuadrícula se colocarán las piezas previamente utilizadas, tratando de representar alguna de las banderas posibles.

El funcionamiento del Modo Reconocimiento es muy similar al algoritmo de reconocimiento de piezas del Modo Cinta (es decir, basado en las funciones encargadas de resaltar los colores buscados en la foto), pero con algunas diferencias: Hasta etapas más avanzadas del proyecto, la cuadrícula donde se construían las banderas era blanca, por lo que el algoritmo solo se encargaba de detectar si la pieza que había en cada casilla era Azul o Roja, en caso de no ser ninguna de las dos, sería considerada blanca (el color blanco viene determinado por un umbral)

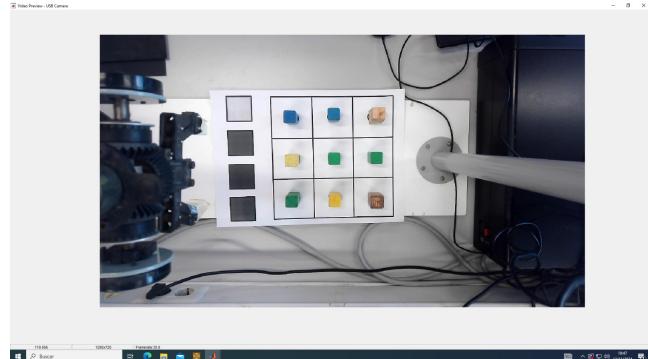


Figura 8: Cuadrícula Inicial

Finalmente, para facilitar el reconocimiento de piezas y evitar problemas relacionados con los valores del umbral blanco, se optó por realizar las banderas sobre una cuadrícula negra (hecha a mano). A partir de entonces el proyecto se realizó sobre esa nueva cuadrícula.

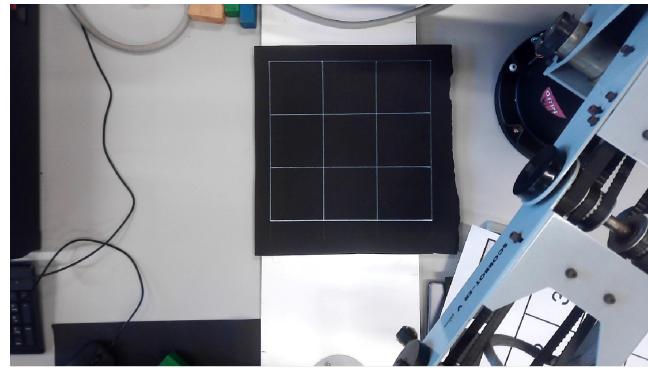


Figura 9: Cuadrícula Final

3.2. Código Principal

El código del Modo Reconocimiento está diseñado para seguir los siguientes pasos:

- Captura de una imagen a través de la cámara.
- Recorte de la región de interés y conversión al espacio de color HSV.
- Dividir la imagen en una cuadrícula para facilitar la localización de colores en cada región.
- Mostrar la imagen recortada y dividida en una cuadrícula 3x3
- Análisis de cada celda para determinar el color predominante, almacenado en una matriz 3x3.
- Comparar los colores detectados con patrones predefinidos de banderas nacionales.
- Indicar si la imagen corresponde a una bandera conocida.

Se adjunta a continuación el pseudocódigo del script de MATLAB que realiza las funciones previamente descritas:

Algorithm 3 Reconocimiento de bandera

```

1: Iniciar cámara web
2: Capturar y guardar imagen: img ← snapshot(cámaras)
3: Recortar la región de interés: img ← imcrop(img, región)
4: Calcular dimensiones de la cuadrícula:
5:     Dividir imagen en celdas  $3 \times 3$ 
6: Iniciar matriz de colores: regionColors ← []
7: for cada celda en la cuadrícula  $3 \times 3$  do
8:     Extraer celda
9:     Detectar color predominante
10:    Guardar color en regionColors
11: end for
12: Comparar regionColors con patrones de banderas conocidas
13: if coincide con algún patrón then
14:     Mostrar el nombre de la bandera detectada
15: else
16:     Mostrar que no se reconoce ninguna bandera
17: end if=0

```

3.3. Problemas encontrados

3.3.1. Posicionamiento de la cuadrícula

La correcta alineación de la cuadrícula para la captura de imágenes resultó problemática debido a la altura de la cámara. Pequeños desajustes en la posición del tablero causaban errores en la delimitación de las casillas.

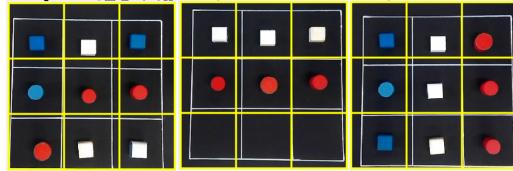


Figura 10: Desajuste en el recorte

Para solucionarlo, se programó un script de calibración que dibujaba un marco con las dimensiones exactas de la cuadrícula sobre la vista previa de la cámara, permitiendo ajustar la posición antes de cada experimento.

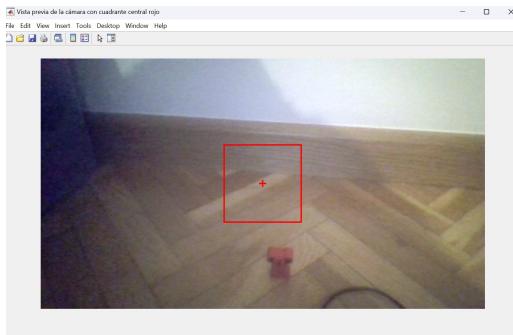


Figura 11: Calibración de la cámara

3.3.2. Cambios de luz

Las variaciones en la iluminación, tanto espaciales (cambio de localización para realizar los experimentos) como temporales (las sesiones de laboratorio duraban de 9:00 a 13:00) afectaban la detección de colores, en especial el azul. Esto se debía a que el algoritmo dependía de rangos de valores estáticos para la detección de colores. Para garantizar condiciones óptimas, se utilizó un foco de luz ajustable el día del proyecto, asegurando consistencia en las imágenes capturadas y mejorando la precisión del sistema.



Figura 12: Foco utilizado

4. Funciones específicas del Scorbot

En este apartado se detallan las funciones operativas del Scorbot, encargado de realizar tareas clave dentro del proyecto. Con la ayuda de un *sensor de movimiento* y la *cinta transportadora*, el Scorbot es capaz de llevar a cabo las tareas de manera fluida y sincronizada.

El robot realiza tres funciones principales:

- **Control de la cinta mediante sensor de movimiento:** El *Scorbot* utiliza el sensor de movimiento junto con la cinta transportadora para trasladar las piezas hacia la zona de recogida
- **Transporte de piezas al depósito:** Una vez que la pieza ha sido trasladada a la zona de recogida, el robot la transporta al depósito.
- **Modo bandera:** Cuando se requiere la construcción de una bandera, el *Scorbot* organiza las piezas en un tablero, siguiendo una instrucción predefinida.

