Universitat de Vic
Escola Politècnica Superior

CODIGO FUENTE

Manipulador robótico con visión artificial (CableBot)

Alberto López Germán

Grado en Ingeniería Electrónica Industrial y Automática
Tutor: Pere Martí Puig
Vic, Septiembre de 2014

INDICE

1. Introducción 2

1.1. Objeto 2

1.2. Alcance 2

2. Códigos fuente 3

2.1. Código fuente software de visión artificial 3

2.1.1. Diagrama de bloques 4

2.2. Código fuente firmware 5

2.2.1. Diagrama de bloques 5

3. Anexos 6

3.1. Código software MATLAB 6

3.1.1. Presentación.m 6

3.1.2. CableBOT.m 7

3.1.3. Calibration.m 26

3.1.4. Motor_Control.m 27

3.1.5. Gamepad_Control.m 32

3.1.6. Scale.m 35

3.1.7. Color.m 36

3.1.8. Place.m 37

3.1.9. Shape.m 39

3.2. Código firmware de control 42

1. Introducción

Una de las partes más importantes en un controlador robótico junto con el propio sistema es la parte destinada al control, ya sea implementado en un PC o en un sistema embebido.
En el caso que a este proyecto atañe, se han realizado dos implementaciones de carácter informático. Por un lado se ha desarrollado una aplicación software de visión artificial sobre el programa MATLAB y otra implementación sobre un microcontrolador en ANSI C para realizar el control del hardware del manipulador robótico.

1.1. Objeto

Como complemento al documento de MEMORIA se ha redactado el presente documento con el fin de explicar de forma más detallada el funcionamiento del código fuente tanto del software de control *CableBot V1.0* como el firmware implementado en el microcontrolador elegido para realizar el control del manipulador robótico.

1.2. Alcance

El documento se centra en explicar el proceso seguido en cada una de las implementaciones informáticas de una forma visual con el fin de comprender el funcionamiento del mismo.
Este documento también muestra el código fuente de ambas implementaciones con el fin de tener recopilada toda la información en un solo documento.

Para saber el funcionamiento del software de visión artificial con más detalle es necesario ver el documento ***Manual de usuario*** en el cual se puede apreciar todos los pasos a seguir para poner en marcha el software así como la forma de proceder para la identificación de las formas elegidas.

2. Códigos fuente

Este apartado intenta describir de una forma visual los códigos fuente implementados en ambos dispositivos, ya sea el PC para el software de visión artificial como el microcontrolador encargado de la gestión del software.

2.1. Código fuente software de visión artificial

Como ya se ha comentado anteriormente, el software de control de visión artificial se ha implementado bajo el programa MATLAB el cual consta de diferentes subrutinas o procedimientos con la extensión propia del programa *.m.
Cada uno de los procedimientos es el encargado de una función concreta, de forma que si se requiere la modificación de una de las partes solamente se accede a dicho procedimiento a realizar la modificación pertinente quedando el resto del programa intacto para evitar posibles conflictos.
Los archivos que forman dicho software son los citados a continuación:

RELACION DE ARCHIVOS	
NOMBRE	**DESCRIPCION**
Presentación.m	Archivo inicial en el cual aparece el botón de inicio del sistema
CableBOT.m	Archivo que nos ofrece la visualización de la pantalla principal del programa en la cual se realiza la adquisición y tratamiento de la imagen.
Calibration.m	Procedimiento encargado de la calibración de la cámara respecto a la zona de trabajo.
Motor_Control.m	Archivo que muestra en pantalla la ventana de control del motor para comenzar el ciclo de almacenamiento de las fichas.
Gamepad_Control.m	Toma de control del actuador desde una joystick inalámbrico de juegos para PC para dar más amplitud al proyecto.
Scale.m	Procedimiento que escala la imagen de forma que pasa de trabajar en pixeles a trabajar en milímetros.
Color.m	Detección del color de las fichas usadas en la maqueta, bien amarillo o rojo.
Place.m	Detección del centroide de cada una de las figuras.
Shape.m	Discriminación de las figuras según su forma. Cuadrada, triangular o circular.

2.1.1. Diagrama de bloques

Diagrama de bloques en el cual se puede observar el proceso que sigue el software de visión artificial para llevar a cabo su cometido, el de discriminar las figuras situadas en la zona de trabajo.

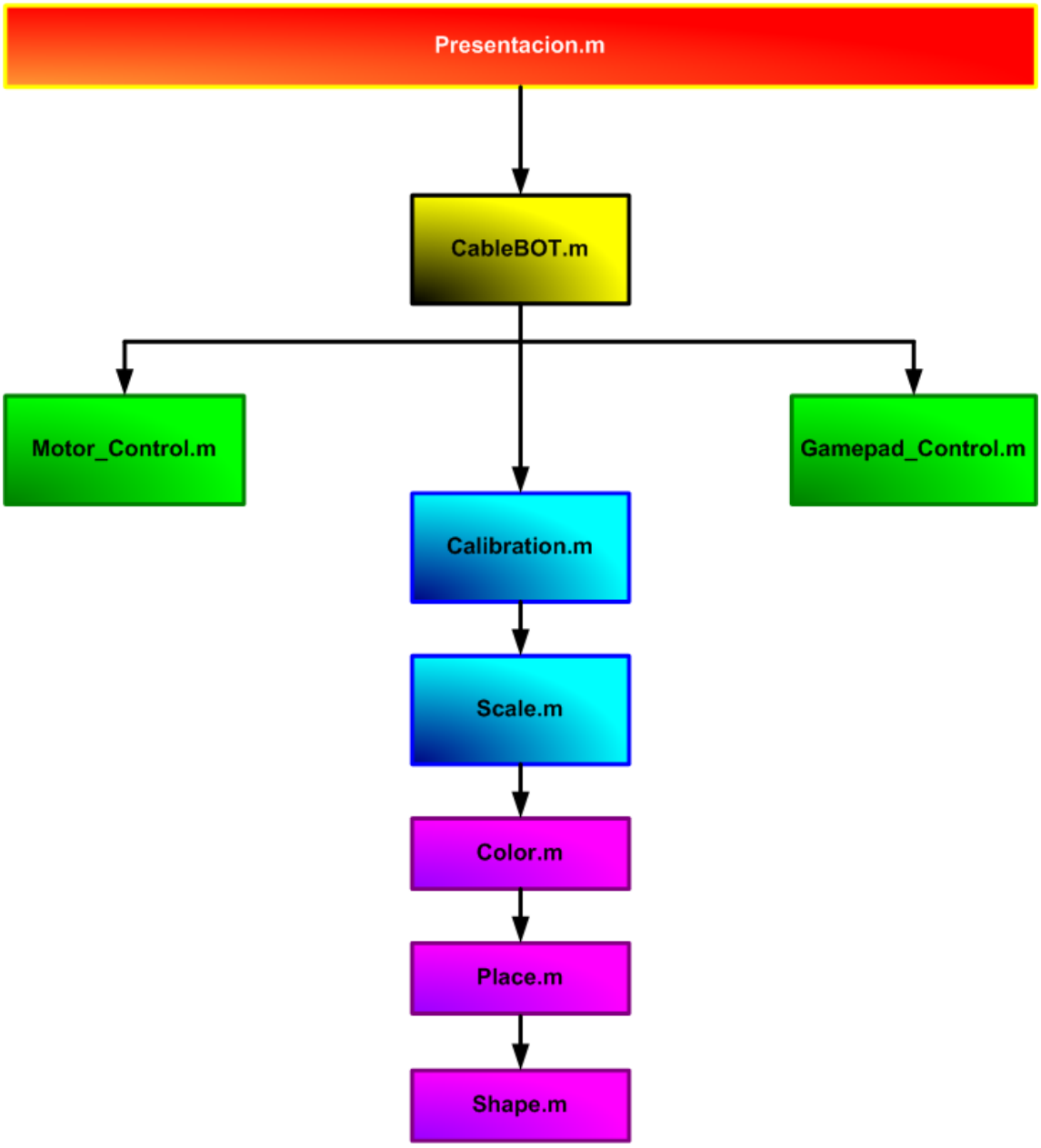

Figura 2.1: Diagrama de bloques del software de visión artificial

2.2. Código fuente firmware

De igual forma que el punto anterior, en este se quiere representar de forma grafica la secuencia que sigue el firmware para llevar a cabo el almacenaje de las piezas ya sea de forma manual como automática.

