

Proyecto: Balancín 2D

Técnicas de Identificación de Sistemas Dinámicos

Nombre

8170

IECSA09

Universidad Aeronáutica en Querétaro
Colón, Querétaro
08 de Mayo, 2023

Índice

Índice de figuras	2
1. Objetivo	3
2. Introducción	3
3. Marco Teórico	3
3.1. Técnicas de identificación	3
3.2. Control	3
3.2.1. Control Proporcional	3
3.2.2. Control Derivativo	3
3.2.3. Control Integral	4
3.3. Modelo en espacio de estados	4
3.4. Modelo de perturbación, ruido y sistema	4
3.5. OpenCV	4
3.6. MatLab	4
3.6.1. Simulink	4
4. Desarrollo	5
4.1. Modelo matemático	5
4.2. Diagrama de cuerpo libre	5
4.3. Materiales	5
4.4. Maqueta	6
4.5. Adquisición de señales	7
4.6. Pre-Control	7
4.6.1. Identificación del sistema	7
4.7. Control	8
5. Resultados	9
6. Conclusiones	9
6.1. Trabajos futuros	10
7. Anexos	11
7.1. Código OpenCV	11
7.2. Código Mínimos Cuadrados	18
7.3. Código Code Composer	19
Bibliografía	26

Índice de figuras

1.	Logo Open CV	4
2.	Logo MatLab	4
3.	Ejemplo de diagrama de bloques en Simulink	5
4.	Diagrama de cuerpo libre para ambos ejes	5
5.	Cámara	6
6.	Servo	6
7.	Maqueta Completa	6
8.	Posición de la cámara	7
9.	Base de la maqueta	7
10.	Ejemplo de prueba de posición	7
11.	Prueba de la posición en x	8
12.	Prueba de la posición en y	8
13.	Aplicación de mínimos cuadrados en X	8
14.	Función de transferencia aproximada del eje X	8
15.	Aplicación de mínimos cuadrados en Y	8
16.	Función de transferencia aproximada del eje Y	8
17.	Sistema con función de transferencia de X en Simulink	8
18.	Sintonizador PID para eje X	9
19.	Parámetros obtenidos con autotuner en el eje X	9
20.	Sistema con función de transferencia de Y en Simulink	9
21.	Sintonizador PID para eje Y	9
22.	Parámetros obtenidos con autotuner en el eje Y	9
23.	Comparación de datos en X	9

1. Objetivo

Se hará un balancín de dos ejes que controle la posición de una pelota sobre una superficie cuadrada plana. Con ayuda de un sistema de control PID se moverán dos servos ubicados en cada uno de los ejes de la superficie y se sensará la posición de la pelota con una cámara colocada de manera que pueda ver la superficie completa para posicionar la pelota en la posición deseada por el usuario.

2. Introducción

El balancín 2D o también llamado *Ball Balancing System*, es un sistema de control en el que tenemos que estabilizar un sistema inestable que consiste en una pelota sobre una superficie controlada por dos ejes x y y .

Este proyecto es de mucha utilidad para entender sistemas de control, pues es necesario tener un sistema PID bien sintonizado para que este pueda balancear de manera efectiva la pelota. También servirá como introducción a los algoritmos de procesamiento de imagen por el uso de la cámara para la detección de la pelota.

3. Marco Teórico

3.1. Técnicas de identificación

Son los estudios de técnicas que persiguen la obtención de modelos matemáticos de sistemas dinámicos a partir de mediciones realizadas en el proceso: entradas, salidas y perturbaciones.

El enfoque de la identificación se puede realizar en función de la estructura del modelo

- Black-box: Los parámetros del modelo no tienen una interpretación física.
- Gray-box: Algunas partes del sistema son modeladas basándose en principios fundamentales. Solo algunos parámetros pueden tener interpretación física.
- White-box: La estructura del modelo se obtiene a partir de leyes fundamentales. Todos los parámetros tienen interpretación física.

3.2. Control

Un controlador es un dispositivo que permite controlar un sistema en lazo cerrado para que alcance un estado deseado. Está compuesto por tres elementos que proporcionan una acción proporcional, integral y derivativa.

Para que el sistema se estabilice necesitaremos una señal de referencia y una señal de error. La señal de referencia indica el estado que se desea conseguir y la señal de error indicará la diferencia que existe entre la referencia y el estado real del sistema.

3.2.1. Control Proporcional

La acción proporcional multiplica la señal de error por una constante K_p que determina la cantidad de acción proporcional que tendrá el controlador.

Modificar la ganancia proporcional tiene los siguientes efectos:

- Aumenta la velocidad de respuesta del sistema
- Disminuye el error del sistema
- Aumenta la inestabilidad del sistema

3.2.2. Control Derivativo

La acción derivativa será la encargada de ajustar la velocidad a la que el sistema se acerca al error. Cuando el sistema se mueve a altas velocidades, el sistema se puede pasarse por la inercia, para controlar esto, el control derivativo debe reconocer la velocidad y frenarla antes de que se acerque a la referencia deseada.

Aumentar la ganancia derivativa tiene los siguientes efectos:

- Aumenta la estabilidad del sistema controlado
- Disminuye la velocidad del sistema

Un problema que presenta el control derivativo es que amplifica las señales que varían rápidamente, como el ruido. Es deseable aplicar filtros para que esto no pase y el sistema se estabilice con éxito.

3.2.3. Control Integral

La acción integral es la encargada de acumular la señal de error, con esto se consigue reducir el error del sistema en el régimen permanente. La desventaja es que hace más inestable al sistema.