4.1. Control de la cinta mediante sensor de movimiento

La cinta va a transportar 9 piezas desde una cámara que se usa para el reconocimiento del tipo de pieza, hasta el *Scorbot* que va a llevarlas hasta un depósito. La lógica de la función cinta es de la siguiente manera:

1. **Captura y detección de color:** Presiona la tecla ‘R’. En este momento se hace una foto y con la función de detección de colores se clasifica la pieza.
2. **Movimiento de la cinta:** Se inicia a mover la cinta.
3. **Lectura de entradas:** Las entradas de esta función son las que nos da el sensor de proximidad, mientras sean cero, la cinta debe moverse.
4. **Clasificación de la pieza:** Se para la cinta al llegar la pieza al sensor, entonces se ejecuta la función *LlevarPiezasDep.*
5. **Vuelta al inicio:** Una vez el robot lleva la pieza al depósito, se espera hasta que el usuario ponga otra pieza en la cinta, si se hace otra foto, se vuelve a ejecutar el código, si no, se le indica por pantalla que se termine de ejecutar.



Figura 13: Pieza detenida a la altura del sensor.

4.2. Transporte de piezas al depósito

Una vez que la pieza se detiene en la zona de recogida de la cinta, el robot se encarga de trasladarla a su posición correspondiente en el depósito.

4.2.1. Nombres de las posiciones

Para facilitar el proceso, se nombraron las posiciones del depósito según el color y la cantidad de piezas de ese mismo color.

pa1	pr1	pb1
pa2	pr2	pb2
pa3	pr3	pb3

Figura 14: Nombres de las posiciones del depósito

Donde:

- pa1: Posición azul 1
- pa2: Posición azul 2
- pa3: Posición azul 3
- pr1: Posición rojo 1
- pr2: Posición rojo 2
- pr3: Posición rojo 3
- pb1: Posición blanco 1
- pb2: Posición blanco 2
- pb3: Posición blanco 3

Es importante destacar que se definieron posiciones de aproximación para cada posición. Se nombraron añadiendo una “a” al final. También se creó una posición de aproximación al depósito llamada “apdep”.

Para evitar errores al determinar cuántas piezas de un mismo color ya se encuentran en el depósito, se utilizan contadores globales. Estos contadores, definidos para cada color (azul, rojo y blanco), llevan un registro del número de piezas colocadas en el depósito.

4.3. Modo bandera

El modo bandera comienza con la selección, por parte del usuario, de la bandera que desea representar. Una vez que el usuario ha elegido la bandera, el robot procede a construirla, transportando las piezas una a una desde el depósito hasta el tablero.

4.3.1. Nombres de las posiciones

Las posiciones del tablero se nombraron de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 15: Nombres de las posiciones del tablero

Donde:

- p1: Posición tablero 1
- p2: Posición tablero 2
- p3: Posición tablero 3
- p4: Posición tablero 4
- p5: Posición tablero 5
- p6: Posición tablero 6
- p7: Posición tablero 7
- p8: Posición tablero 8
- p9: Posición tablero 9

Es importante destacar que se definieron posiciones de aproximación para cada posición. Se nombraron añadiendo una “a” al final. También se creó una posición de aproximación al tablero llamada “aptab”.

4.3.2. Instrucción por bandera

En función de la bandera seleccionada, el robot ejecuta una instrucción específica.

Francia

Para construir la bandera de Francia, las piezas se desplazan siguiendo esta secuencia:

$pa1 \rightarrow p1, pa2 \rightarrow p4, pa3 \rightarrow p7, pr1 \rightarrow p3, pr2 \rightarrow p6, pr3 \rightarrow p9, pb1 \rightarrow p2, pb2 \rightarrow p5, pb3 \rightarrow p8.$

El tablero queda de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 16: Bandera de Francia sobre el tablero

Chile

Para construir la bandera de Chile, las piezas se desplazan siguiendo esta secuencia:

$pa1 \rightarrow p1, pr1 \rightarrow p4, pr2 \rightarrow p5, pr3 \rightarrow p6, pb1 \rightarrow p2, pb2 \rightarrow p3.$

El tablero queda de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 17: Bandera de Chile sobre el tablero

Rusia

Para construir la bandera de Rusia, las piezas se desplazan siguiendo esta secuencia:

$pa1 \rightarrow p4, pa2 \rightarrow p5, pa3 \rightarrow p6, pr1 \rightarrow p7, pr2 \rightarrow p8, pr3 \rightarrow p9, pb1 \rightarrow p1, pb2 \rightarrow p2, pb3 \rightarrow p3.$

El tablero queda de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 18: Bandera de Rusia sobre el tablero

Holanda

Para construir la bandera de Holanda, las piezas se desplazan siguiendo esta secuencia:

$pa1 \rightarrow p7, pa2 \rightarrow p8, pa3 \rightarrow p9, pr1 \rightarrow p1, pr2 \rightarrow p2, pr3 \rightarrow p3, pb1 \rightarrow p4, pb2 \rightarrow p5, pb3 \rightarrow p6.$

El tablero queda de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 19: Bandera de Holanda sobre el tablero

Polonia

Para construir la bandera de Polonia, las piezas se desplazan siguiendo esta secuencia:

$pr1 \rightarrow p4, pr2 \rightarrow p5, pr3 \rightarrow p6, pb1 \rightarrow p1, pb2 \rightarrow p2, pb3 \rightarrow p3.$

El tablero queda de la siguiente manera:

p1	p2	p3
p4	p5	p6
p7	p8	p9

Figura 20: Bandera de Polonia sobre el tablero

4.4. Problemas encontrados

4.4.1. Desajuste en las posiciones

Con el paso de las semanas, las posiciones se desajustaron, especialmente las del depósito de piezas. Para solucionarlo, se añadió una plataforma de unos cinco centímetros de espesor en la zona de depósito.

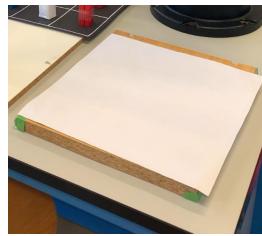


Figura 21: Plataforma añadida

4.4.2. Retraso en la colocación de piezas

En algunas secuencias de construcción de banderas, el robot soltaba la pieza antes de que estuviera correctamente colocada sobre el tablero. Esto se debía a que algunas posiciones del tablero estaban definidas muy cerca del límite del espacio de trabajo del robot, lo que provocaba que tardara más en llegar a ellas. Para solucionar este problema, se implementaron pausas más prolongadas antes de abrir la garra.

4.4.3. Problemas con la cinta

El movimiento de la cinta ha sido un dolor de cabeza porque la velocidad de la misma dependía de las condiciones del día. En función del día la cinta se movía a una velocidad u otra, a veces incluso no se movía. Para solucionar este problema hemos ido probando comprobando cada día que valor poner en: `fcsetanout(ps, 7, -1300)`. Siendo el -1300 el valor a modificar.