2.2.1. Diagrama de bloques

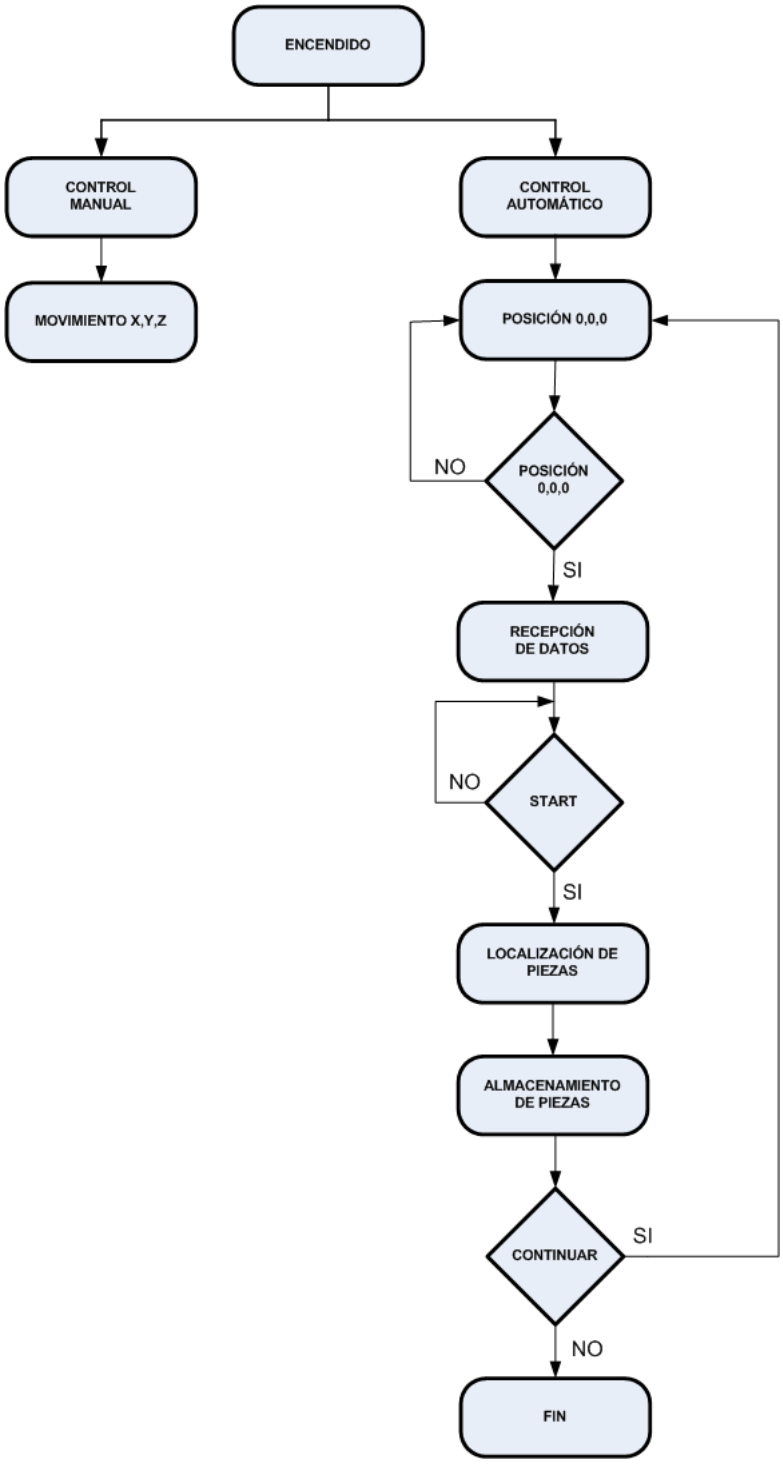

Figura 2.2: Diagrama de bloques del firmware de control

3. Anexos

3.1. Código software MATLAB

3.1.1. Presentación.m

```
function presentacion
%**************************************************************************
%Autor:  Alberto Lopez German
%Centro: Universidad Politecnica de Vic
%Fecha:  08/09/2013
%Descripcion: Función que inicia el programa con una ventana de
%             presentación antes de lanzar el programa principal
%             con el motivo de mostrar información relativa al programa
%**************************************************************************
%% Inicializacion del programa
clear,clc,cla,close all   %Borramos todo lo que pueda haber abierto
warning off all           % Suprimir avisos

%% Creamos figura
figdiag=figure('Units','Pixels',...
'Position',[1 1 1024 768],...%Tamaño de la presentación
'Number','off',...
'Name','CableBOT', ...
'Menubar','none', ...
'color',[0 0 0]);

%% Ubicamos ejes en figura
axes('Units','Normalized',...
'Position',[0 0 1 1]);

%% Incluir imagen de fondo
%Importamos imagen *.jpg,junto con su mapa de colores
[x,map]=imread('fondo_presentacion.jpg','jpg');
%Representamos imagen en figura, con su mapa de colores
image(x),colormap(map),axis off,hold on

%% Títulos sobre imagen
%Título
%text(220,50,'CableBOT','Fontname','Arial','Fontsize',60,...
%'Fontweight','Bold','color','b');
%Nombre del autor
text(30,590,'Autor: Alberto Lopez German','Fontname',...
'Arial','Fontweight','Bold',...
'Fontsize',14,'color','r');
%Nombre del curso
text(30,620,'Curso: Electronica industrial y Automatica','Fontname',...
'Arial','Fontweight','Bold',...
'Fontsize',14,'color','r');
%Nombre del centro
text(30,650,'Centro: UNIVERSIDAD DE VIC','Fontname',...
'Arial','Fontweight','Bold',...
'Fontsize',14,'color','r');

%% Botón Comenzar
Continue=uicontrol('Style','pushbutton', ...
'Units','normalized', ...
```

```
'Position',[.82 .15 .15 .07], ...
'String','COMENZAR',...
'Fontsize',14,...
'BackgroundColor',[0 1 0],...
'Callback','clear all; close all;clc; CableBOT;');
```