Aumentar la ganancia integral tiene los siguientes efectos:

- Disminuye el error en régimen permanente del sistema
- Aumenta la inestabilidad del sistema
- Aumenta un poco la velocidad del sistema

3.3. Modelo en espacio de estados

Estos modelos son modelos que usan variables para describir un sistema por un conjunto de ecuaciones diferenciales de primer orden.

Este tipo de modelos son una buena elección para estimaciones rápidas porque solo requiere que un parámetro sea especificado, *el orden n del modelo*. El orden del modelo es un entero igual a la dimensión de $x(t)$ y se relaciona con el número de entradas y salidas retardadas utilizadas en la ecuación de diferencia correspondiente.

3.4. Modelo de perturbación, ruido y sistema

El modelo de una planta relaciona la entrada con la salida y es útil para el diseño del controlador, sensor y el monitoreo del desempeño. La entrada del sistema, la perturbación y el ruido son señales deterministas o aleatorias. La perturbación representa una entrada que afecta el desempeño del sistema pero no puede ser manipulada.

- **Señales aleatorias:** Son señales que solo se pueden caracterizar de manera estadística ya que toman valores aleatorios en cualquier tiempo dado.
- **Señales deterministas:** Son aquellas cuyos valores están definidos para cualquier momento en el tiempo. Se puede modelar por una función $x(t)$.

3.5. OpenCV

Open Source Computer Vision Library es una librería de software de aprendizaje automático y visión artificial de código abierto.

Los algoritmos dentro de la librería pueden ser utilizados para reconocimiento facial, identificación de objetos, clasificar acciones humanas, seguir objetos en movimiento, extraer modelos 3D de objetos, identificar imágenes parecidas dentro de una base de datos, reconocer escenarios, etcétera.



Figura 1: Logo Open CV

3.6. MatLab

MatLab es una plataforma de programación y cálculo en forma de software orientada a tareas de ingeniería y matemáticas. Es un software creado por la empresa Mathworks y es considerada como una de las herramientas más potentes en cálculo y representación gráfica de problemas complejos.

Se caracteriza por tener su propio lenguaje. Sus funciones varían según las toolboxes que utilicemos.



Figura 2: Logo MatLab

3.6.1. Simulink

Es un entorno de diagramas de bloques que se utiliza para diseñar sistemas con modelos multidimensionales, simular antes de implementar en hardware y desplegar sin necesidad de escribir código. Es una herramienta muy usada en ingeniería de control y procesamiento de señal para la simulación de sistemas y el diseño basado en modelos.

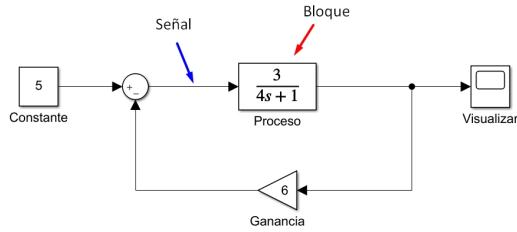


Figura 3: Ejemplo de diagrama de bloques en Simulink

4. Desarrollo

4.1. Modelo matemático

Variables que afectan al sistema		
Id	Variable	Valor
m	Masa de la bola (g)	3
g	Constante gravitacional (m/s)	9.81
R	Radio de la bola (m)	0.04
F_t	Fuerza de traslación (N)	
F_r	Fuerza de rotación (N)	
x	Distancia de la bola al centro (m)	
α	Ángulo de la plataforma (deg)	

La aceleración de la pelota es descrita por:

$$\ddot{x} = \frac{d^2x}{dt^2}$$

La fuerza resultante del movimiento de traslación es:

$$F_t = m\ddot{x}$$

El momento de inercia de un objeto esférico es:

$$J = \frac{2}{5}mR^2$$

La fuerza rotacional se calcula dividiendo el torque por el radio:

$$F_r = \frac{T_r}{R} = \frac{J\ddot{x}}{R^2} = \frac{2}{5}m\ddot{x}$$

Con las dos fuerzas calculadas, ahora se puede resolver la siguiente ecuación:

$$F_r + F_t = mg \sin \alpha$$

$$\ddot{x} = \frac{5}{7}g \sin \alpha$$

Usando Laplace para la posición respecto a los ángulos dados obtenemos:

$$H(s) = \frac{X(s)}{\theta(s)} = \frac{5g}{7s^2} = \frac{0.91}{s^2}$$

4.2. Diagrama de cuerpo libre

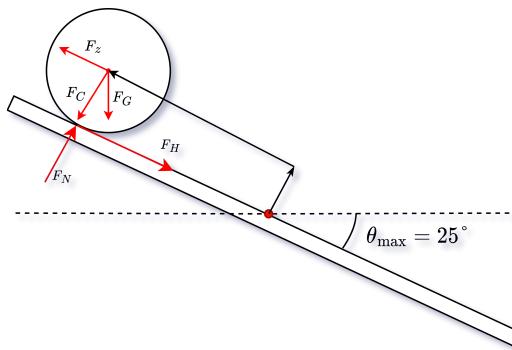


Figura 4: Diagrama de cuerpo libre para ambos ejes

Se consideraron las siguientes fuerzas en el diagrama:

- F_G : Fuerza de gravedad
- F_z : Fuerza centrífuga
- F_C : Fuerza traslacional
- F_H : Movimiento
- F_N : Normal

4.3. Materiales

Se utilizaron los siguientes materiales para la construcción del proyecto:

- Madera
- Cable
- Cámara web HD USB VS-120A ViewSonic

4.4. Maqueta



Figura 5: Cámara

- Servo MG995



Figura 6: Servo

Se utilizaron estos servos porque son los que mejor se ajustaron a las necesidades del proyecto. Se alimentan con 5.5V y tienen fuerza de sobra para cargar la placa.