5. Anexo: Funciones completas

5.1. Funciones *resaltadoras* de colores

5.1.1. *resaltarColorAzul.m*

```
1 function [imagenes, nombresImagenes] = cargaImagenes()
2 function mask = resaltarColorAzul(imagen)
3 % Convertir la imagen al espacio de color HSV
4 imagenHSV = rgb2hsv(imagen);
5
6 % Separar los canales HSV
7 H = imagenHSV(:, :, 1); % Tono (Hue)
8 S = imagenHSV(:, :, 2); % Saturación
9 V = imagenHSV(:, :, 3); % Valor (brillo)
10
11 % Definir los rangos para detectar el color azul
12 % El azul típico tiene un tono entre 0.5 y 0.7 en el espacio HSV
13 rangoAzul = (H >= 0.5 & H <= 0.7);
14
15 % La pieza debe tener saturación alta y buen brillo
16 saturacionAlta = S > 0.5;
17 brilloAlto = V > 0.1;
18
19 % M scara para el color azul
20 mask = rangoAzul & saturacionAlta & brilloAlto;
21
22 % Opcional: Resaltar la pieza en la imagen original
23 imagenResaltada = imagen; % Copia la imagen
24 imagenResaltada(repmat(~mask, [1, 1, 3])) = 0; % Todo lo que no es azul se pone negro
25
26 % % Mostrar resultados
27 % figure;
28 % subplot(1, 2, 1); imshow(imagen); title('Imagen Original');
29 % subplot(1, 2, 2); imshow(imagenResaltada); title('Pieza Azul Resaltada');
30 end
```

Listing 1: Función resaltarColorAzul en MATLAB

5.1.2. *resaltarColorBlanco.m*

```
1 function mask = resaltarColorBlanco(imagen)
2 % Convertir la imagen al espacio de color HSV
3 imagenHSV = rgb2hsv(imagen);
4
5 % Separar los canales HSV
6 H = imagenHSV(:, :, 1); % Tono (Hue)
7 S = imagenHSV(:, :, 2); % Saturación
8 V = imagenHSV(:, :, 3); % Valor (brillo)
9
10 % Definir los rangos para detectar el color blanco
11 % El blanco tiene saturación muy baja y brillo alto
12 saturacionBaja = S < 0.4; % Baja saturación indica colores desaturados (cercaos a blanco
13 , gris o negro)
13 brilloAlto = V > 0.8; % Brillo alto indica que es cercano a blanco
14
15 % M scara para el color blanco
16 mask = saturacionBaja & brilloAlto;
17
18 % Opcional: Resaltar la pieza en la imagen original
19 imagenResaltada = imagen; % Copia la imagen
20 imagenResaltada(repmat(~mask, [1, 1, 3])) = 0; % Todo lo que no es blanco se pone negro
21
22 % % Mostrar resultados (opcional)
23 % figure;
24 % subplot(1, 2, 1); imshow(imagen); title('Imagen Original');
25 % subplot(1, 2, 2); imshow(imagenResaltada); title('Pieza Blanca Resaltada');
26 end
```

Listing 2: Función resaltarColorBlanco en MATLAB

5.1.3. resaltarColorRojo.m

```

1 function mask = resaltarColorRojo(imagen)
2 % Convertir la imagen al espacio de color HSV
3 imagenHSV = rgb2HSV(imagen);
4
5 % Separar los canales HSV
6 H = imagenHSV(:, :, 1); % Tono (Hue)
7 S = imagenHSV(:, :, 2); % Saturación
8 V = imagenHSV(:, :, 3); % Valor (brillo)
9
10 % Definir los rangos para detectar el color rojo
11 % Nota: El rojo tiene valores de tono cerca de 0 y alrededor de 1
12 rangoRojoBajo = (H >= 0 & H <= 0.05); % Parte baja del rojo
13 rangoRojoAlto = (H >= 0.95 & H <= 1); % Parte alta del rojo
14
15 % La pieza debe tener saturación alta y buen brillo
16 saturacionAlta = S > 0.5;
17 brilloAlto = V > 0.3;
18
19 % Mscara para el color rojo
20 mask = (rangoRojoBajo | rangoRojoAlto) & saturacionAlta & brilloAlto;
21
22 % Opcional: Resaltar la pieza en la imagen original
23 imagenResaltada = imagen; % Copia la imagen
24 imagenResaltada(repmat(~mask, [1, 1, 3])) = 0; % Todo lo que no es rojo se pone negro
25
26 % Mostrar resultados
27 % figure;
28 % subplot(1, 2, 1); imshow(imagen); title('Imagen Original');
29 % subplot(1, 2, 2); imshow(imagenResaltada); title('Pieza Roja Resaltada');
30 end

```

Listing 3: Función resaltarColorRojo en MATLAB

5.2. Función detectador de colores

```

1
2 function colorPieza = detectaColorPieza(imagenR, umbralBlanco)
3 % rec = [560,320,280,240];
4 % imagen=imcrop(imagenR,rec);
5 imagen = imagenR;
6 if nargin == 1
7     umbralBlanco = 900 ; %px
8 end
9 % Aplicar mscara =====
10 mascaraRoja = resaltarColorRojo(imagen);
11 imagenResaltadaRoja = imagen;
12 imagenResaltadaRoja(repmat(~mascaraRoja, [1, 1, 3])) = 0;
13
14 mascaraAzul = resaltarColorAzul(imagen);
15 imagenResaltadaAzul = imagen;
16 imagenResaltadaAzul(repmat(~mascaraAzul, [1, 1, 3])) = 0;
17
18 mascaraBlanca = resaltarColorBlanco(imagen);
19 imagenResaltadaBlanca = imagen;
20 imagenResaltadaBlanca(repmat(~mascaraBlanca, [1, 1, 3])) = 0;
21
22 % Escoger el color mayoritario =====
23 %Contadores
24 Azul = 0;
25 Roja = 0;
26 Blanco = 0;
27
28 % definir un umbral bajo el cual se considera blanco
29
30
31 % Convertir a vectores
32 ResaltadaAzul = reshape(mascaraAzul, [],1);
33 ResaltadaRoja = reshape(mascaraRoja, [],1);
34 ResaltadaBlanca = reshape(mascaraBlanca, [],1);

```

```

35 % Contar
36 for ii = 1 : length(ResaltadaRoja)
37     if floor(ResaltadaRoja(ii)) ~= 0; Roja = Roja +1; end
38 end
39 for ii = 1 : length(ResaltadaAzul)
40     if floor(ResaltadaAzul(ii)) ~= 0; Azul = Azul +1; end
41 end
42 for ii = 1 : length(ResaltadaBlanca)
43     if floor(ResaltadaBlanca(ii)) ~= 0; Blanco = Blanco +1; end
44 end
45
46 % Clasificar color de la pieza
47 if Blanco > Roja && Blanco > Azul
48     colorPieza = "blanco";
49 else
50     if Roja > Azul && Roja > Blanco
51         % imshow(imagenResaltadaRoja);
52         colorPieza = "rojo";
53     else
54         % imshow(imagenResaltadaAzul);
55         colorPieza = "azul";
56     end
57 end
58
59 figure
60 hold on
61 subplot(1,3,1);
62 imshow(imagenResaltadaAzul)
63 title("Imagen resaltada Azul")
64 xlabel("Contador color: "+ num2str(Azul))
65
66 subplot(1,3,2);
67 imshow(imagenResaltadaRoja)
68 title("Imagen resaltada Roja")
69 xlabel("Contador color: "+ num2str(Roja))
70
71 subplot(1,3,3)
72 imshow(imagenResaltadaBlanca)
73 title("Imagen resaltada Blanca")
74 xlabel("Contador color: "+ num2str(Blanco))
75 hold off
76 end