3.1.2. CableBOT.m

```
function varargout = CableBOT(varargin)

%*************************************************************************
%Autor:  Alberto Lopez German
%Centro: Universidad Politecnica de Vic
%Fecha:  08/07/2014
%Descripcion:   Programa principal de configuracion de una GUIDE en MatLab
%               Se trata de la pantalla principal de la aplicacion en la
%               cual tenemos todos los controles encargados de la deteccion
%               de piezas de nuestro proyecto.
%*************************************************************************

% Last Modified by GUIDE v2.5 25-Apr-2014 22:17:53

% ******************Begin initialization code - DO NOT EDIT**************

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CableBOT_OpeningFcn, ...
                   'gui_OutputFcn',  @CableBOT_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% ***************End initialization code - DO NOT EDIT*********************



% --- Executes just before CableBOT is made visible.
function CableBOT_OpeningFcn(hObject, eventdata, handles, varargin)

% En esta funcion se colocan las instrucciones que queremos que se ejecuten
% antes de realizar ninguna accion, es decir, se ejecutan nada mas abrir la
% aplicacion GUIDE

%% Inicio de variables
global threshold
global panoramica_CAM
global azimutal_CAM
azimutal_CAM = 1;
panoramica_CAM = 2;
```

```
clc
warning off
threshold=0;

%% Ruido de encendido
% cargamos el archivo wav
[y,Fs,NBITS]=wavread('.\Sonidos\Entrada de Windows XP.wav'); %carga los datos
del archivo wav a MATLAB
sound(y, Fs); %Reproduce el sonido cargado

%% Insertar Logo de la UVic
%Con estas instrucciones leo la imagen del logo y la represento en el axes
%correspondiente
%logo=imread('C:\Documents and
Settings\LOPEZ\Escritorio\GRADO\PFG\CableBOT\GUI MatLab\Logos\logoHome.jpg');
logo=imread('.\Logos\logoHome.jpg');
image(logo)
axis off

%% Personalizar boton SALIR
%Insertar imagen en el boton cerrar de forma que vemos el simbolo salir en
%el boton
[a,map]=imread('.\Logos\cerrar.jpg');
[r,c,d]=size(a);
x=ceil(r/50);
y=ceil(c/50);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.1*255;
set(handles.Salir,'CData',g);

%% Ocultar boton de lamparas_off al inicio
%En el inicio del programa oculto el boton de apagar lamparas para que
%aparezca por encima el boton de encender lamparas
set(handles.Lamparas_OFF,'visible','off');

%% Ocultar panel de herramientas de imagen
set(handles.uipanel7,'Visible','off')%Visibilidad OFF del panel de
herramientas de imagen

%% Ocultar tabla de resultados
set(handles.uitable1,'Visible','off')%Visibilidad OFF de la tabla

%% Ocultar botones de deteccion de COLOR, PLACE y SHAPE
%set(handles.deteccion_color,'enable','off');
set(handles.deteccion_lugar,'enable','off');
set(handles.deteccion_forma,'enable','off');

%% Centrar ventana del GUIDE
%Con estas instrucciones realizamos un centrado de la pantalla de la GUIDE
%de forma que tenga una mejor vision de cara al usuario
scrsz=get(0,'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3)-pos_act(3);
xp=round(xr/2);
yr=scrsz(4)-pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
```

```
%% Deshabilitar botones de TOOLS
% Hasta que no carguemos una imagen en la zona de SbapShot no podemos
% actuar sobre la paleta de herramientas
set(handles.tools_on,'Enable','off'); %Habilitamos el boton ON para volver a
activar el panel
set(handles.tools_off,'Enable','off'); %Deshabilitamos el boton de OFF porque
ya no es visible el panel

%% Deshabilitar boton START (Comienzo de transferencia de datos)
set(handles.comienzo_transferencia,'Enable','off'); %Deshabilitamos el boton
de OFF porque ya no es visible el panel

%% Guardamos los handles
handles.output = hObject;
guidata(hObject, handles);

%% Deshabilita el boton de filtro B&W hasta que no este en escala de grises
%Comienza la GUI con el boton deshabilitado. Sigue en boton imagen_gris
set(handles.imagen_blanco_negro,'Enable','off');

% UIWAIT makes CableBOT wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = CableBOT_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Captura_Imagen.
function Captura_Imagen_Callback(hObject, eventdata, handles)

%% Captura de imagen mediante WebCam
%Parte del programa encargada de la captura de la imagen de la webcam
%Usamos las sentencias 'try' y 'catch' para evaluar si hay o no webcam
conectada y de este
%modo mostrar un mensaje de error en pantalla

global original_imagen_cam
global azimutal_CAM

%% Ocultar tabla de resultados
set(handles.uitable1,'Visible','off')%Visibilidad OFF de la tabla

    try
        canalVideo=videoinput('winvideo',azimutal_CAM,'RGB24_800x600'); % esto
crea el canal de video a la camara web
        preview(canalVideo); % abrimos una ventana de preview para ver a donde
apuntamos con la camara
    catch
```

```
        errordlg('NO HAY CAMARA CONECTADA', 'Mensaje de error')
    end
    start(canalVideo); % inicializamos el canal de Video
    pause(5); % espera 5 segundos para que estabilice la imagen de la webcam
    imgAdq=getsnapshot(canalVideo); % tomamos una instantanea con la camara
que se guarda en imgAdq

    %Guardamos la imagen capturada en el PC
    imwrite(imgAdq,'.\Imagenes CAM\imagenCam.jpg','jpg');

    %Guardamos la imagen en la variable global para usarla donde nosotros
    %queramos dentro de la GUIDE
    original_imagen_cam = imread('.\Imagenes CAM\imagenCam.jpg');

%% Ruido de camara
% cargamos el archivo wav
[y,Fs,NBITS]=wavread('.\Sonidos\CAMERA.wav'); %carga los datos del archivo wav
a MATLAB
sound(y, Fs); %Reproduce el sonido cargado

%% Cerrar video
closepreview(canalVideo); % cerramos la ventana de preview
delete(canalVideo); % borramos el canal de Video

set(handles.text1,'String','SNAPSHOT'); % Cambio el texto de la imagen de ZONA
DE CAPTURA a IMAGEN CAPTURADA
%% Zona de visualizacion de la imagen capturada
%Creamos la zona en la cual se mostrará la imagen capturada
    axes(handles.Zona_de_Captura); % zona de captura
    background = imread('.\Imagenes CAM\imagenCam.jpg'); % cargamos la imagen
capturada
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada

%% Habilitamos la paleta de herramientas de imagen (Tools Imagen)
set(handles.tools_on,'Enable','on'); %Habilitamos el boton ON para volver a
activar el panel

    %Guarmos la variable canalVideo para usarla en otra funcion
    handles.canalVideo=canalVideo;
    guidata(hObject,handles);



% --- Executes on button press in Abrir_Video.
function Abrir_Video_Callback(hObject, eventdata, handles)

global azimutal_CAM
%% Abrir Video para visualizacion del proceso

    canalVideo=videoinput('winvideo',azimutal_CAM,'RGB24_800x600'); % esto
crea el canal de video a la camara web
    preview(canalVideo); %// abrimos una ventana de preview para ver a donde
apuntamos con la camara
    start(canalVideo); %// inicializamos el canal de Video

    handles.canalVideo=canalVideo;
    guidata(hObject,handles);
```

```
% --- Executes on button press in Lamparas_ON.
function Lamparas_ON_Callback(hObject, eventdata, handles)
% hObject    handle to Lamparas_ON (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Abrir y configurar puerto Serie COM1
%Secuencia de instrucciones para la configuracion del serial port
Serial_PORT = serial('COM1');
fclose(Serial_PORT);
set(Serial_PORT,'Baudrate',9600);
set(Serial_PORT,'Terminator','CR/LF');
set(Serial_PORT,'DataBits',8);
set(Serial_PORT,'Parity','none');
set(Serial_PORT,'StopBits',1);
set(Serial_PORT,'FlowControl','none');
fopen(Serial_PORT);

%% Envio de datos a traves del puerto serie
% Segun el checkbox activado se manda un caracter u otro para activar un
% circuito o dos circuitos de iluminacion
circuit_1=get(handles.circuito_1,'Value');
circuit_2=get(handles.circuito_2,'Value');

if circuit_1 && circuit_2              %Si los dos checkbox estan habilitados
    fprintf(Serial_PORT,'%s\n','A');   %Envio 'A' por el serial
    set(handles.circuito_1,'Enable','off') %Deshabilito los checkbox
    set(handles.circuito_2,'Enable','off')
    % Cerramos y borramos el puerto serie para poder usarlo de nuevo
    fclose(Serial_PORT);
    delete(Serial_PORT);
    clear Serial_PORT

elseif circuit_1 && ~(circuit_2)       %Si esta un circuito (circuito_1)
activado
    fprintf(Serial_PORT,'%s\n','B');   %Envio 'B' por el serial
    set(handles.circuito_1,'Enable','off') %Deshabilito los checkbox
    set(handles.circuito_2,'Enable','off')
    % Cerramos y borramos el puerto serie para poder usarlo de nuevo
    fclose(Serial_PORT);
    delete(Serial_PORT);
    clear Serial_PORT

elseif ~(circuit_1) && circuit_2       %Si esta un circuito (circuito_2)
activado
    fprintf(Serial_PORT,'%s\n','C');   %Envio 'C' por el serial
    set(handles.circuito_1,'Enable','off') %Deshabilito los checkbox
    set(handles.circuito_2,'Enable','off')
    % Cerramos y borramos el puerto serie para poder usarlo de nuevo
    fclose(Serial_PORT);
    delete(Serial_PORT);
    clear Serial_PORT

else   %Si no hay ningun checkbox habilitado envio msg de error
    errordlg('Selecciona un circuito de encendido','ERROR');
    % Cerramos y borramos el puerto serie para poder usarlo de nuevo
```

```
    fclose(Serial_PORT);
    delete(Serial_PORT);
    clear Serial_PORT
    return
end

%% Instrucciones para mostrar u ocultar los botones de ON/OFF de iluminacion
set(handles.Lamparas_ON,'visible','off');
set(handles.Lamparas_OFF,'visible','on');



% --- Executes on button press in Lamparas_OFF.
function Lamparas_OFF_Callback(hObject, eventdata, handles)
% hObject    handle to Lamparas_OFF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Abrir y configurar puerto Serie COM1
Serial_PORT = serial('COM1');
fclose(Serial_PORT);
set(Serial_PORT,'Baudrate',9600);
set(Serial_PORT,'Terminator','CR/LF');
set(Serial_PORT,'DataBits',8);
set(Serial_PORT,'Parity','none');
set(Serial_PORT,'StopBits',1);
set(Serial_PORT,'FlowControl','none');
fopen(Serial_PORT);
%% Envio de datos a traves del puerto serie, en este caso el caracter 'B'
fprintf(Serial_PORT,'%s\n','D');
%% cerramos y borramos el puerto serie para poder usarlo de nuevo
fclose(Serial_PORT);
delete(Serial_PORT);
clear Serial_PORT

%% Instrucciones para mostrar u ocultar los botones de ON/OFF de iluminacion
set(handles.Lamparas_OFF,'visible','off');
set(handles.Lamparas_ON,'visible','on');

%% Borrado de los checkbox
set(handles.circuito_1,'Value',0)
set(handles.circuito_2,'Value',0)

%% Habilitado de los checkbox
% Usando la propiedad 'Enable' habilito los checkbox para poder seleccionar
% una opcion de nuevo
set(handles.circuito_1,'Enable','on')
set(handles.circuito_2,'Enable','on')

% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
% hObject    handle to Salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Mensaje que se muestra al presionar el boton inferior de SALIR
opc=questdlg('¿Desea salir del programa?',...
    'SALIR', 'SI','NO','NO');
if strcmp(opc,'NO')
```

```
    return;
end
%% Ruido de apagado
% cargamos el archivo wav
[y,Fs,NBITS]=wavread('.\Sonidos\Apagado de Windows XP.wav'); %carga los datos
del archivo wav a MATLAB
sound(y, Fs); %Reproduce el sonido cargado
delete(handles.figure1);



% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
delete(hObject);



% --- Executes on button press in deteccion_lugar.
function deteccion_lugar_Callback(hObject, eventdata, handles)
% hObject    handle to deteccion_lugar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global original_imagen_cam
global imagen_formas_amarillas
global imagen_formas_rojas
global numero_rojas
global numero_amarillas
global centroide_rojas
global centroide_amarillas
global num_element
global relacion

% global threshold
%% Deteccion de lugar de piezas amarillas
am=1;
[centroide_amarillas numero_amarillas]=place(imagen_formas_amarillas,
original_imagen_cam , am);
%disp(['El numero de formas AMARILLAS es: ', num2str(numero_amarillas)])
%disp('  Eje X     Eje Y')
%disp(centroide_amarillas)

%% Deteccion de lugar de piezas rojas
ro=0;
[centroide_rojas numero_rojas]=place(imagen_formas_rojas, original_imagen_cam
, ro);
%disp(['El numero de formas ROJAS es: ', num2str(numero_rojas)])
%disp('  Eje X     Eje Y')
%disp(centroide_rojas)



%% Configuramos el numero de filas-columnas de la tabla
filas=numero_rojas+numero_amarillas;
```

```
columnas=6;
num_element=cell(filas,columnas);

%% Cargamos el numero de elementos que tenemos
for n=1:filas
num_element(n,1)={n};
end

%% Cargamos coordenada X coordenada Y de AMARILLAS
for n=1:numero_amarillas
num_element(n,2)={round(((centroide_amarillas(n,1))/relacion))};
num_element(n,3)={round(((centroide_amarillas(n,2))/relacion))};
num_element(n,5)={'Amarilla'};
end

%% Cargamos coordenada X coordenada Y de ROJAS
x=1;
for n=(numero_amarillas+1):filas
num_element(n,2)={round(((centroide_rojas(x,1))/relacion))};
num_element(n,3)={round(((centroide_rojas(x,2))/relacion))};
num_element(n,5)={'Roja'};
x=x+1;
end

%% Visibilidad boton SHAPE
set(handles.deteccion_forma,'enable','on');



% --- Executes on button press in deteccion_forma.
function deteccion_forma_Callback(hObject, eventdata, handles)
% hObject    handle to deteccion_forma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global original_imagen_cam
global new_imagen_cam
global threshold
global num_element
global imagen_formas_amarillas
global imagen_formas_rojas
global n_amarillas
global n_rojas

shape(new_imagen_cam , original_imagen_cam , threshold);



%% Deteccion de formas amarillas
imagen_formas_amarillas=not(imagen_formas_amarillas);
imagen_formas_amarillas1 = bwareaopen(imagen_formas_amarillas,100);
[L N]=bwlabel(imagen_formas_amarillas1);
%figure, imshow(imagen_formas_amarillas1)
prop=regionprops(L,'all');
n_amarillas=length(prop);

figure, imshow(original_imagen_cam);
hold on
for k=1:n_amarillas
```

```
    if(prop(k).Area)>3900 && (prop(k).Area<5000)
       num_element(k,4)={'Circulo'};
       text(prop(k).Centroid(1)-50,prop(k).Centroid(2),'YELLOW
CIR','FontSize', 11,'Color','c','FontWeight', 'bold')
       disp(prop(k).Area);
    elseif (prop(k).Area)>5000 && (prop(k).Area<6500)
        num_element(k,4)={'Cuadrado'};
        text(prop(k).Centroid(1)-50,prop(k).Centroid(2),'YELLOW
SQ','FontSize', 11,'Color','c','FontWeight', 'bold')
        disp(prop(k).Area);
    elseif (prop(k).Area)>2000 && (prop(k).Area<3000)
        num_element(k,4)={'Triangulo'};
        text(prop(k).Centroid(1)-50,prop(k).Centroid(2),'YELLOW
TRI','FontSize', 11,'Color','c','FontWeight', 'bold')
        disp(prop(k).Area);
    end
end

%% Deteccion de formas rojas
imagen_formas_rojas1 = bwareaopen(imagen_formas_rojas,100);
[L N]=bwlabel(imagen_formas_rojas1);
%figure, imshow(imagen_formas_rojas1)
prop=regionprops(L,'all');
n_rojas=length(prop);
x=1;
for k=n_amarillas+1:n_amarillas+n_rojas
    if(prop(x).Area)>4000 && (prop(x).Area<5000)
       num_element(k,4)={'Circulo'};
       text(prop(x).Centroid(1)-40,prop(x).Centroid(2),'RED CIR','FontSize',
11,'Color','c','FontWeight', 'bold')
       disp(prop(x).Area);
    elseif (prop(x).Area)>5000 && (prop(x).Area<6500)
        num_element(k,4)={'Cuadrado'};
        text(prop(x).Centroid(1)-40,prop(x).Centroid(2),'RED SQ','FontSize',
11,'Color','c','FontWeight', 'bold')
        disp(prop(x).Area);
    elseif (prop(x).Area)>2000 && (prop(x).Area<3000)
        num_element(k,4)={'Triangulo'};
        text(prop(x).Centroid(1)-40,prop(x).Centroid(2),'RED TRI','FontSize',
11,'Color','c','FontWeight', 'bold')
        disp(prop(x).Area);
    end
    x=x+1;
end


%% Cargamos en tabla
set(handles.uitable1,'Data',num_element);

%% Visualizar tabla de resultados
set(handles.uitable1,'Visible','on')%Visibilidad ON de la tabla
set(handles.uitable1, 'RowName', []);
%set(handles.uitable1, 'ColumnWidth', {60 115 115 85 85 77});
%set(handles.uitable1,'ColumnFormat',{'short','bank'})
set(handles.text1,'String','TABLE'); % Cambio el texto de la imagen de IMAGEN
CAPTURADA a TABLA
```

```
% --- Executes on button press in deteccion_color.
function deteccion_color_Callback(hObject, eventdata, handles)
% hObject    handle to deteccion_color (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global new_imagen_cam %Rescatamos la imagen a utilizar

%% Creamos matriz del color a buscar
% Es una prueba practica, ya anteriormente se han buscado estos valores
% para que sean lo mas parecido posibles a los que buscamos
RGB_red=[0    0    0]; %REVISAR VALORES
RGB_yellow=[255   0    0]; %REVISAR VALORES

%% OBTENER IMAGENES DEL MISMO COLOR
% El valor de los pixeles de cada figura pueden variar levemente de uno a
% otro, asi que creamos un valor de umbral o histeresis. De esta forma
% tenemos el valor del pixel +/- este umbral para discriminar entre los dos
% colores.
% Es una prueba practica, de forma que puede variar dependiento de la
% imagen usada
color_umbral_red=70; % Valor empírico 70
color_umbral_yellow=70; % Valor empírico 70

%% Llamadas a funcion color
% En esta parte de codigo se realizan las llamadas correspondientes para
% realizar la discriminacion de color.
% Deteccion de piezas amarillas
global imagen_formas_amarillas
global imagen_formas_rojas

% Deteccion de piezas amarillas
imagen_formas_amarillas =