- Pelota de ping-pong
- Articulaciones para movimiento en dos ejes
- Hercules RM42x



Figura 7: Maqueta Completa

Para la maqueta se usó una base pesada para que esté fija y no aporte turbulencias. Se colocó un poste para poder poner la cámara lo más cercana a 90° respecto a la placa.



Figura 8: Posición de la cámara

Para el movimiento de la placa se insertó una vara en el centro con una articulación circular que le permitiera moverse sobre dos ejes. En la parte baja y a un costado fijamos las articulaciones que se encargarán de mover nuestra placa con los servos.

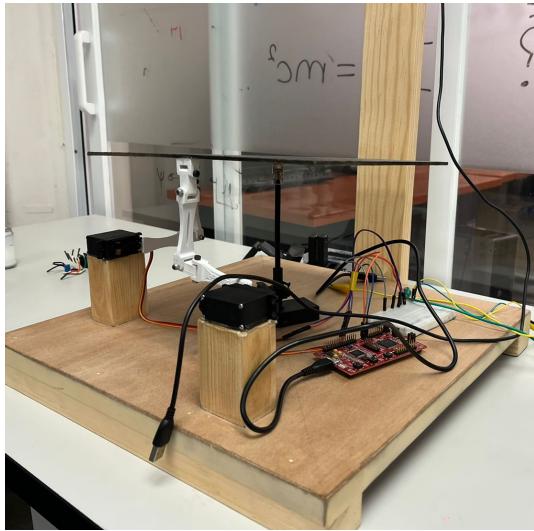


Figura 9: Base de la maqueta

La placa se pintó de negro para que refleje lo menos posible y así no afectara a nuestro programa de reconocimiento de colores para la pelota.

4.5. Adquisición de señales

Para la adquisición de señales fue necesario implementar OpenCV para poder reconocer la posición de la pelota sobre la placa. Con el código de

OpenCV se reconoce la posición dentro del eje x y y y se envía por medio del SCI del microcontrolador.

Una vez recibida la información, es guardada en una cola de variables tipo `char`. Con una algoritmo sencillo, se importan estas variables y se convierten a sus respectivos valores en centenas, decenas y unidades.

```
x_cam = (int32_t)((M_SCI_task.inx1 - '0')
+ 10*(M_SCI_task.inx2 - '0') + 100*(M_SCI_task.inx3
- '0'));
```

Solo se necesitan estos datos para empezar a trabajar, pues los servos solo necesitan la posición actual de la pelota para posicionarla en donde el usuario quiera.

4.6. Pre-Control

Para el precontrol se usó un controlador PD. El pre-control no necesita ser muy preciso respecto a la referencia pero sí lograr que la pelota oscile muy cerca de ella.

Una vez aproximadas estas ganancias, se hicieron pruebas con cambios de referencia para posteriormente identificar al sistema. Se hicieron pruebas con arreglos de doscientos cincuenta datos, lo que es equivalente a pruebas de cinco segundos cada una.

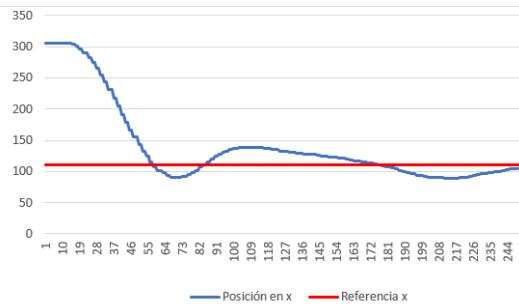


Figura 10: Ejemplo de prueba de posición

4.6.1. Identificación del sistema

Para la identificación se utilizó Excel para aplicar la media a las salidas de las pruebas que realizamos, esto para obtener la salida promedio que tiene el sistema.

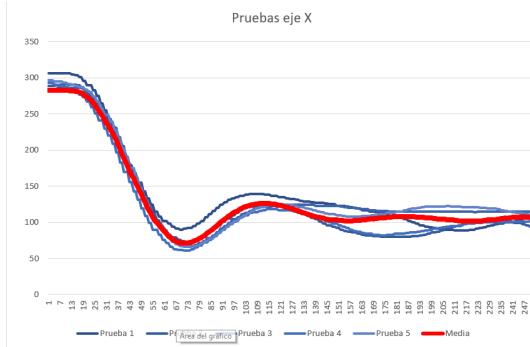


Figura 11: Prueba de la posición en x

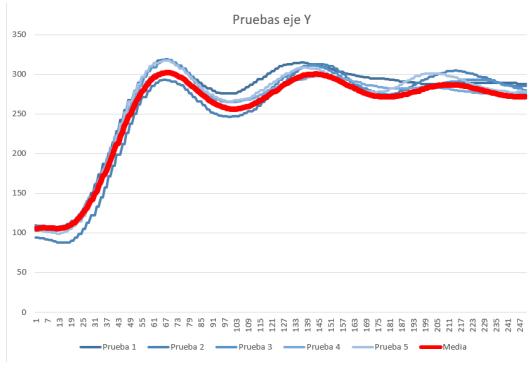


Figura 12: Prueba de la posición en y

Ahora los datos de la media los importamos a nuestro código de mínimos cuadrados, que nos aproximarán una función de transferencia según los datos de nuestro sistema. Esto se hace para los dos ejes.