```

Listing 4: Función detectaColorPieza en MATLAB

5.3. Función para Modo Reconocimiento

```

1 function completa_NuevaR()
2 %clc, clear
3 % %%TOMAR LA FOTO DE LA BANDERA
4 % % Inicializar la webcam
5 % cam = webcam; % Si tienes varias c maras , puedes especificar 'webcam( ndice )'
6 %
7 % % Capturar una imagen
8 % disp('Capturando imagen... ');
9 % img = snapshot(cam);
10 %
11 % % Mostrar la imagen capturada
12 % figure;
13 % imshow(img);
14 % title('Imagen Capturada');
15 %
16 % % Guardar la imagen en un archivo (opcional)
17 % imwrite(img, 'imagen_capturada.jpg');
18 % disp('La imagen ha sido guardada como "imagen_capturada.jpg". ');
19 %
20 % % Liberar la webcam
21 % clear cam;
22 % Cargar la imagen
23 cam=webcam(1);
24 img=snapshot(cam);
25 imshow(img)

```

```

26 % Convertir la imagen a HSV para facilitar la detección de colores
27 img_hsv = rgb2hsv(img);
28 %% recorte- prueba
29 % img = snapshot(webcam("USB Camera"));
30 % Convertir la imagen a HSV para facilitar la detección de colores
31 img_hsv = img;
32 %% recorte- prueba
33
34 % tamRecorte = [330, 325];
35 recorte = [515, 158, 340, 340];
36 img = imcrop(img, recorte);
37 img_hsv = (img);
38
39 %% toma dimensiones
40 % Obtener las dimensiones de la imagen
41 [height, width, ~] = size(img);
42
43 %% divisiones e inicialización
44 % Calcular el tamaño de cada celda en la cuadrícula 3x3
45 %
46 % Parte de la suposición que la imagen ha sido perfectamente recortada,
47 % esto debe ser hecho previamente
48 % tamRecorte = [330, 325];
49 % recorte = centerCropWindow2d(size(img_hsv), tamRecorte);
50 % img = imcrop(img, recorte);
51 % img_hsv = rgb2hsv(img);
52 %
53 cellHeight = floor(height / 3);
54 cellWidth = floor(width / 3);
55
56 % Inicializar una matriz para almacenar el color dominante de cada celda
57 regionColors = strings(3, 3);
58
59 % Mostrar la división en celdas de la imagen
60 figure;
61 imshow(img);
62 hold on;
63 for row = 1:3
64     for col = 1:3
65         % Calcular los límites de cada celda
66         yStart = (row - 1) * cellHeight + 1;
67         yEnd = row * cellHeight;
68         xStart = (col - 1) * cellWidth + 1;
69         xEnd = col * cellWidth;
70
71         % Dibujar un rectángulo sobre cada celda
72         rectangle('Position', [xStart, yStart, cellWidth, cellHeight], 'EdgeColor', 'yellow',
73                     'LineWidth', 2);
74     end
75 end
76 title('Imagen con Cuadrícula 3x3');
77
78 % Recorrer cada celda de la cuadrícula 3x3 para detectar los colores
79 for row = 1:3
80     for col = 1:3
81         % Calcular los límites de cada celda
82         yStart = (row - 1) * cellHeight + 1;
83         yEnd = row * cellHeight;
84         xStart = (col - 1) * cellWidth + 1;
85         xEnd = col * cellWidth;
86
87         % Extraer la celda de la imagen
88         cellImage = img_hsv(yStart:yEnd, xStart:xEnd, :);
89
90         umbralBlanco = 500; %px
91
92         regionColors(row, col) = detectaColorPieza1(cellImage, umbralBlanco);
93     end
94 end
95
96 % Suponemos que 'regionColors' es una matriz 3x3 de colores detectados.
97 % Por ejemplo:

```

```

98 % regionColors = ["blanco", "blanco", "blanco";
99 %                 "rojo", "rojo", "rojo";
100 %                "azul", "azul", "azul"];
101
102 % Mostrar los colores detectados
103 disp('Colores detectados en la cuadr cula 3x3:');
104 disp(regionColors);
105
106 % Inicializaci n de bandera reconocida
107 banderaReconocida = false;
108
109 % Caso 1: Polonia
110 % Regiones 4, 5 y 6 deben ser "rojo"
111 if isequal(regionColors(1, :), ["blanco", "blanco", "blanco"]) && ...
112     isequal(regionColors(2, :), ["rojo", "rojo", "rojo"])
113     disp('La imagen corresponde a la bandera de Polonia.');
114     banderaReconocida = true;
115
116 % Caso 2: Francia
117 elseif isequal(regionColors(:, 1)', ["azul", "azul", "azul"]) && ...
118     isequal(regionColors(:, 2)', ["blanco", "blanco", "blanco"]) && ...
119     isequal(regionColors(:, 3)', ["rojo", "rojo", "rojo"])
120     disp('La imagen corresponde a la bandera de Francia.');
121     banderaReconocida = true;
122
123 % Caso 3: Chile
124 elseif isequal(regionColors(1, :), ["azul", "blanco", "blanco"]) && ...
125     isequal(regionColors(2, :), ["rojo", "rojo", "rojo"])
126     disp('La imagen corresponde a la bandera de Chile.');
127     banderaReconocida = true;
128
129 % Caso 5: Rusia
130 elseif isequal(regionColors(1, :), ["blanco", "blanco", "blanco"]) && ...
131     isequal(regionColors(2, :), ["azul", "azul", "azul"]) && ...
132     isequal(regionColors(3, :), ["rojo", "rojo", "rojo"])
133     disp('La imagen corresponde a la bandera de Rusia.');
134     banderaReconocida = true;
135
136 % Caso 6: Holanda
137 % Regiones 1, 2, 3 deben ser "azul" y 4, 5, 6 "amarillo"
138 elseif isequal(regionColors(2, :), ["blanco", "blanco", "blanco"]) && ...
139     isequal(regionColors(3, :), ["azul", "azul", "azul"]) && ...
140     isequal(regionColors(1, :), ["rojo", "rojo", "rojo"])
141     disp('La imagen corresponde a la bandera de Holanda.');
142     banderaReconocida = true;
143
144 % Ninguna bandera reconocida
145 else
146     disp('No se ha reconocido ninguna bandera conocida.');
147 end
148 end

```

5.4. Funciones específicas del Scortbot

5.4.1. Control de la cinta mediante sensor de movimiento

```

1 %Cinta
2 cam=webcam;
3 preview(cam)
4 img = snapshot(cam);
5 Foto=0;
6 consola = lower(input("Presione R para tomar una foto: "));
7 if consola == 'r'
8     img= snapshot(cam);
9     imshow(img); shg
10    colorPieza = detectaColorPieza(img);
11    disp(colorPieza);
12    Foto=1;
13 end
14
15 if Foto==1