color(new_imagen_cam,RGB_yellow,color_umbral_yellow);
amarillas_1=not(imagen_formas_amarillas); %Invertir imagen
amarillas_2=bwareaopen(amarillas_1,700); %Eliminar ruido
se=strel('square',5);% Rellenar huecos vacios
amarillas_3=imclose(amarillas_2,se);
figure
subplot(1,2,1); imshow(new_imagen_cam);
title('Imagen ORIGINAL');
subplot(1,2,2); imshow(amarillas_3);
title('Piezas AMARILLAS');
helpdlg('Seguir proceso pulsando tecla',' CableBOT ');
pause();
% Deteccion de piezas rojas
imagen_formas_rojas_1 = color(new_imagen_cam,RGB_red,color_umbral_red);
imagen_formas_rojas_2=xor(imagen_formas_rojas_1,imagen_formas_amarillas);
imagen_formas_rojas_3=bwareaopen(imagen_formas_rojas_2,100);
se=strel('square',10);% Rellenar huecos vacios
imagen_formas_rojas=imclose(imagen_formas_rojas_3,se);
figure
subplot(1,2,1); imshow(new_imagen_cam);
title('Imagen ORIGINAL');
subplot(1,2,2); imshow(imagen_formas_rojas);
title('Piezas ROJAS');
pause();
```

```
%% Mensaje de aviso para continuar en la GUIDE
opc=questdlg('CONTINUAR CON BOTON PLACE',...
    'Deteccion Finalizada', 'SI','NO','SI');
if strcmp(opc,'SI')
   close figure 2;
   close figure 1;

else
    return
end

%% Visibilidad boton PLACE
set(handles.deteccion_lugar,'enable','on');



% --- Executes on button press in tools_on.
function tools_on_Callback(hObject, eventdata, handles)
% hObject    handle to tools_on (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.uipanel7,'Visible','on') %Visibilidad ON del panel de herramientas
de imagen
set(handles.tools_on,'Enable','off'); %Deshabilitamos el boton ON ya que no
tiene ninguna funcion
set(handles.tools_off,'Enable','on'); %Habilitamos el boton de OFF

% Control del Slider10 (Contraste)
set(handles.slider10,'Value',1); % Colocamos el valor a 1
handles.slider10=get(handles.slider10,'Value'); %Lo cargamos en 'Value'
set(handles.text7,'String',handles.slider10); %Escribe el valor de Slider en
statictext

% Control del Slider10 (Contraste)
set(handles.slider15,'Value',0); % Colocamos el valor a 1
handles.slider15=get(handles.slider15,'Value'); %Lo cargamos en 'Value'
set(handles.text10,'String',handles.slider15); %Escribe el valor de Slider en
statictext

% --- Executes on button press in tools_off.
function tools_off_Callback(hObject, eventdata, handles)
% hObject    handle to tools_off (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.uipanel7,'Visible','off')%Visibilidad OFF del panel de
herramientas de imagen
set(handles.tools_on,'Enable','on'); %Habilitamos el boton ON para volver a
activar el panel
set(handles.tools_off,'Enable','off'); %Deshabilitamos el boton de OFF porque
ya no es visible el panel

% --- Executes on button press in aceptar_valores.
function aceptar_valores_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar_valores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Con esta funcion lo que hacemos es asignar la nueva imagen que hemos
% creado para usarla en la deteccion de formas
%global imagen_cam
global new_imagen_cam
global new_imagen_cam_contrast

new_imagen_cam = new_imagen_cam_contrast;
%Guardamos la nueva imagen capturada en la carpeta del proyecto
imwrite(new_imagen_cam,'.\Imagenes CAM\new_imagenCam.jpg','jpg');
%Mensaje de confirmacion de seleccion de imagen
helpdlg('La imagen ha sido seleccionada',' CableBOT ');

%% Visibilidad del boton COLOR
set(handles.deteccion_color,'enable','on');



% --- Executes on button press in valores_defecto.
function valores_defecto_Callback(hObject, eventdata, handles)
% hObject    handle to valores_defecto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Control del slider8 (Brillo)
set(handles.slider8,'Value',0); % Colocamos el valor a 0
handles.slider8=get(handles.slider8,'Value'); %Lo cargamos en 'Value'
set(handles.text5,'String',handles.slider8); %Escribe el valor de Slider en
statictext

%% Control del slider10 (Contraste)
set(handles.slider10,'Value',1); % Colocamos el valor a 1
handles.slider10=get(handles.slider10,'Value'); %Lo cargamos en 'Value'
set(handles.text7,'String',handles.slider10); %Escribe el valor de Slider en
statictext


%% Control del slider15 (Umbral)
set(handles.slider15,'Value',0); % Colocamos el valor a 1
handles.slider15=get(handles.slider15,'Value'); %Lo cargamos en 'Value'
set(handles.text10,'String',handles.slider15); %Escribe el valor de Slider en
statictext

%% Imagen de la zona de SNAPSHOT
%Vualve a poner en la zona de captura la imagen original de la webcam
global original_imagen_cam
global imagen_cam
global new_imagen_cam

imagen_cam=original_imagen_cam;
new_imagen_cam=original_imagen_cam;
axes(handles.Zona_de_Captura);
imshow(imagen_cam);

set(handles.uitable1,'Visible','off')%Oculto visibilidad de la tabla
set(handles.text1,'String','SNAPSHOT'); % Cambio el texto
```

```
% --- Executes on slider movement.
function slider8_Callback(hObject, eventdata, handles)
% hObject    handle to slider8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider
%% Cargar el valor del Slider en la zona de visualizacion
handles.slider8=get(hObject,'Value'); %Carga en handles.slider1 el valor
delSlider
handles.slider8=round(handles.slider8);
set(handles.text5,'String',handles.slider8); %Escribe el valor de Slider en
statictext

%% Modificar el brillo en la imagen original y mostrala zona de captura
val=0.5*(get(hObject,'Value')-0.5);
global original_imagen_cam
global new_imagen_cam_bright

brt=original_imagen_cam;
new_imagen_cam_bright=brightness(brt,val);
axes(handles.Zona_de_Captura);
imshow(new_imagen_cam_bright);



% --- Executes on button press in imagen_negativo.
function imagen_negativo_Callback(hObject, eventdata, handles)
% hObject    handle to imagen_negativo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global new_imagen_cam_contrast
black=new_imagen_cam_contrast;
black=255-black;
axes(handles.Zona_de_Captura);
imshow(black);

%% Funcion para ajuste del brillo
% Con esta funcion modificamos el brillo de la imagen original y creamos
% una nueva imagen para proceder a la deteccion de las figuras.
% Lo que se quiere conseguir es una imagen mejoradas para no tener
% problemas a la hora de la deteccion.

function [ brt ] = brightness( im,val )

[a b c]=size(im);
for i=1:a
    for j=1:b
        for k=1:3
        im(i,j,k)=im(i,j,k)+val;
            if im(i,j,k)>255
            im(i,j,k)=255;
            elseif im(i,j,k)<0
            im(i,j,k)=0;
```

```
            end
        end
    end
end

brt=im;


% --- Executes on slider movement.
function slider10_Callback(hObject, eventdata, handles)
% hObject    handle to slider10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

%% Cargar el valor del Slider (contraste) en la zona de visualizacion
handles.slider10=get(hObject,'Value'); %Carga en handles.slider10 el valor del
Slider
set(handles.text7,'String',sprintf( '%1.2f' ,handles.slider10)); %Escribe el
valor de Slider en statictext

global new_imagen_cam_bright
global new_imagen_cam_contrast
val_contrast=get(hObject,'Value');
J = imadjust(new_imagen_cam_bright, [0 val_contrast], [0 1]);
axes(handles.Zona_de_Captura);
imshow(J);
new_imagen_cam_contrast = J;


% --- Executes on button press in imagen_gris.
function imagen_gris_Callback(hObject, eventdata, handles)
% hObject    handle to imagen_gris (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Convertir imagen en escala de grises

global new_imagen_cam_contrast
global img_g
img=new_imagen_cam_contrast;
img_g=rgb2gray(img);
axes(handles.Zona_de_Captura);
imshow(img_g);
set(handles.imagen_blanco_negro,'Enable','on'); %Habilito el boton B&W para
poder binarizar

% --- Executes on button press in imagen_blanco_negro.
function imagen_blanco_negro_Callback(hObject, eventdata, handles)
% hObject    handle to imagen_blanco_negro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Convertir imagen en blanco y negro (Binarizada)
global img_g
```

```
global threshold

%% Realzar los bordes
img_b=double(img_g)/255;     % Convierte a double
h=firpm(16, [0 .1 .3 1], [0 0 1 1]);  % Cálculo de un filtro equiripple paso
alto
h=ftrans2(h);      % Convierte en filtro 2D
imf=filter2(h,img_b);     % Filtrar la señal

%% Binarizar imagen
try
    umb=graythresh(img_b+imf);
    img_bw=im2bw((img_b+imf),(umb-threshold));
catch
    errordlg('AJUSTAR VALOR UMBRAL', 'Mensaje de error')
end


%% Eliminar ruido (Formas menores a 500pixel)
img_bn=bwareaopen(img_bw,500);


%% Mostrar imagen en zona SNAPSHOT
axes(handles.Zona_de_Captura);
imshow(img_bn);


% --- Executes on button press in comienzo_transferencia.
function comienzo_transferencia_Callback(hObject, eventdata, handles)
% hObject    handle to comienzo_transferencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global num_element
global n_amarillas
global n_rojas

Serial_PORT = serial('COM1');
set(Serial_PORT,'Terminator','CR/LF');
fopen(Serial_PORT);

%% Envio de datos de la TABLA a traves del puerto serie
numero_formas=n_amarillas+n_rojas;

for n=1:numero_formas
j = sprintf('$,%02i,%03i,%03i,%s,%s,%03i',cell2mat(num_element(n,1)), ...
                                          cell2mat(num_element(n,2)), ...
                                          cell2mat(num_element(n,3)), ...
                                          cell2mat(num_element(n,4)), ...
                                          cell2mat(num_element(n,5)), ...
                                          cell2mat(num_element(n,6)) ...
            );
disp(j)
fprintf(Serial_PORT, '%s', j);
fprintf(Serial_PORT,'\n');
pause(0.1);
end
```

```
helpdlg('TRANSFERENCIA TERMINADA',' CableBOT ');

% Cerramos y borramos el puerto serie para poder usarlo de nuevo
fclose(Serial_PORT);
delete(Serial_PORT);
clear Serial_PORT



% --- Executes on button press in transferir_datos.
function transferir_datos_Callback(hObject, eventdata, handles)
% hObject    handle to transferir_datos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Abrir y configurar puerto Serie COM1
%Secuencia de instrucciones para la configuracion del serial port
Serial_PORT = serial('COM1');
set(Serial_PORT,'Terminator','CR');
fopen(Serial_PORT);

%% Envio de datos a traves del puerto serie
fprintf(Serial_PORT,'%s\n','T');   %Envio 'T' por el serial

%
%% Recepcion por el puerto serie

%Se mantiene leyendo el puerto durante un determinado tiempo
A=fgets(Serial_PORT);
B = int32(str2num(A));
if B==1
set(handles.comienzo_transferencia,'enable','on');
helpdlg('Continuar con START',' CableBOT ');
else
    errordlg('NO HAY COMUNICAION SERIE', 'Mensaje de error')
    disp(A)
end
% Cerramos y borramos el puerto serie para poder usarlo de nuevo
fclose(Serial_PORT);
delete(Serial_PORT);
clear Serial_PORT



    % --- Executes on slider movement.
function slider15_Callback(hObject, eventdata, handles)
% hObject    handle to slider15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

%% Cargar el valor del Slider (umbral) en la zona de visualizacion
handles.slider15=get(hObject,'Value'); %Carga en handles.slider15 el valor del
Slider
set(handles.text10,'String',sprintf( '%1.2f' ,handles.slider15)); %Escribe el
valor de Slider en statictext
```

```
global threshold
threshold=get(hObject,'Value'); %Carga en threshold el valor del Slider


% --- Executes on button press in calibracion_distancia.
function calibracion_distancia_Callback(hObject, eventdata, handles)
% hObject    handle to calibracion_distancia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global relacion
global azimutal_CAM


opc=questdlg('¿Que deseas hacer. Escalado o Centrado?',...
    'CALIBRATION', 'Scale','Centered','Centered');
if strcmp(opc,'Centered')
    %% Camara azimutal
    try
    vid=videoinput('winvideo',1,'RGB24_800x600'); % esto crea el canal de
video a la camara web
    catch
    errordlg('CAMARA AZIMUTAL NO CONECTADA', 'Mensaje de error')
    end

%% Parte del centrado

% Camara panoramica con seguimiento del efector
% Tomo las caracteristicas que la camara tiene por defecto
% Configuracion de las propiedades del objeto
VidSrc = getselectedsource(vid)
%set(vid, 'FocusMode', 'auto');
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 1;
flushdata(vid);
x=1;
while(x)
for n=1:3
%Comienzo de la adquisicion
start(vid);
%Bucle de toma de imagenes
while(vid.FramesAcquired<=200)

    % Tomamos una foto de la camara
    data = getsnapshot(vid);

    %Para el tracking sacamos la componente azul de cada imagen y la
    %tratamos de forma que identificamos solamente las formas de color azul
    diff_im = imsubtract(data(:,:,3), rgb2gray(data));
    %Usamos filtro de media para reducir el ruido
    diff_im = medfilt2(diff_im, [3 3]);
    %Binarizamos imagen
    diff_im = im2bw(diff_im,0.18);
    %Fuera ruido menor a 500 pixels
    diff_im = bwareaopen(diff_im,500);
```

```
    %Etiquetamos todos los componentes de la imagen
    bw = bwlabel(diff_im, 8);

    %Tomamos la informacion de los objetos detectados
    stats = regionprops(bw, 'BoundingBox', 'Centroid');

%% Zona de visualizacion de la imagen capturada
set(handles.text1,'String','CENTERED'); % Cambio el texto de la zona de
visualización
%Creamos la zona en la cual se mostrará la imagen capturada
    axes(handles.Zona_de_Captura); % zona de captura
    background = data; % cargamos la imagen capturada
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada
    hold on

    %Marcamos los objetos azules con un cuadro
    for object = 1:length(stats)
        bb = stats(object).BoundingBox;
        bc = stats(object).Centroid;
        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
        plot(bc(1),bc(2), '-m+')
        a=text(bc(1)+35,bc(2)-20, strcat('X: ', num2str(round(bc(1))), '    Y:
', num2str(round(bc(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'r');
        % Lineas de centrado en la imagen
        line([0,800], [300,300],'color', 'c','LineWidth',2);%Linea
longitudinal
        line([400,400], [0,600],'color', 'c','LineWidth',2);%Linea transversal
    end

    hold off

end

% Stop de la adquisicion de video.
stop(vid);
%Limpiamos el buffer de la memoria de las instantaneas tomadas para volver
%a iniciar el ciclo
flushdata(vid);

%% Mensaje de continuar o no
    if(n==2)
        opc=questdlg('¿Desea continuar?',...
        'CONTINUAR', 'SI','NO','SI');
        if strcmp(opc,'SI')
        x=1;
        end
        if strcmp(opc,'NO')
        x=0;
        end
    end
end
end
%Limpiamos el buffer de la memoria de las instantaneas tomadas para salir
flushdata(vid);
disp('Final del proceso')
```

```
%% Cerrar videos
closepreview(vid); % cerramos la ventana de preview
delete(vid); % borramos el canal de Video
helpdlg('Fin de video',' CableBOT ');
uiwait
end
%% Parte de la calibracion
try
    canalVideo=videoinput('winvideo',azimutal_CAM,'RGB24_800x600'); % esto
crea el canal de video a la camara web
    preview(canalVideo); % abrimos una ventana de preview para ver a donde
apuntamos con la camara
catch
    errordlg('NO HAY CAMARA CONECTADA', 'Mensaje de error')
end

start(canalVideo); % inicializamos el canal de Video
pause(5); % espera 5 segundos para que estabilice la imagen de la webcam
img_calibracion=getsnapshot(canalVideo); % tomamos una instantanea con la
camara que se guarda en img_calibracion

%Guardamos la imagen capturada en el PC
imwrite(img_calibracion,'.\Imagenes CAM\Calibration_img.jpg','jpg');

closepreview(canalVideo); % cerramos la ventana de preview
delete(canalVideo); % borramos el canal de Video
[imagen_calibrada x y]=calibration(img_calibracion);
%
set(handles.text1,'String','CALIBRATION'); % Cambio el texto de la imagen de
ZONA DE CAPTURA
axes(handles.Zona_de_Captura); % zona de captura
background = imagen_calibrada; % cargamos la imagen capturada
axis off; % quitamos los ejes de la zona de captura (axes)
imshow(background) % mostramos la imagen en la zona capturada

%
hold on
%line([xini, xend], [yini,yend] ayuda para comando line
line([x(1), x(1)], [y(1),y(2)],'color', 'k','LineWidth',2);%cateto opuesto
line([x(1), x(2)], [y(2),y(2)],'color', 'g','LineWidth',2);%cateto adyacente
line([x(1), x(2)], [y(1),y(2)],'color', 'r','LineWidth',2);%hipotenusa

global scale
global hipot

cat_op=(y(2)-y(1));
cat_ad=(x(2)-x(1));
hipot=sqrt((cat_op*cat_op)+(cat_ad*cat_ad));

% Llamada a la GUIDE de calibracion
Scale;
uiwait
relacion=hipot/scale;

cat_op=num2str(cat_op);
cat_ad=num2str(cat_ad);
hipot=num2str(hipot);
relacion_text=num2str(relacion);
```

```
text(20,70,['A=',cat_op],'Color','k'); %visualiza linea cat_op
text(20,120,['B=',cat_ad],'Color','g'); %visualiza linea cat_ad
text(20,170,['C=',hipot],'Color','r'); %visualiza linea hipot

rectangle('Position',[10,50,150,150],'EdgeColor','k','LineWidth',2)
line([20,500], [30,30],'color', 'k','LineWidth',1);%hipotenusa

text(20,20,['La relacion pixel - mm es: ', relacion_text ,' pixeles es 1mm'
]); %visualiza el punto C


% --- Executes on button press in control_gamepad.
function control_gamepad_Callback(hObject, eventdata, handles)
% hObject    handle to control_gamepad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
gamepad_control;
warndlg('Has salido del control GAMEPAD','CableBOT');

% --- Executes on button press in motor_control.
function motor_control_Callback(hObject, eventdata, handles)
%% Llamada a la GUIDE de control del motor
Motor_Control;
```