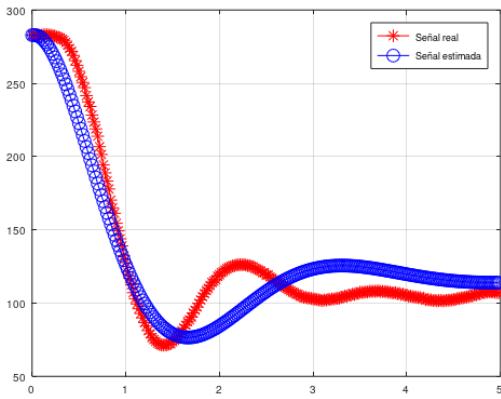


Figura 13: Aplicación de mínimos cuadrados en X

$$\begin{aligned} y_1: & -0.3983 z^3 - 0.5293 z^2 + 0.9293 z \\ & z^3 - 1.282 z^2 - 0.3743 z + 0.6588 \end{aligned}$$

Figura 14: Función de transferencia aproximada del eje X

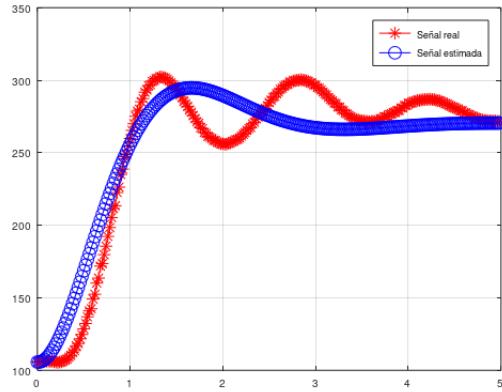


Figura 15: Aplicación de mínimos cuadrados en Y

$$\begin{aligned} y_1: & -0.05212 z^3 - 0.3761 z^2 + 0.4331 z \\ & z^3 - 1.099 z^2 - 0.7143 z + 0.8165 \end{aligned}$$

Figura 16: Función de transferencia aproximada del eje Y

4.7. Control

Ya con las funciones de transferencia de los ejes X y Y, se utilizó Simulink de Matlab para obtener las ganancias de nuestro control.

Se aplicó la herramienta PID tuner para encontrar las ganancias que proporcionarán una buena respuesta a nuestro sistema.

Primero se ingresa la función de transferencia en la planta del sistema.

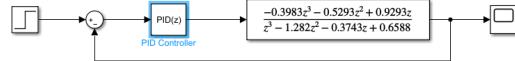


Figura 17: Sistema con función de transferencia de X en Simulink

Posteriormente se entra en el bloque PID y se selecciona la opción "Tune...". Esta opción abrirá

un cuadro donde se sintonizará el sistema con un lazo externo.

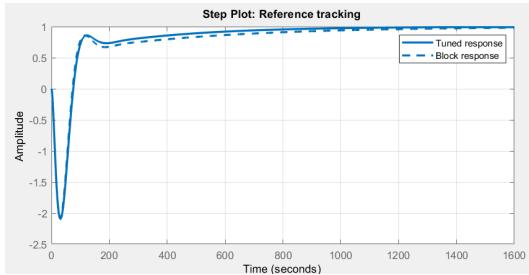


Figura 18: Sintonizador PID para eje X

Con esto se varía qué tan rápido y seguro se necesita que sea nuestro sistema. Una vez se tenga una salida satisfactoria, aplicamos las ganancias obtenidas en nuestro bloque PID y con eso se obtuvo el control.



Figura 19: Parámetros obtenidos con autotuner en el eje X

Para finalizar, solo hay que repetir el procedimiento con el eje Y y aplicar las ganancias al código del proyecto.

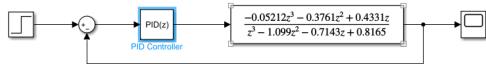


Figura 20: Sistema con función de transferencia de Y en Simulink

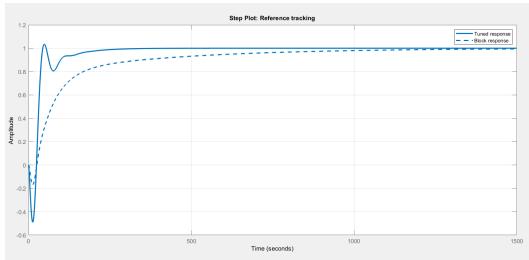


Figura 21: Sintonizador PID para eje Y

P	0.20807	0.586
I	0.012561	0.0042028

Figura 22: Parámetros obtenidos con autotuner en el eje Y

5. Resultados

Ya aplicado nuestro control, se logró estabilizar al sistema en la referencia dada por el usuario. El control se aplicó en un lazo añadido con un periodo de 50ms, que fue el periodo que mejor funcionó.

El sistema estabiliza bastante rápido la pelota, una vez aplicado el control, el precontrol ayuda a mantener la pelota en la referencia hasta recibir nuevas referencias del control, lo que lo hace bastante estable.

A continuación se mostrará una comparación entre el precontrol y el control respecto a la referencia.



Figura 23: Comparación de datos en X

Se observa que nuestra respuesta no es solo más rápida, también es más precisa a la hora de acercarse a nuestra referencia.

6. Conclusiones

Este proyecto fue de gran utilidad para entender las aplicaciones de las herramientas para identificación de sistemas. Gracias a MatLab y Simulink se pudo diseñar y modelar el comportamiento del sistema. Se mejoró la precisión del sistema gracias al control en otro lazo y el ajuste de parámetros.

Se aplicaron varios de los conceptos teóricos y las herramientas vistas en clase. Estos conceptos

fueron clave dentro del proyecto para lograr terminarlo, sin ellos, o el sistema hubiera sido más inestable o lograr el control que tenemos hubiera sido muy tardado.