```

```

16 entradas=fc_leer_entradas(ps);
17 pause(0.25)
18 fc_set_anout(ps, 7, -1300)
19 pause(0.25)
20 while entradas(2) == 0
21     entradas=fc_leer_entradas(ps);
22     pause(0.01);
23 end
24 if entradas(2)==1
25     disp('Detecta')
26     Foto=0;
27     fc_set_anout(ps, 7, 0)
28     pause(3)
29     disp('Ejecuto LlevarPiezas')
30     LlevarPiezasDep(ps, img);
31 end
32 end

```

Listing 5: Función Cinta en MATLAB

5.4.2. Función transporte de piezas al depósito

```

1 function LlevarPiezasDep(ps, img)
2 % Declarar los contadores como globales
3 global ContAzul ContRojo ContBlanco;
4
5 % Detectar el color de la pieza
6 colorPieza = detectaColorPieza(img, 300);
7
8 % Posiciones asignadas para cada color
9 PosAzula = {'pa1a', 'pa2a', 'pa3a'};
10 PosAzul = {'pa1', 'pa2', 'pa3'};
11
12 PosRojaA = {'pr1a', 'pr2a', 'pr3a'};
13 PosRojo = {'pr1', 'pr2', 'pr3'};
14
15 PosBlancaA = {'pb1a', 'pb2a', 'pb3a'};
16 PosBlanca = {'pb1', 'pb2', 'pb3'};
17
18 if strcmp(colorPieza, 'azul')
19     ContAzul = ContAzul + 1;
20     if ContAzul <= length(PosAzula)
21         acl_open(ps);
22         pause(2);
23         acl_move(ps, 'apcin');
24         pause(2);
25         acl_move(ps, 'cinta');
26         pause(2);
27         acl_close(ps);
28         pause(2);
29         acl_move(ps, 'apcin');
30         pause(2);
31         acl_move(ps, 'apdep');
32         pause(5);
33         acl_move(ps, PosAzula{ContAzul});
34         pause(3);
35         acl_move(ps, PosAzul{ContAzul});
36         pause(3);
37         acl_open(ps);
38         pause(3);
39         acl_move(ps, 'apdep');
40         pause(2);
41         acl_move(ps, 'apcin');
42     else
43         disp('Error: ContAzul excede el límite.');
44     end
45 end
46
47 if strcmp(colorPieza, 'rojo')
48     ContRojo = ContRojo + 1;
49     if ContRojo <= length(PosRojaA)
50         acl_open(ps);

```

```

51     pause(2);
52     acl_move(ps, 'apcin');
53     pause(2);
54     acl_move(ps, 'cinta');
55     pause(2);
56     acl_close(ps);
57     pause(2);
58     acl_move(ps, 'apcin');
59     pause(2);
60     acl_move(ps, 'apdep');
61     pause(5);
62     acl_move(ps, PosRojoA{ContRojo});
63     pause(3);
64     acl_move(ps, PosRojo{ContRojo});
65     pause(3);
66     acl_open(ps);
67     pause(3);
68     acl_move(ps, 'apdep');
69     pause(2);
70     acl_move(ps, 'apcin');

71 else
72     disp('Error: ContRojo excede el limite.');
73 end
74
75 if strcmp(colorPieza, 'blanco')
76     ContBlanco = ContBlanco + 1;
77     if ContBlanco <= length(PosBlancaA)
78         acl_open(ps);
79         pause(2);
80         acl_move(ps, 'apcin');
81         pause(2);
82         acl_move(ps, 'cinta');
83         pause(2);
84         acl_close(ps);
85         pause(2);
86         acl_move(ps, 'apcin');
87         pause(2);
88         acl_move(ps, 'apdep');
89         pause(5);
90         acl_move(ps, PosBlancaA{ContBlanco});
91         pause(3);
92         acl_move(ps, PosBlanca{ContBlanco});
93         pause(3);
94         acl_open(ps);
95         pause(3);
96         acl_move(ps, 'apdep');
97         pause(2);
98         acl_move(ps, 'apcin');

99 else
100    disp('Error: ContBlanco excede el limite.');
101 end
102
103 end
104 end

```

Listing 6: Función LlevarPiezasDep en MATLAB

5.4.3. Modo bandera

<pre> 1 % Función para el Modo Bandera 2 disp('Iniciando el Modo Bandera...'); 3 y=input('Introduzca la bandera deseada Francia 4 (1) Chile (2) Rusia (3) Polonia (4) 5 Holanda (5): '); 6 if y==1 7 acl_open(ps); 8 pause(2); 9 acl_move(ps, 'apdep'); 10 pause(2); 11 acl_move(ps, 'pa1'); 12 pause(2); 13 acl_move(ps, 'pa1'); 14 pause(2); 15 acl_close(ps); 16 pause(2); 17 acl_move(ps, 'apdep'); </pre>	<pre> 18 pause(2); 19 acl_move(ps, 'aptab'); 20 pause(2) 21 acl_move(ps, 'p1a'); 22 pause(2); 23 acl_move(ps, 'p1'); 24 pause(2); 25 acl_open(ps) 26 pause(2); 27 acl_move(ps, 'aptab') 28 pause(1); 29 30 acl_open(ps); 31 pause(2); 32 acl_move(ps, 'apdep'); 33 pause(2); 34 acl_move(ps, 'pa2a'); 35 pause(2); 36 acl_move(ps, 'pa2'); </pre>	<pre> 37 pause(2); 38 acl_close(ps); 39 pause(2); 40 acl_move(ps, 'apdep'); 41 pause(2); 42 acl_move(ps, 'aptab'); 43 pause(2); 44 acl_move(ps, 'p4a'); 45 pause(2); 46 acl_move(ps, 'p4'); 47 pause(2); 48 acl_open(ps) 49 pause(2); 50 acl_move(ps, 'aptab') 51 pause(1); 52 53 acl_open(ps); 54 pause(2); 55 acl_move(ps, 'apdep'); </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```