3.1.3. Calibration.m

```
function [imagen_tratada, centro_X, centro_Y] = calibration(imagen)
%
%
%Esta funcion se encarga de la calibracion de la camara.
%convirtiendo la distancia de pixeles en mm. Se le ha dotado de un boton
%para realizar el centrado de la zona de trabajo respecto a la posicion de
%la camara.

% PARÁMETROS DE ENTRADA
%imagen = Ultima imagen arreglada para la deteccion

% PARÁMETROS DE SALIDA
%centro_X = centro de masa de cada forma del patron.
%centro_Y = centro de masa de cada forma del patron.
%imagen_tratada = Imagen arreglada para la deteccion del escalado


img=imagen; % Cargo la imagen traida desde la GUIDE
%% Pasar a escala de grises
img_g=rgb2gray(img);

%% Realzar los bordes
img_b=double(img_g)/255;     % Convierte a double
h=firpm(16, [0 .1 .3 1], [0 0 1 1]);  % Cálculo de un filtro equiripple paso
alto
h=ftrans2(h);      % Convierte en filtro 2D
imf=filter2(h,img_b);     % Filtrar la señal
```

```
%% Binarizar imagen
umb=graythresh((img_b+imf)+0.2);
img_bw=im2bw((img_b+imf),umb);

%% Eliminar ruido
img_bn=bwareaopen(img_bw,500);

%% Rellenar huecos vacios
se=strel('disk',15);
img_br=imclose(img_bn,se);

%% Etiquetar elementos
[L Ne]=bwlabel(img_br);

%% Encontrar propiedades de los elementos
prop=regionprops(L,'all');
%% Delimitar formas
for n=1:length(prop)
   % rectangle('Position',prop(n).BoundingBox,'EdgeColor','r','LineWidth',2)
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    centro_X(n)=x;
    centro_Y(n)=y;
end
hold off

imagen_tratada=label2rgb(L);

end
```