Este proyecto mejoró los conocimientos en control y sistemas físicos y me enseñó cómo se combina el software avanzado con las técnicas de análisis numérico para hacer de este proceso más eficiente.

6.1. Trabajos futuros

- Control avanzado: Proyectos donde se invocan técnicas más avanzadas de control, como control adaptativo.
- Control multivariable: Se puede aumentar la dificultad del proyecto si se controla más de un balancín 2D al mismo tiempo.
- Uso de trayectorias: No solo controlar el balancín, también se podrían diseñar trayectorias para que la pelota haga alguna figura sobre la placa o haga una rutina donde se vea claro el camino que sigue.
- Cambio de sensores: Hacer el mismo proyecto pero hacerlo más eficiente usando menos recursos, como cambiar la cámara por sensores infrarrojos o atacar el problema desde otro punto de vista.

7. Anexos

7.1. Código OpenCV

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <windows.h>
#include <stdio.h>
#include <string>
#include <fstream>
#include <stdlib.h>
#include <unistd.h>

#define tope 32

using namespace cv;
using namespace std;

int x, y = 0;
char *num;
char buffer[tope];
char esp[] = "x";
char esp1[] = "y";
char ceroAUX[32];
int cero=0;
char Aux_X[32];
char Aux_Y[32];
int i,j;
int x_1, x_2;
int y_1, y_2;
int arrX[3];
int arrY[3];
int ix = 0;
int iy = 0;

//Parametros HSV, maximos y minimos

    int iLowH = 27; //13
    int iHighH = 36; //49

    int iLowS = 130; //92
    int iHighS = 227; //234

    int iLowV = 190; //74
    int iHighV = 255; //189

void barras();
```

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

int main(int argc, char** argv)
{

//ofstream archivo;
//ofstream archi;
//archivo.open("RegistroXY.txt");

    //windows type variables
HANDLE hCom; //Handle variable
DWORD rrlen; // read/write length

    char send[32];

    int port,n;
    char port_name[16];

    VideoCapture cap(0);//capture the video from webcam 0 computadora 1 webcam

    if (!cap.isOpened()) // if not success, exit program
    {
        cout << "Cannot open the web cam" << endl;
        return -1;
    }

    printf("Please input serial port number ");
    scanf("%d",&port);

    sprintf( port_name, "\\\\.\\"COM%d", port );

    hCom = CreateFile( port_name, GENERIC_READ|GENERIC_WRITE,
0, 0, OPEN_EXISTING, 0, 0 );

    //this is the CreateFile() which creates an instance of the SerialPort
    //accessible for read or write operations
    //the code in it is self-explanatory

    if( hCom==INVALID_HANDLE_VALUE )
    {
        printf( "\terror: COM%d is not available.\n", port );
        return -2;
    }
    DCB dcbSerialParams = {0};
```

```
dcbSerialParams.DCBlength=sizeof(dcbSerialParams);

if (!GetCommState(hCom, &dcbSerialParams))
{
    printf("Unable to get the state of serial port");
//error getting state
}

dcbSerialParams.BaudRate=CBR_115200;
dcbSerialParams.ByteSize=8;
dcbSerialParams.StopBits=ONESTOPBIT;
dcbSerialParams.Parity=NOPARITY;

if(!SetCommState(hCom, &dcbSerialParams))
{
    printf("Unable to set serial port settings\n");
    //error setting serial port state
}
COMMTIMEOUTS timeouts={0};

timeouts.ReadIntervalTimeout=50;
timeouts.ReadTotalTimeoutConstant=50;
timeouts.ReadTotalTimeoutMultiplier=10;
timeouts.WriteTotalTimeoutConstant=50;
timeouts.WriteTotalTimeoutMultiplier=10;

if(!SetCommTimeouts(hCom, &timeouts))
{
    printf("Error setting Serial Port timeouts property\n");
    //error occurred. Inform user
}

printf("COM%d opened successfully\n",port);

//Create trackbars in "Control" window
//barras();

int iLastX = -1;
int iLastY = -1;

//Capture a temporary image from the camera
Mat imgTmp;
cap.read(imgTmp);

//Create a black image with the size as the camera output
Mat imgLines = Mat::zeros(imgTmp.size(), CV_8UC3);;

Mat imgHSV;
```

```

    Mat roi_image; //para ROI
    Mat imgThresholded;
    Mat imgOriginal;
////////// ROI///////////
    float scale = 0.55;

int frame_width = cap.get(CAP_PROP_FRAME_WIDTH);
int frame_height = cap.get(CAP_PROP_FRAME_HEIGHT);

int roi_width = frame_width * scale;
int roi_height = frame_height * scale;

int x = 170; //170; // x coordinate of the top-left corner
int y = 110;//110; // y coordinate of the top-left corner

cv::Rect roi_rect;
roi_rect.x = x;
roi_rect.y = y;
roi_rect.width = roi_width;
roi_rect.height = roi_height*1.3;

while (true)
{
    bool bSuccess = cap.read(imgOriginal); // read a new frame from video

    if (!bSuccess) //if not success, break loop
    {
        cout << "Cannot read a frame from video stream" << endl;
        break;
    }

    /* Crop the original image to the defined ROI */
    roi_image = imgOriginal(roi_rect);
    cvtColor(roi_image, imgHSV,COLOR_BGR2HSV);
    inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV),

    imgThresholded); //Threshold the image
    //morphological opening (removes small objects from the foreground)
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
    dilate(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
    //morphological closing (removes small holes from the foreground)
    dilate(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
    erode(imgThresholded, imgThresholded, getStructuringElement(MORPH_ELLIPSE, Size(5, 5)));
    //Calculate the moments of the thresholded image
    Moments oMoments = moments(imgThresholded);
    double dM01 = oMoments.m01;
    double dM10 = oMoments.m10;
    double dArea = oMoments.m00;
    // if the area <= 10000, I consider that there are no object in the