56     pause(2);
57     acl_move(ps, 'pa3a');
58     pause(2);
59     acl_move(ps, 'pa3');
60     pause(2);
61     acl_close(ps);
62     pause(2);
63     acl_move(ps, 'apdep');
64     pause(2);
65     acl_move(ps, 'aptab');
66     pause(2);
67     acl_move(ps, 'p7a');
68     pause(2);
69     acl_move(ps, 'p7');
70     pause(2);
71     acl_open(ps)
72     pause(2);
73     acl_move(ps, 'aptab')
74     pause(1);
75
76     acl_open(ps);
77     pause(2);
78     acl_move(ps, 'apdep');
79     pause(2);
80     acl_move(ps, 'pri1');
81     pause(2);
82     acl_move(ps, 'pri1');
83     pause(2);
84     acl_close(ps);
85     pause(2);
86     acl_move(ps, 'apdep');
87     pause(2);
88     acl_move(ps, 'aptab');
89     pause(2);
90     acl_move(ps, 'p3a');
91     pause(2);
92     acl_move(ps, 'p3');
93     pause(2);
94     acl_open(ps)
95     pause(2);
96     acl_move(ps, 'aptab')
97     pause(1);
98
99     acl_open(ps);
100    pause(2);
101    acl_move(ps, 'apdep');
102    pause(2);
103    acl_move(ps, 'pr2a');
104    pause(2);
105    acl_move(ps, 'pr2');
106    pause(2);
107    acl_close(ps);
108    pause(2);
109    acl_move(ps, 'apdep');
110    pause(2);
111    acl_move(ps, 'aptab');
112    pause(2);
113    acl_move(ps, 'p6a');
114    pause(2);
115    acl_move(ps, 'p6');
116    pause(2);
117    acl_open(ps)
118    pause(2);
119    acl_move(ps, 'aptab')
120    pause(1);
121
122    acl_open(ps);
123    pause(2);
124    acl_move(ps, 'apdep');
125    pause(2);
126    acl_move(ps, 'pr3a');
127    pause(2);
128    acl_move(ps, 'pr3');
129    pause(2);
130    acl_close(ps);
131    pause(2);
132    acl_move(ps, 'apdep');
133    pause(2);
134    acl_move(ps, 'aptab');
135    pause(2);
136    acl_move(ps, 'p9a');
137    pause(2);
138    acl_move(ps, 'p9');
139    pause(2);
140    acl_open(ps)
141    pause(2);
142    acl_move(ps, 'aptab')
143    pause(1);
144
145    acl_open(ps);
146    pause(2);
147    acl_move(ps, 'apdep');
148    pause(2);
149    acl_move(ps, 'pb1a');
150    pause(2);
151    acl_move(ps, 'pb1');
152    pause(2);
153    acl_close(ps);
154    pause(2);
155    acl_move(ps, 'apdep');
156    pause(2);
157    acl_move(ps, 'aptab');
158    pause(2);
159    acl_move(ps, 'p2a');
160    pause(2);
161    acl_move(ps, 'p2');
162    pause(2);
163    acl_open(ps)
164    pause(2);
165    acl_move(ps, 'aptab')
166    pause(1);
167
168    acl_open(ps);
169    pause(2);
170    acl_move(ps, 'apdep');
171    pause(2);

172
173     acl_move(ps, 'pb2a');
174     pause(2);
175     acl_move(ps, 'pb2');
176     acl_close(ps);
177     pause(2);
178     acl_move(ps, 'apdep');
179     pause(2);
180     acl_move(ps, 'aptab');
181     pause(2);
182     acl_move(ps, 'p5a');
183     pause(2);
184     acl_move(ps, 'p5');
185     pause(2);
186     acl_open(ps)
187     pause(2);
188     acl_move(ps, 'aptab')
189     pause(1);
190
191     acl_open(ps);
192     pause(2);
193     acl_move(ps, 'apdep');
194     pause(2);
195     acl_move(ps, 'pb3a');
196     pause(2);
197     acl_move(ps, 'pb3');
198     pause(2);
199     acl_close(ps);
200     pause(2);
201     acl_move(ps, 'apdep');
202     pause(2);
203     acl_move(ps, 'aptab');
204     pause(2);
205     acl_move(ps, 'p8a');
206     pause(2);
207     acl_move(ps, 'p8');
208     pause(2);
209     acl_open(ps)
210     pause(2);
211     acl_move(ps, 'aptab')
212     pause(1);
213
214 end
215 if y==2
216     acl_open(ps);
217     pause(2);
218     acl_move(ps, 'apdep');
219     pause(2);
220     acl_move(ps, 'pala');
221     pause(2);
222     acl_move(ps, 'pai');
223     acl_close(ps);
224     pause(2);
225     acl_move(ps, 'apdep');
226     pause(2);
227     acl_move(ps, 'aptab');
228     pause(2);
229     acl_move(ps, 'pia');
230     pause(2);
231     acl_move(ps, 'p1');
232     pause(2);
233     acl_open(ps)
234     pause(2);
235     acl_move(ps, 'aptab')
236     pause(1);
237
238 end
239 if y==3
240     acl_open(ps);
241     pause(2);
242     acl_move(ps, 'pria');
243     pause(2);
244     acl_move(ps, 'pri1');
245     pause(2);
246     acl_close(ps);
247     pause(2);
248     acl_move(ps, 'apdep');
249     pause(2);
250     acl_move(ps, 'aptab');
251     pause(2);
252     acl_move(ps, 'p4a');
253     pause(2);
254     acl_move(ps, 'p4');
255     pause(2);
256     acl_open(ps)
257     pause(2);
258     acl_move(ps, 'aptab')
259     pause(1);
260
261 end
262 if y==4
263     acl_open(ps);
264     pause(2);
265     acl_move(ps, 'pr2a');
266     pause(2);
267     acl_move(ps, 'pr2');
268     pause(2);
269     acl_close(ps);
270     pause(2);
271     acl_move(ps, 'apdep');
272     pause(2);
273     acl_move(ps, 'aptab');
274     pause(2);
275     acl_move(ps, 'p5a');
276     pause(2);
277     acl_move(ps, 'p5');
278     pause(2);
279     acl_open(ps)
280     pause(2);
281     acl_move(ps, 'aptab')
282     pause(1);
283
284 end
285 if y==5
286     acl_open(ps);
287     pause(2);
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403

```

```

404    acl_move(ps, 'pa3a');
405    pause(2);
406    acl_move(ps, 'pa3');
407    pause(2);
408    acl_close(ps);
409    pause(2);
410    acl_move(ps, 'apdep');
411    pause(2);
412    acl_move(ps, 'aptab');
413    pause(2);
414    acl_move(ps, 'p6a');
415    pause(2);
416    acl_move(ps, 'p6');
417    pause(2);
418    acl_open(ps);
419    pause(2);
420    acl_move(ps, 'aptab')
421    pause(1);
422
423    acl_open(ps);
424    pause(2);
425    acl_move(ps, 'apdep');
426    pause(2);
427    acl_move(ps, 'pria');
428    pause(2);
429    acl_move(ps, 'pri');
430    pause(2);
431    acl_close(ps);
432    pause(2);
433    acl_move(ps, 'apdep');
434    pause(2);
435    acl_move(ps, 'aptab');
436    pause(2);
437    acl_move(ps, 'p7a');
438    pause(2);
439    acl_move(ps, 'p7');
440    pause(2);
441    acl_open(ps)
442    pause(2);
443    acl_move(ps, 'aptab')
444    pause(1);
445
446    acl_open(ps);
447    pause(2);
448    acl_move(ps, 'apdep');
449    pause(2);
450    acl_move(ps, 'pr2a');
451    pause(2);
452    acl_move(ps, 'pr2');
453    pause(2);
454    acl_close(ps);
455    pause(2);
456    acl_move(ps, 'apdep');
457    pause(2);
458    acl_move(ps, 'aptab');
459    pause(2);
460    acl_move(ps, 'p8a');
461    pause(2);
462    acl_move(ps, 'p8');
463    pause(2);
464    acl_open(ps)
465    pause(2);
466    acl_move(ps, 'aptab')
467    pause(1);
468
469    acl_open(ps);
470    pause(2);
471    acl_move(ps, 'apdep');
472    pause(2);
473    acl_move(ps, 'pr3a');
474    pause(2);
475    acl_move(ps, 'pr3');
476    pause(2);
477    acl_close(ps);
478    pause(2);
479    acl_move(ps, 'apdep');
480    pause(2);
481    acl_move(ps, 'aptab');
482    pause(2);
483    acl_move(ps, 'p9a');
484    pause(2);
485    acl_move(ps, 'p9');
486    pause(2);
487    acl_open(ps)
488    pause(2);
489    acl_move(ps, 'aptab')
490    pause(1);
491
492    acl_open(ps);
493    pause(2);
494    acl_move(ps, 'apdep');
495    pause(2);
496    acl_move(ps, 'pbia');
497    pause(2);
498    acl_move(ps, 'pb1');
499    pause(2);
500    acl_close(ps);
501    pause(2);
502    acl_move(ps, 'apdep');
503    pause(2);
504    acl_move(ps, 'aptab');
505    pause(2);
506    acl_move(ps, 'p1a');
507    pause(2);
508    acl_move(ps, 'p1');
509    pause(2);
510    acl_open(ps)
511    pause(2);
512    acl_move(ps, 'aptab')
513    pause(1);
514
515    acl_open(ps);
516    pause(2);
517    acl_move(ps, 'apdep');
518    pause(2);
519    acl_move(ps, 'pb2a');