3.1.4. Motor_Control.m

```
function varargout = Motor_Control(varargin)

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Motor_Control

% Last Modified by GUIDE v2.5 25-Apr-2014 00:30:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Motor_Control_OpeningFcn, ...
                   'gui_OutputFcn',  @Motor_Control_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Motor_Control is made visible.
function Motor_Control_OpeningFcn(hObject, eventdata, handles, varargin)
%% Centrar ventana del GUIDE
%Con estas instrucciones realizamos un centrado de la pantalla de la GUIDE
%de forma que tenga una mejor vision de cara al usuario
scrsz=get(0,'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3)-pos_act(3);
xp=round(xr/2);
yr=scrsz(4)-pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

%% Insertar Logo de la UVic
%Con estas instrucciones leo la imagen del logo y la represento en el axes
%correspondiente
%logo=imread('C:\Documents and
Settings\LOPEZ\Escritorio\GRADO\PFG\CableBOT\GUI MatLab\Logos\logoHome.jpg');
logo=imread('.\Logos\logoHome.jpg');
image(logo)
axis off

%% Personalizar boton SALIR
%Insertar imagen en el boton cerrar de forma que vemos el simbolo salir en
%el boton
[a,map]=imread('.\Logos\cerrar.jpg');
[r,c,d]=size(a);
x=ceil(r/50);
y=ceil(c/50);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.1*255;
set(handles.Salir,'CData',g);


%% Carta de ajuste en axes
% Comienzo de lops axes con carta de ajuste en pantalla

    axes(handles.video_1); % zona de captura
    background = imread('.\Logos\cartadeajuste.jpg'); % cargamos la imagen
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada

    axes(handles.video_2); % zona de captura
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada


% Choose default command line output for Motor_Control
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes Motor_Control wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Motor_Control_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


%% Recuperacion de variables
global panoramica_CAM
global azimutal_CAM

%% Camara panoramica
    try
        vid=videoinput('winvideo',panoramica_CAM,'RGB24_640x480'); % esto crea
el canal de video a la camara web
    catch
        errordlg('CAMARA PANORAMICA NO CONECTADA', 'Mensaje de error')
    end

 %% Camara azimutal
    try
    vid_1=videoinput('winvideo',azimutal_CAM,'RGB24_800x600'); % esto crea el
canal de video a la camara web
    catch
    errordlg('CAMARA AZIMUTAL NO CONECTADA', 'Mensaje de error')
    end

vid_1.TriggerFrameDelay = 25;
vid_1.FramesPerTrigger = 1;
vidRes = get(vid_1, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid_1, 'NumberOfBands');
hImage = image( zeros(imHeight, imWidth, nBands),'Parent',handles.video_2 );
preview(vid_1,hImage);
start(vid_1);


%% Camara panoramica con seguimiento del efector
% Tomo las caracteristicas que la camara tiene por defecto
%VidSrc = getselectedsource(vid);
% Configuracion de las propiedades del objeto
VidSrc = getselectedsource(vid)
%set(vid, 'FocusMode', 'auto');
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 1;
flushdata(vid);
x=1;
while(x)
for n=1:3
%Comienzo de la adquisicion
```

```
start(vid);
%Bucle de toma de imagenes
while(vid.FramesAcquired<=200)

    % Tomamos una foto de la camara
    data = getsnapshot(vid);

    %Para el tracking sacamos la componente azul de cada imagen y la
    %tratamos de forma que identificamos solamente las formas de color azul
    diff_im = imsubtract(data(:,:,3), rgb2gray(data));
    %Usamos filtro de media para reducir el ruido
    diff_im = medfilt2(diff_im, [3 3]);
    %Binarizamos imagen
    diff_im = im2bw(diff_im,0.18);
    %Fuera ruido menor a 500 pixels
    diff_im = bwareaopen(diff_im,500);
    %Etiquetamos todos los componentes de la imagen
    bw = bwlabel(diff_im, 8);

    %Tomamos la informacion de los objetos detectados
    stats = regionprops(bw, 'BoundingBox', 'Centroid');

%% Zona de visualizacion de la imagen capturada
%Creamos la zona en la cual se mostrará la imagen capturada
    axes(handles.video_1); % zona de captura
    background = data; % cargamos la imagen capturada
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada
    hold on

    %Marcamos los objetos azules con un cuadro
    for object = 1:length(stats)
        bb = stats(object).BoundingBox;
        bc = stats(object).Centroid;
        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
        plot(bc(1),bc(2), '-m+')
        a=text(bc(1)+15,bc(2), strcat('X: ', num2str(round(bc(1))), '    Y: ',
num2str(round(bc(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'magenta');
    end

    hold off

end

% Stop de la adquisicion de video.
stop(vid);
%Limpiamos el buffer de la memoria de las instantaneas tomadas para volver
%a iniciar el ciclo
flushdata(vid);
sprintf('%s %i','El numero de ciclo es:', n)

%% Mensaje de continuar o no
    if(n==2)
        opc=questdlg('¿Desea continuar?',...
        'CONTINUAR', 'SI','NO','SI');
        if strcmp(opc,'SI')
```

```
        x=1;
        sprintf('%s %i','valor de:', x)
        end
        if strcmp(opc,'NO')
        x=0;
        sprintf('%s %i','valor de:', x)
        end
    end
end
end
%Limpiamos el buffer de la memoria de las instantaneas tomadas para salir
flushdata(vid);
disp('Final del proceso')
%% Cerrar videos
closepreview(vid_1); % cerramos la ventana de preview
delete(vid_1); % borramos el canal de Video
helpdlg('Fin de video',' CableBOT ');

    % --- Executes on button press in cerrar_video.
function cerrar_video_Callback(hObject, eventdata, handles)
%% Carta de ajuste en axes
% Como el proceso ha finalizado colocamos una carta de ajuste en pantalla

%% Zona de visualizacion de la imagen capturada
%Creamos la zona en la cual se mostrará la imagen capturada
    axes(handles.video_1); % zona de captura
    background = imread('.\Logos\cartadeajuste.jpg'); % cargamos la imagen
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada

    axes(handles.video_2); % zona de captura
    axis off; % quitamos los ejes de la zona de captura (axes)
    imshow(background) % mostramos la imagen en la zona capturada


% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
%% Mensaje que se muestra al presionar el boton inferior de SALIR
opc=questdlg('¿Desea salir del programa?',...
    'SALIR', 'SI','NO','NO');
if strcmp(opc,'NO')
    return;
end

delete(handles.figure1);
```