```

```
image and it's because of the noise, the area is not zero
if (dArea > 10000)
{
    //calculate the position of the ball
    int x = dM10 / dArea;
    int y = dM01 / dArea;

    sprintf(ceroAUX, "%d", cero);
    sprintf(Aux_X, "%d", x);
    sprintf(Aux_Y, "%d", y);
    // Comentar esta linea mas adelante
    // cout<< "x "<< x << "y " << y << endl;
    strcpy(send, esp);
    if(x<10)
    {
        strcat(send,ceroAUX);
        strcat(send,ceroAUX);
        strcat(send,Aux_X);

    }else if(x<100)
    {
        strcat(send,ceroAUX);
        strcat(send,Aux_X);
    }else
    {
        strcat(send,Aux_X);
    }
    strcat(send,esp1);
    if(y<10)
    {
        strcat(send,ceroAUX);
        strcat(send,ceroAUX);
        strcat(send, Aux_Y);

    }else if(y<100)
    {
        strcat(send,ceroAUX);
        strcat(send, Aux_Y);
    }

n = strlen(send); //number of bytes to transmit
WriteFile( hCom, send, n, &rwlen, 0 ); //send data through serial port

cout<<send<<endl;
if (iLastX >= 0 && iLastY >= 0 && x >= 0 && y >= 0)
{
    //Draw a red line from the previous point to the current point
    line(imgLines, Point(x+170, y+110),
```

```

        Point(iLastX+170, iLastY+110), Scalar(0, 0, 255), 2);
    }

    iLastX = x;
    iLastY = y;

}

line(roi_image,Point(0,0) , Point(352,350), Scalar(255, 255, 255), 1);
line(roi_image,Point(0,350) , Point(352,0), Scalar(255, 255, 255), 1);

imgOriginal = imgOriginal + imgLines;

imshow("Original", imgOriginal); //show the original image

imshow("Thresholded Image", imgThresholded); //show the thresholded image

imshow("roi image", roi_image);

// strset(receive,0);//clears the string buffer "receive"

//ReadFile( hCom, receive, sizeof(receive), &rwlen, 0 ); // read data from

the serial port buffer of the OS
//printf("%d of out of %d bytes read from port and data is

%s\n",rwlen,sizeof(receive),receive);
//
//      for(j=0;j<sizeof(receive);j++)
//{
//          archivo<<receive[j];
//          if (receive[j] =='\r')
//          {
//              archivo<<endl;
//          }
//      }
if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key

is pressed, break loop
{
    cout << "esc key is pressed by user" << endl;
    break;
}
}//fin de while

CloseHandle( hCom );//close the handle
return 0;
}
void barras()

```

```
{  
  
namedWindow("Control",WINDOW_AUTOSIZE); //create a window called "Control"  
createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)  
createTrackbar("HighH", "Control", &iHighH, 179);  
  
createTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)  
createTrackbar("HighS", "Control", &iHighS, 255);  
  
createTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 255)  
createTrackbar("HighV", "Control", &iHighV, 255);  
  
}
```

7.2. Código Mínimos Cuadrados

```

clc;
clear;
pkg load control;
% Carga de datos
Data = load('pruebas.txt') ;
%Y = load('dfy2.txt');
Y = Data(:,1) ;
Ts = 20e-03; %TIEMPO DE MUESTREO;
N = length(Y);
t = (0 : Ts : Ts*(N - 1)).';
for k = 1 : N
    u(k, 1) = 200;
end

YY = Y(2:end) ;

c1 = Y(1 : end-1);
c2 = [0; Y(1 : end-2)];
c3 = [0; 0; Y(1 : end-3)];
c4 = [u(1 : end-1)];
c5 = [0; u(1 : N - 2)];
c6 = [0; 0; u(1 : N - 3)];
phi = [c1 c2 c3 c4 c5 c6] ;
B = (((phi'*phi)^(-1))*phi'*YY) ;

den = [1, -B(1), -B(2), -B(3)] ;
num = [B(4), B(5), B(6), 0] ;

sys = tf(num, den, Ts)

y_sim = zeros(N, 1) ;
y_sim(1) = Y(1) ;
y_sim(2) = B(1)*y_sim(1) + B(4)*u(1) ;
y_sim(3) = B(1)*y_sim(2) + B(2)*y_sim(1) + B(4)*u(2) + B(5)*u(1);

for i=4:N
    y_sim(i) = B(1)*y_sim(i-1) + B(2)*y_sim(i-2) + B(3)*y_sim(i-3)
    + B(4)*u(i-1) + B(5)*u(i-2)+B(6)*u(i-3);
end

%Hs = tf([0,B(5:end).'],[1,B(1:4).'], Ts, 'Variable', 'z^-1')

figure(1) ;
plot(t, Y, 'r*-', t, y_sim, 'b-o') ;
grid on ;
legend('Señal real', 'Señal estimada');
figure(2)
step(sys);