520    pause(2);
521    acl_move(ps, 'pb2');
522    pause(2);
523    acl_close(ps);
524    pause(2);
525    acl_move(ps, 'apdep');
526    pause(2);
527    acl_move(ps, 'aptab');
528    pause(2);
529    acl_move(ps, 'p2a');
530    pause(2);
531    acl_move(ps, 'p2');
532    pause(2);
533    acl_open(ps)
534    pause(2);
535    acl_move(ps, 'aptab')
536    pause(1);
537
538    acl_open(ps);
539    pause(2);
540    acl_move(ps, 'apdep');
541    pause(2);
542    acl_move(ps, 'pb3a');
543    pause(2);
544    acl_move(ps, 'pb3');
545    pause(2);
546    acl_close(ps);
547    pause(2);
548    acl_move(ps, 'apdep');
549    pause(2);
550    acl_move(ps, 'aptab');
551    pause(2);
552    acl_move(ps, 'p3a');
553    pause(2);
554    acl_move(ps, 'p3');
555    pause(2);
556    acl_open(ps)
557    pause(2);
558    acl_move(ps, 'aptab')
559    pause(1);
560
561    end
562    if y==4
563        acl_open(ps);
564        pause(2);
565        acl_move(ps, 'apdep');
566        pause(2);
567        acl_move(ps, 'pria');
568        pause(2);
569        acl_move(ps, 'pri');
570        pause(2);
571        acl_close(ps);
572        pause(2);
573        acl_move(ps, 'apdep');
574        pause(2);
575        acl_move(ps, 'aptab');
576        pause(2);
577        acl_move(ps, 'p4a');
578        pause(2);
579        acl_move(ps, 'p4');
580        acl_open(ps)
581        pause(2);
582        acl_move(ps, 'aptab')
583        pause(1);
584
585        acl_open(ps);
586        pause(2);
587        acl_move(ps, 'apdep');
588        pause(2);
589        acl_move(ps, 'pr2a');
590        pause(2);
591        acl_move(ps, 'pr2');
592        pause(2);
593        acl_close(ps);
594        pause(2);
595        acl_move(ps, 'apdep');
596        pause(2);
597        acl_move(ps, 'aptab');
598        pause(2);
599        acl_move(ps, 'p5a');
600        pause(2);
601        acl_move(ps, 'p5');
602        pause(2);
603        acl_open(ps)
604        pause(2);
605        acl_move(ps, 'aptab')
606        pause(1);
607
608        acl_open(ps);
609        pause(2);
610        acl_move(ps, 'apdep');
611        pause(2);
612        acl_move(ps, 'pr3a');
613        pause(2);
614        acl_move(ps, 'pr3');
615        pause(2);
616        acl_close(ps);
617        pause(2);
618        acl_move(ps, 'apdep');
619        pause(2);
620        acl_move(ps, 'aptab');
621        pause(2);
622        acl_move(ps, 'p6a');
623        pause(2);
624        acl_move(ps, 'p6');
625        pause(4);
626        acl_open(ps)
627        pause(2);
628        acl_move(ps, 'aptab')
629        pause(1);
630
631        acl_open(ps);
632        pause(2);
633        acl_move(ps, 'apdep');
634        pause(2);
635        acl_move(ps, 'pbia');