3.1.5. Gamepad_Control.m

```
function [] = gamepad_control ()

%% Control mediante GamePad
%
%Esta funcion toma el control del robot mediante el GamePad asociado al PC
%Para salir basta con pulsar el boton 10
%

clc;
    try
    JOY = VRJOYSTICK(1);
    catch
    errordlg('NO HAY JOYSTICK CONECTADO', 'Mensaje de error')
    uiwait
    end

stop_button=10;
buttons(stop_button)=0;
disp('CONTROL GAMEPAD')
warndlg('Has tomado el control GAMEPAD','CableBOT');
uiwait

%% Abrir y configurar puerto Serie COM1
%Secuencia de instrucciones para la configuracion del serial port
Serial_PORT = serial('COM1');
fclose(Serial_PORT);
set(Serial_PORT,'Baudrate',9600);
set(Serial_PORT,'Terminator','CR/LF');
set(Serial_PORT,'DataBits',8);
set(Serial_PORT,'Parity','none');
set(Serial_PORT,'StopBits',1);
set(Serial_PORT,'FlowControl','none');
fopen(Serial_PORT);

%% Bucle central
while buttons(10)~=1

    [axes, buttons, povs]=read(JOY);


%% Flancos de subida (Ejemplo)

% %Flanco boton 1
%     actual_1=buttons(1);
%     if actual_1==1 && anterior_1==0
%         disp('Boton 1');
%     end
%     anterior_1=actual_1;
%



%% Combinacion de 2 botones hacia adelante
% Con esta combinacion movemos los motores de forma individual adelante
 %Flanco boton 1-6. Motor 1 adelante
    actual_1_6=buttons(1)& buttons(6);
```

```
     if actual_1_6==1 && anterior_1_6 ==0
         fprintf(Serial_PORT,'%s\n','1');   %Envio '1' por el serial
         disp('Motor 1 (adelante)');
     end
     anterior_1_6=actual_1_6;

%Flanco boton 2-6. Motor 2 adelante
    actual_2_6=buttons(2)& buttons(6);
     if actual_2_6==1 && anterior_2_6 ==0
         fprintf(Serial_PORT,'%s\n','3');   %Envio '3' por el serial
         disp('Motor 2 (adelante)');
     end
     anterior_2_6=actual_2_6;

%Flanco boton 3-6. Motor 3 adelante
    actual_3_6=buttons(3)& buttons(6);
     if actual_3_6==1 && anterior_3_6 ==0
         fprintf(Serial_PORT,'%s\n','5');   %Envio '5' por el serial
         disp('Motor 3 (adelante)');
     end
     anterior_3_6=actual_3_6;

%Flanco boton 4-6. Motor 4 adelante
    actual_4_6=buttons(4)& buttons(6);
     if actual_4_6==1 && anterior_4_6 ==0
         fprintf(Serial_PORT,'%s\n','7');   %Envio '7' por el serial
         disp('Motor 4 (adelante)');
     end
     anterior_4_6=actual_4_6;


%% Combinacion de 2 botones hacia atras
% Con esta combinacion movemos los motores de forma individual hacia atras

%Flanco boton 1-8. Motor 1 atras
    actual_1_8=buttons(1)& buttons(8);
     if actual_1_8==1 && anterior_1_8 ==0
         fprintf(Serial_PORT,'%s\n','2');   %Envio '2' por el serial
         disp('Motor 1 (atras)');
     end
     anterior_1_8=actual_1_8;

%Flanco boton 2-8. Motor 2 atras
    actual_2_8=buttons(2)& buttons(8);
     if actual_2_8==1 && anterior_2_8 ==0
         fprintf(Serial_PORT,'%s\n','4');   %Envio '4' por el serial
         disp('Motor 2 (atras)');
     end
     anterior_2_8=actual_2_8;

%Flanco boton 3-8. Motor 3 atras
    actual_3_8=buttons(3)& buttons(8);
     if actual_3_8==1 && anterior_3_8 ==0
         fprintf(Serial_PORT,'%s\n','6');   %Envio '6' por el serial
         disp('Motor 3 (atras)');
     end
     anterior_3_8=actual_3_8;
```

```
%Flanco boton 4-8. Motor 4 atras
    actual_4_8=buttons(4)& buttons(8);
     if actual_4_8==1 && anterior_4_8 ==0
         fprintf(Serial_PORT,'%s\n','8');   %Envio '8' por el serial
         disp('Motor 4 (atras)');
     end
     anterior_4_8=actual_4_8;

%% Control mediante los botones en cruz
% Con estos botones movemos el robot en la direccion seleccionada

%Flanco robot adelante (Flechas)
    actual_povs_0=povs;
    if actual_povs_0==0 && anterior_povs_0~=0
        fprintf(Serial_PORT,'%s\n','E');   %Envio 'E' por el serial
        disp('ROBOT ADELANTE');
    end
    anterior_povs_0=actual_povs_0;


%Flanco robot derecha (Flechas)
    actual_povs_90=povs;
    if actual_povs_90==90 && anterior_povs_90~=90
        fprintf(Serial_PORT,'%s\n','H');   %Envio 'H' por el serial
        disp('ROBOT DERECHA');
    end
    anterior_povs_90=actual_povs_90;


%Flanco robot atras (Flechas)
    actual_povs_180=povs;
    if actual_povs_180==180 && anterior_povs_180~=180
        fprintf(Serial_PORT,'%s\n','F');   %Envio 'F' por el serial
        disp('ROBOT ATRAS');
    end
    anterior_povs_180=actual_povs_180;


%Flanco robot izquierda (Flechas)
    actual_povs_270=povs;
    if actual_povs_270==270 && anterior_povs_270~=270
        fprintf(Serial_PORT,'%s\n','G');   %Envio 'G' por el serial
        disp('ROBOT IZQUIERDA');
    end
    anterior_povs_270=actual_povs_270;


end
fclose(Serial_PORT);
delete(Serial_PORT);
clear Serial_PORT
end
```