```

7.3. Código Code Composer

```
/** @file sys_main.c
 * @brief Application main file
 * @date 11-Dec-2018
 * @version 04.07.01
 *
 * This file contains an empty main function,
 * which can be used for the application.
 */

/*
 * Copyright (C) 2009-2018 Texas Instruments Incorporated - www.ti.com
 *
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in the
 * documentation and/or other materials provided with the
 * distribution.
 *
 * Neither the name of Texas Instruments Incorporated nor the names of
 * its contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
*/
/* USER CODE BEGIN (0) */
//defines,directivas
/* USER CODE END */

/* Include Files */
```

```

#include "sys_common.h"

/* USER CODE BEGIN (1) */
#include "FreeRTOS.h"
#include "os_task.h"
#include "os_queue.h"
#include "het.h"
#include "gio.h"
#include "sci.h"
/* USER CODE END */

/** @fn void main(void)
 *  @brief Application main function
 *  @note This function is empty by default.
 *
 *  This function is called after startup.
 *  The user can use this function to implement the application.
*/
/* USER CODE BEGIN (2) */

////////////////// VARIABLES Y FUNCIONES //////////////////

int32_t x_cam; //Posicion en x
int32_t y_cam; //Posicion en y

int i=2; //Iteraciones para lazos

void vControl(void *PvParameters); //Tarea control
void sciNotification(sciBASE_t *sci, uint32 flags);

unsigned char ReceivedData[8]; //Arreglo de datos recibidos del sci

typedef struct {
    unsigned char inx3;
    unsigned char inx2;
    unsigned char inx1;
    unsigned char iny3;
    unsigned char iny2;
    unsigned char iny1;
}miDatSCI;
miDatSCI M_SCI;
xQueueHandle ColaSCI;

typedef struct {
    int32_t x_pos;
    int32_t y_pos;
}CAM_POS;

void pwmSetSignalMIO(hetRAMBASE_t * hetRAM, uint32 pwm, hetSIGNAL_t signal);

```

```

static const uint32 s_het1pwmPolarity[8U] =
{
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
};

void limitControl(int32_t MIN, int32_t MAX, int32_t *x);

hetSIGNAL_t miPWMy, miPWMex;
/* USER CODE END */

int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    hetInit();
    sciInit();
    sciReceive(scilinREG,8,ReceivedData);

    ColaSCI = xQueueCreate(1,sizeof(miDatSCI));
    if (xTaskCreate(vControl, "Control", configMINIMAL_STACK_SIZE, NULL, 0, NULL) != pdTRUE){
        while(1);
    } //error
    vTaskStartScheduler();
    while(1);

/* USER CODE END */

    return 0;
}

/* USER CODE BEGIN (4) */
void vControl(void *PvParameters) //Tarea Control
{
    miDatSCI M_SCI_task;

    // VALORES PRECONTROL
    float Kpx = 0.5;
    float Kdx = 3.5;
    int32_t Upx;
    int32_t Udx = 0;
    int32_t Ux=0;
    float Kpy = 0.5;
    float Kdy = 3.5;
}