636    pause(2);
637    acl_move(ps, 'pb1');
638    pause(2);
639    acl_close(ps);
640    pause(2);
641    acl_move(ps, 'apdep');
642    pause(2);
643    acl_move(ps, 'aptab');
644    pause(2);
645    acl_move(ps, 'pia');
646    pause(2);
647    acl_move(ps, 'p1');
648    pause(2);
649    acl_open(ps)
650    pause(2);
651    acl_move(ps, 'aptab')
652    pause(1);
653
654    acl_open(ps);
655    pause(2);
656    acl_move(ps, 'apdep');
657    pause(2);
658    acl_move(ps, 'pb2a');
659    pause(2);
660    acl_move(ps, 'pb2');
661    pause(2);
662    acl_close(ps);
663    pause(2);
664    acl_move(ps, 'apdep');
665    pause(2);
666    acl_move(ps, 'aptab');
667    pause(2);
668    acl_move(ps, 'p2a');
669    pause(2);
670    acl_move(ps, 'p2');
671    pause(2);
672    acl_open(ps)
673    pause(2);
674    acl_move(ps, 'aptab')
675    pause(1);
676
677    acl_open(ps);
678    pause(2);
679    acl_move(ps, 'apdep');
680    pause(2);
681    acl_move(ps, 'pb3a');
682    pause(2);
683    acl_move(ps, 'pb3');
684    pause(2);
685    acl_close(ps);
686    pause(2);
687    acl_move(ps, 'apdep');
688    pause(2);
689    acl_move(ps, 'aptab');
690    pause(2);
691    acl_move(ps, 'p3a');
692    pause(2);
693    acl_move(ps, 'p3');
694    pause(2);
695    acl_open(ps)
696    pause(2);
697    acl_move(ps, 'aptab')
698    pause(1);
699
700    end
701    if y==5
702        acl_open(ps);
703        pause(2);
704        acl_move(ps, 'apdep');
705        pause(2);
706        acl_move(ps, 'pala');
707        pause(2);
708        acl_move(ps, 'pal');
709        pause(2);
710        acl_close(ps);
711        pause(2);
712        acl_move(ps, 'apdep');
713        pause(2);
714        acl_move(ps, 'aptab');
715        pause(2);
716        acl_move(ps, 'p7a');
717        pause(2);
718        acl_move(ps, 'p7');
719        acl_open(ps)
720        pause(2);
721        acl_move(ps, 'aptab')
722        pause(1);
723
724        acl_open(ps);
725        pause(2);
726        acl_move(ps, 'apdep');
727        pause(2);
728        acl_move(ps, 'p2a');
729        pause(2);
730        acl_move(ps, 'pa2');
731        pause(2);
732        acl_close(ps);
733        pause(2);
734        acl_move(ps, 'apdep');
735        pause(2);
736        acl_move(ps, 'aptab');
737        pause(2);
738        acl_move(ps, 'p8a');
739        pause(2);
740        acl_move(ps, 'p8');
741        pause(2);
742        acl_open(ps)
743        pause(2);
744        acl_move(ps, 'aptab')
745        pause(1);
746
747        acl_open(ps);
748        pause(2);
749        acl_move(ps, 'apdep');
750        pause(2);
751        acl_move(ps, 'p3a');


```

752	pause (2);	805	acl_move(ps, 'aptab');	858	pause (2);
753	acl_move(ps, 'pa3');	806	pause (2);	859	acl_move(ps, 'aptab')
754	pause (2);	807	acl_move(ps, 'p2a');	860	pause (1);
755	acl_close(ps);	808	pause (2);	861	acl_open(ps);
756	pause (2);	809	acl_move(ps, 'p2');	862	pause (2);
757	acl_move(ps, 'apdep');	810	pause (2);	863	acl_move(ps, 'apdep');
758	pause (2);	811	acl_open(ps)	864	pause (2);
759	acl_move(ps, 'aptab');	812	pause (2);	865	acl_move(ps, 'pb2a');
760	pause (2);	813	acl_move(ps, 'aptab')	866	pause (2);
761	acl_move(ps, 'p9a');	814	pause (1);	867	acl_close(ps);
762	pause (2);	815	acl_open(ps);	868	pause (2);
763	acl_move(ps, 'p9');	816	pause (2);	869	acl_move(ps, 'apdep');
764	pause (2);	817	acl_move(ps, 'apdep')	870	pause (2);
765	acl_open(ps)	818	pause (2);	871	acl_move(ps, 'apdep');
766	pause (2);	819	acl_move(ps, 'aptab')	872	pause (2);
767	acl_move(ps, 'aptab');	820	pause (2);	873	acl_move(ps, 'pr3a');
768	pause (1);	821	acl_move(ps, 'pr3');	874	pause (2);
769	acl_open(ps);	822	pause (2);	875	acl_move(ps, 'aptab');
770	pause (2);	823	acl_close(ps);	876	pause (2);
771	acl_move(ps, 'apdep');	824	pause (2);	877	acl_move(ps, 'p5a');
772	pause (2);	825	acl_move(ps, 'apdep')	878	pause (2);
773	acl_move(ps, 'pri');	826	pause (2);	879	acl_open(ps);
774	pause (2);	827	acl_move(ps, 'aptab');	880	pause (2);
775	acl_move(ps, 'pri');	828	pause (2);	881	acl_move(ps, 'p5');
776	pause (2);	829	acl_move(ps, 'p3a');	882	pause (1);
777	acl_close(ps);	830	pause (2);	883	acl_move(ps, 'aptab')
778	pause (2);	831	acl_move(ps, 'p3');	884	end
779	acl_move(ps, 'apdep');	832	pause (2);	885	
780	pause (2);	833	acl_open(ps);	886	
781	acl_move(ps, 'aptab');	834	pause (2);	887	
782	pause (2);	835	acl_move(ps, 'aptab')	888	
783	pause (1);	836	pause (1);	889	
784	acl_move(ps, 'p1a');	837	acl_move(ps, 'p1');	890	
785	pause (2);	838	pause (2);	891	
786	acl_move(ps, 'p1');	839	acl_open(ps);	892	
787	pause (2);	840	pause (2);	893	
788	acl_open(ps)	841	acl_move(ps, 'apdep');	894	
789	pause (2);	842	pause (2);	895	
790	acl_move(ps, 'aptab');	843	acl_move(ps, 'pb1a');	896	
791	pause (1);	844	pause (2);	897	
792	acl_open(ps);	845	acl_move(ps, 'pb1');	898	
793	pause (2);	846	pause (2);	899	
794	acl_move(ps, 'apdep');	847	acl_close(ps);	900	
795	pause (2);	848	pause (2);	901	
796	acl_move(ps, 'pr2a');	849	acl_move(ps, 'apdep');	902	
797	pause (2);	850	pause (2);	903	
798	acl_move(ps, 'pr2');	851	acl_move(ps, 'aptab');	904	
799	pause (2);	852	pause (2);	905	
800	acl_close(ps);	853	acl_move(ps, 'p4a');	906	
801	pause (2);	854	pause (2);	907	
802	acl_move(ps, 'apdep');	855	acl_move(ps, 'p4');		
803	pause (2);	856	pause (2);		
804	acl_open(ps)	857	acl_open(ps)		

5.5. Código Main

```

1 clc; clear;
2
3 % Inicializaci n
4 disp('Iniciando robot.');
5
6 ps=Inicializa
7
8 % Declarar contadores como variables globales
9 global ContAzul ContRojo ContBlanco;
10 ContAzul = 0; ContRojo = 0; ContBlanco = 0;
11
12 cam = webcam; % Configuraci n de la c mara
13 preview(cam);
14
15 % Procesar 9 piezas
16 for i = 1:9
17     disp(['Procesando pieza ', num2str(i), ' de 9.']);
18
19     % Captura de imagen
20     consola = lower(input("Presione R para tomar una foto: ", 's'));
21     if consola == 'r'
22         img = snapshot(cam);
23         imshow(img); shg;
24         colorPieza = detectaColorPieza(img);
25         disp(colorPieza);
26
27     % Verificaci n y movimiento inicial de la pieza
28     Foto = 1; % Marca que se tom la foto
29     if Foto == 1
30         entradas = fc_leer_entradas(ps);
31         pause(0.25);
32         fc_set_anout(ps, 7, -1300); % Activa el movimiento del actuador
33         pause(0.25);
34
35     % Espera hasta que el sensor detecte la pieza
36     while entradas(2) == 0
37         entradas = fc_leer_entradas(ps);
38         pause(0.01);
39     end
40
41     % Si el sensor detecta la pieza, detener el actuador y llamar a LlevarPiezasDep
42     if entradas(2) == 1
43         disp('Detecta pieza en posici n.');
44         Foto = 0; % Resetea la marca
45         fc_set_anout(ps, 7, 0); % Detiene el actuador
46         pause(3);
47         disp('Ejecuto LlevarPiezas.');
48         LlevarPiezasDep(ps, img);
49     end
50
51 else
52     disp('Int ntalo de nuevo.');
53     i = i - 1; % No cuenta esta iteraci n
54 end
55 end
56
57 disp('Todas las piezas han sido procesadas.');
58 acl_open(ps);
59 %Aqui acaba el modo CINTA, de esta manera, empieza el selector de modos
60
61 while true
62     disp('Seleccione el modo que desea ejecutar:');
63     disp('1. Reconocimiento');
64     disp('2. Bandera');
65     disp('3. Salir');
66     opcion = input('Ingrese su opci n: ');
67
68 switch opcion
69     case 1
70         completa_NuevaR();
71     case 2
72         disp('Modo Bandera seleccionado.');
73         modoBandera(ps);
74     case 3
75         disp('Bye Bye.');
76         break;
77     otherwise
78         disp('Opc i n inv lida. Por favor, intente de nuevo.');
79 end
80 end

```