3.1.6. Scale.m

```
function varargout = Scale(varargin)
% SCALE M-file for Scale.fig
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Scale

% Last Modified by GUIDE v2.5 18-Aug-2014 10:48:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Scale_OpeningFcn, ...
                   'gui_OutputFcn',  @Scale_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Scale is made visible.
function Scale_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Scale (see VARARGIN)

% Choose default command line output for Scale
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
global hipot

set(handles.edit2,'String',hipot);

% UIWAIT makes Scale wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Scale_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;




% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global scale
Val=get(handles.edit1,'String'); %Almacenar valor ingresado
scale = str2double(Val);  %Transformar a formato double
delete(handles.figure1);


% --- Executes on button press in cancelar.
function cancelar_Callback(hObject, eventdata, handles)
% hObject    handle to cancelar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
centered;
```

3.1.7. Color.m

```
function [imagen_discriminada] = color(imagen,RGB,valor_umbral)
%
% Esta función se encarga de evaluar el color de cada pieza para
% discriminar de qué color se trata. Con esto conseguimos diferenciar el
% número de piezas de cada color que existen en la base de trabajo

% PARÁMETROS DE ENTRADA
%imagen = Ultima imagen arreglada con el panel de herramientas de imagen

% PARÁMETROS DE SALIDA
%imagen_discriminada = imagen en la cual solo estan presentes las formas
%del color buscado

I=imagen;
% Separar la matriz RGB en R(rojo), B (azul) y G (verde)
R=RGB(1);
G=RGB(2);
B=RGB(3);
```

```
% Cálculo de la referenica
% Realizar una comparación de cada pixel de la imagen de entrada con los
% valores de la matriz RGB (esta diferencia debe estar dentro del umbral para
encontrar el color)
imagen_discriminada=(abs(I(:,:,1)-R)<valor_umbral)&(abs(I(:,:,2)-
G)<valor_umbral)&(abs(I(:,:,3)-B)<valor_umbral);
% En otras palabras, si el color que se busca, por ejemplo es de tendencia
% azul ([0 0 255]), la matriz "imagen_discriminada" será uno (1 - pixel
blanco)
% solo en aquellos sectores de la imagen donde la diferencia esté dentro
% del umbral (o sea, donde esté el color más cercano al azul).


end
```

3.1.8. Place.m

```
function [centro N] = place(imagen_tratada , imagen_original , color)
% Esta función separa cada forma de la imagen de entrada. Asimismo, obtiene
% el centro de masa de cada objeto.
% Devuelve también en número de elementos que hay en la imagen

% PARÁMETROS DE ENTRADA
%imagen = Ultima imagen arreglada para la deteccion de las formas
%color = Tag que nos indica que imagen vamos a tratar, amarillas o rojas

% PARÁMETROS DE SALIDA
%centro = centro de masa de cada forma (objeto).
%N = número de formas dentro de la imagen.


%% Inicio
img_o=imagen_original;
imagen=imagen_tratada;
if color==1
    img=not(imagen);
    imagen_amarillas = bwareaopen(img,100);
    figure,imshow(imagen_amarillas,'InitialMagnification',200);
    title('Piezas color AMARILLO');
    helpdlg('Seguir proceso pulsando tecla',' CableBOT ');
else
    img=imagen;
    figure,imshow(img,'InitialMagnification',200);
    title('Piezas color ROJO');
    %helpdlg('Seguir proceso pulsando tecla',' CableBOT ');
end
pause(); %Espera a pulsar tecla
% %% Pasar a escala de grises
% img_g=rgb2gray(img);
% %subplot(3,3,3),imshow(img_g);
% figure,imshow(img_g);
% title('Grises');
% pause();%Espera a pulsar tecla
% %% Realzar los bordes
% img_b=double(img_g)/255;     % Convierte a double
```

```
% h=firpm(16, [0 .1 .3 1], [0 0 1 1]);  % Cálculo de un filtro equiripple paso
alto
% h=ftrans2(h);      % Convierte en filtro 2D
% imf=filter2(h,img_b);     % Filtrar la señal
% % subplot(3,3,4), imshow(img_b+imf);
% figure, imshow(img_b+imf);
% title('Resalte bordes');
% pause();%Espera a pulsar tecla
% %% Binarizar imagen
% umb=graythresh(img_b+imf);
% img_bw=im2bw((img_b+imf),(umb-umbral));
% %subplot(3,3,5),imshow(img_bw);
% figure,imshow(img_bw);
% title('Binarizada');
% pause();%Espera a pulsar tecla
%% Eliminar ruido
img_bn=bwareaopen(img,700);
% figure,imshow(img_bn);
% %subplot(3,3,6),imshow(img_bn);
% title('Sin ruido');
% pause();%Espera a pulsar tecla
%% Rellenar huecos vacios
se=strel('square',5);
img_br=imclose(img_bn,se);
se= strel('disk',5);
img_br1=imclose(img_br,se);
% %subplot(3,3,7),imshow(img_br);
figure,imshow(img_br1);
title('Rellenada');
pause();%Espera a pulsar tecla
%% Etiquetar elementos
[L N]=bwlabel(img_br1);
% figure,imshow(label2rgb(L));
%subplot(3,3,8),imshow(label2rgb(L,'jet'));
%title('Etiquetada');
figure,imshow(label2rgb(L));
title('Etiquetada');
pause();%Espera a pulsar tecla
%% Si no existe ningún elemento (forma), retorna al programa principal
if N==0
    centro=[ ];
    N=0;
    opc=questdlg('No hay piezas rojas. ¿Desea continuar?',...
    'CONTINUE', 'SI','NO','SI');
    if strcmp(opc,'SI')
    close figure 3;
    close figure 2;
    close figure 1;
    return
    end
end


%% Encontrar propiedades de los elementos
prop=regionprops(L,'all');

% Contar el número de objetos
np=length(prop);
% Matriz vacía para ir concatenando los centros de las figuras
```

```
centro=[ ];
figure,imshow(img_o);
title('Deteccion de centros');
%% Delimitar formas
hold on
for n=1:np
    rectangle('Position',prop(n).BoundingBox,'EdgeColor','r','LineWidth',2)
    x=prop(n).Centroid(1);
    y=prop(n).Centroid(2);
    centro=[centro;prop(n).Centroid];% Guardar el centro del objeto
    plot(x,y,'+')
end
hold off


pause();%Espera a pulsar tecla

%% Mensaje de aviso para continuar en la GUIDE
opc=questdlg('Deteccion Finalizada',...
    'CONTINUAR', 'SI','NO','SI');
if strcmp(opc,'SI')
   close figure 4;
   close figure 3;
   close figure 2;
   close figure 1;

else
    return
end


end
```

3.1.9. Shape.m

```
function [prop] = shape(imagen_tratada , imagen_original , umbral)

% Esta función se encarga de detectar la forma de cada pieza para cargarla
% en la tabla de visualizacion. Nos dice si se trata de un cuadrado,
% circulo o triangulo.
% Separa cada forma de la imagen de entrada delimitando su perimetro para
% luego compararlas con unas plantillas creadas previamente

% PARÁMETROS DE ENTRADA
%imagen = Ultima imagen arreglada para la deteccion de las formas
%umbral = Valor del threshold hayyado en el panel de herramientas para
%binarizar sin tener problemas

% PARÁMETROS DE SALIDA
%forma = Vector con la forma de cada figura

img_o=imagen_original;
%% Muestra la imagen inicial
img=imagen_tratada;
```

```
figure,imshow(img_o,'InitialMagnification',200);
title('Original');
helpdlg('Seguir proceso pulsando tecla',' CableBOT ');
pause(); %Espera a pulsar tecla
%% Pasar a escala de grises
img_g=rgb2gray(img);
%subplot(3,3,3),imshow(img_g);
figure,imshow(img_g);
title('Grises');
pause();%Espera a pulsar tecla

%% Realzar los bordes
img_b=double(img_g)/255;     % Convierte a double
h=firpm(80, [0 .1 .3 1], [1 1 1 1]);  % Cálculo de un filtro equiripple paso
alto
h=ftrans2(h);      % Convierte en filtro 2D
imf=filter2(h,img_b);     % Filtrar la señal
% subplot(3,3,4), imshow(img_b+imf);
figure, imshow(img_b+imf);
title('Resalte bordes');
pause();%Espera a pulsar tecla

%% Binarizar imagen
try
    umb=graythresh(img_b+imf);
    img_bw=im2bw((img_b+imf),(umb-umbral));
catch
    errordlg('AJUSTAR VALOR UMBRAL', 'Mensaje de error')
end
%subplot(3,3,5),imshow(img_bw);
figure,imshow(img_bw);
title('Binarizada');
pause();%Espera a pulsar tecla

%% Eliminar ruido
img_bn=bwareaopen(img_bw,500);
figure,imshow(img_bn);
%subplot(3,3,6),imshow(img_bn);
title('Sin ruido');
pause();%Espera a pulsar tecla

%% Rellenar huecos vacios
se=strel('square',5);
img_br=imclose(img_bn,se);
%subplot(3,3,7),imshow(img_br);
figure,imshow(img_br);
title('Rellenada');
pause();%Espera a pulsar tecla

%% Etiquetar elementos
[L N]=bwlabel(img_br);

%% Si no existe ningún elemento (forma), retorna al programa principal
if N==0
    errordlg('NO EXISTE NINGUNA FIGURA', 'Mensaje de error')
%     close figure 3;
%     close figure 2;
%     close figure 1;
```

```
    return
end

%% Encontrar propiedades de los elementos
prop=regionprops(L,'all');

% Contar el número de objetos
np=length(prop);

%% Marcar perimetro
%figure,imshow(img_br);
figure,imshow(img_o);
title('Perimetrada');
hold on
B=bwboundaries(img_br);
for k = 1:np
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'c', 'LineWidth', 2)
    %disp(prop(k).Area);
end
hold off
%pause();%Espera a pulsar tecla

%% Mensaje de aviso para continuar en la GUIDE
opc=questdlg('Deteccion Finalizada',...
    'CONTINUAR', 'SI','NO','SI');
if strcmp(opc,'SI')
   close figure 7;
   close figure 6;
   close figure 5;
   close figure 4;
   close figure 3;
   close figure 2;
   close figure 1;

else
    return
end
```

3.2. Código firmware de control

Este capítulo queda pendiente al no conseguir llegar a tiempo a realizar una programación digna de ser comentada.
Al tratarse de un proyecto con carácter personal, es decir, que debido al tema tratado es de alto interés para mí, esta parte se va realizar con posterioridad a la defensa del trabajo dando al robot las funcionalidades esperadas.

Ya se han realizado pruebas de programación en la placa de desarrollo elegida para ello por lo tanto solo es cuestión de tiempo conseguir un firmware de control.