```

```
int32_t Upy;
int32_t Udy = 0;
int32_t Uy=0;
CAM_POS ref_task;
ref_task.x_pos = 0;
ref_task.y_pos = 0;
int32_t e_x[5] = {0};
int32_t e_y[5] = {0};

//VALORES CONTROL
float nKpx = 0.08; //Control X
float nKdx = 0.005;
float nKix = 0;
int32_t nUpx;
int32_t nUdx = 0;
int32_t nUiix=0;
int32_t nUx=0;

float nKpy = 0.06; //Control Y
float nKdy = 0.01;
float nKiy = 0;
int32_t nUpy;
int32_t nUdy = 0;
int32_t nUiy = 0;
int32_t nUy=0;

CAM_POS nref_task;
nref_task.x_pos = 175;
nref_task.y_pos = 175;
int32_t ne_x[5] = {0};
int32_t ne_y[5] = {0};

for(;;)
{
    xQueueReceive(ColaSCI, &M_SCI_task, 20/portTICK_RATE_MS);

    x_cam =(int32_t)((M_SCI_task.inx1 - '0') + 10*(M_SCI_task.inx2 - '0') +
    100*(M_SCI_task.inx3 - '0'));
    y_cam =(int32_t)((M_SCI_task.iny1 - '0') + 10*(M_SCI_task.iny2 - '0') +
    100*(M_SCI_task.iny3 - '0'));

    if(i%2==0){//PARA QUE SE USE EL CONTROL CADA 40ms

        ne_x[4] = ne_x[3];
        ne_x[3] = ne_x[2];
        ne_x[2] = ne_x[1];
        ne_x[1] = ne_x[0];
        ne_x[0] = x_cam - nref_task.x_pos;
```

```
ne_y[4] = ne_y[3];
ne_y[3] = ne_y[2];
ne_y[2] = ne_y[1];
ne_y[1] = ne_y[0];
ne_y[0] = y_cam - nref_task.y_pos;

nUpx = (int32_t)(nKpx*ne_x[0]);
nUdx = (int32_t)(nKdx*(ne_x[0]-ne_x[4]));
nUix=nUix+(int32_t)(nKix*ne_x[0]);
nUx = nUpx + nUdx + nUix;

limitControl(-100, 100, &nUx);
nUpy = (int32_t)(nKpy*ne_y[0]);
nUdy = (int32_t)(nKdy*(ne_y[0]-ne_y[4]));
nUiy=nUiy+nKiy*ne_y[0];
nUy= nUpy + nUdy + nUiy;
limitControl(-100, 100, &nUy);

ref_task.x_pos= nref_task.x_pos - nUx;
ref_task.y_pos= nref_task.y_pos - nUy;
}

e_x[4] = e_x[3];
e_x[3] = e_x[2];
e_x[2] = e_x[1];
e_x[1] = e_x[0];
e_x[0] = x_cam - ref_task.x_pos;
e_y[4] = e_y[3];
e_y[3] = e_y[2];
e_y[2] = e_y[1];
e_y[1] = e_y[0];
e_y[0] = y_cam - ref_task.y_pos;

Upx = (int32_t)(Kpx*e_x[0]);
Udx = (int32_t)(Kdx*(e_x[0]-e_x[4]));
Ux = Upx + Udx;
limitControl(-100, 100, &Ux);

Upy = (int32_t)(Kpy*e_y[0]);
Udy = (int32_t)(Kdy*(e_y[0]-e_y[4]));
Uy = Upy + Udy;
limitControl(-100, 100, &Uy);

miPWMx.duty = 767 - Ux;
miPWMx.period = 20000;
pwmSetSignalMIO(hetRAM1, pwm1, miPWMx);

miPWMy.duty = 795 + Uy;
miPWMy.period = 20000;
pwmSetSignalMIO(hetRAM1, pwm0, miPWMy);
```

```
/* if(i%250==0){ //Pruebas con cambio de referencia
    switch (ref_task.x_pos){
        case 100:
            ref_task.x_pos = 250;
            ref_task.y_pos = 250;
            break;
        case 250:
            ref_task.x_pos = 100;
            ref_task.y_pos = 100;
            break;
    }
}*/
i++;

}

void sciNotification(sciBASE_t *sci, uint32 flags)
{
    miDatSCI M_SCI_interr;
    BaseType_t xCoRoutinePreviouslyWoken=0;
    sciReceive(scilinREG,8,ReceivedData);
    M_SCI_interr.inx3 = ReceivedData[1];
    M_SCI_interr.inx2 = ReceivedData[2];
    M_SCI_interr.inx1 = ReceivedData[3];
    M_SCI_interr.iny3 = ReceivedData[5];
    M_SCI_interr.iny2 = ReceivedData[6];
    M_SCI_interr.iny1 = ReceivedData[7];
    xQueueSendFromISR(ColaSCI,&M_SCI_interr,xCoRoutinePreviouslyWoken );
}

void limitControl(int32_t MIN, int32_t MAX, int32_t *x)
{
    int32_t *aux = x;
    if(*aux < MIN)
    {
        *aux = MIN;
    }
    if(*aux > MAX)
    {
        *aux = MAX;
    }
}

void pwmSetSignalMIO(hetRAMBASE_t * hetRAM, uint32 pwm, hetSIGNAL_t signal)
{
    uint32 action;
    uint32 pwmPolarity = 0U;
    float64 pwmPeriod = 0.0F;

    if(hetRAM == hetRAM1)
    {
```

```
pwmPeriod = (signal.period * 1000.0F) / 640.000F;
pwmPolarity = s_het1pwmPolarity[pwm];
}
else
{
}
if (signal.duty == 0U)
{
    action = (pwmPolarity == 3U) ? 0U : 2U;
}
else if (signal.duty >= 10000U)//modificamos el 10000
{
    action = (pwmPolarity == 3U) ? 2U : 0U;
}
else
{
    action = pwmPolarity;
}

hetRAM->Instruction[(pwm << 1U) + 41U].Control =
((hetRAM->Instruction[(pwm << 1U) + 41U].Control) & (~(uint32)(0x00000018U)))
| (action << 3U);
hetRAM->Instruction[(pwm << 1U) + 41U].Data =
(((uint32)pwmPeriod * signal.duty) / 10000U) << 7U ) + 128U;//se modifica el 10000
hetRAM->Instruction[(pwm << 1U) + 42U].Data =
((uint32)pwmPeriod << 7U) - 128U;

/*
/* USER CODE END */
```

Bibliografía

- [1] Cámara Web para PC ViewSonic VS-120A, . URL <https://www.officedepot.com.mx/officedepot/en/Categor%C3%ADa/Todas/computo/teclados-mouse-y-camaras-web/camaras-web/C%C3%A1mara-Web-para-PC-ViewSonic-VS-120A/p/100019112>.
- [2] Centro de ayuda de PTC, . URL https://support.ptc.com/help/mathcad/r9.0/es/index.html#page/PTC_Mathcad_Help/example_signals_and_classification_of_signals.html.
- [3] MG995 Servo motor, . URL <https://electrocrea.com/products/servo-motor-mg995>.
- [4] Simulación y diseño basado en modelos con Simulink, . URL <https://la.mathworks.com/products/simulink.html>.
- [5] What are State-Space Models? - MATLAB SimulInk - MathWorks América Latina, . URL <https://la.mathworks.com/help/ident/ug/what-are-state-space-models.html>.
- [6] MATLAB logo, 6 2022. URL <https://1000marcas.net/matlab-logo/>.
- [7] J. S. Franco. Introducción a la identificación de sistemas, 1 2009. URL <https://www.tecnicaindustrial.es/introduccion-a-la-identificacion-de-sistemas/>.
- [8] J. Llamas. Matlab. *Economipedia*, 2 2023. URL <https://economipedia.com/definiciones/tecnologia/matlab.html>.
- [9] M. Merino. Modelamiento de Sistemas usando Simulink (I) - Trivial Report, 9 2021. URL <https://trivialreport.com/simulink/modelamiento-de-sistemas-usando-simulink-i/>.
- [10] OpenCV. Media Kit - OpenCV, 7 2020. URL <https://opencv.org/resources/media-kit/>.
- [11] OpenCV. About - OpenCV, 11 2020. URL <https://opencv.org/about/>.