



MANUAL DE USUARIO

Programación y operación de robot unicycle

Sebastián Jair Granados Canarúa
Francisco Javier Camacho Jaimes

2025



ESCUELA DE INGENIERÍA
MECÁNICA

CONTENIDO

1. Generalidades del robot unicycle.
2. Preparación de entornos (Arduino IDLE)
3. Configuración del sistema de comunicación.
4. Configuración del sistema controlador basado en ADRC.
5. Establecimiento de trayectorias.
6. Puesta en marcha del sistema de control

1.GENERALIDADES DEL ROBOT UNICICLO.

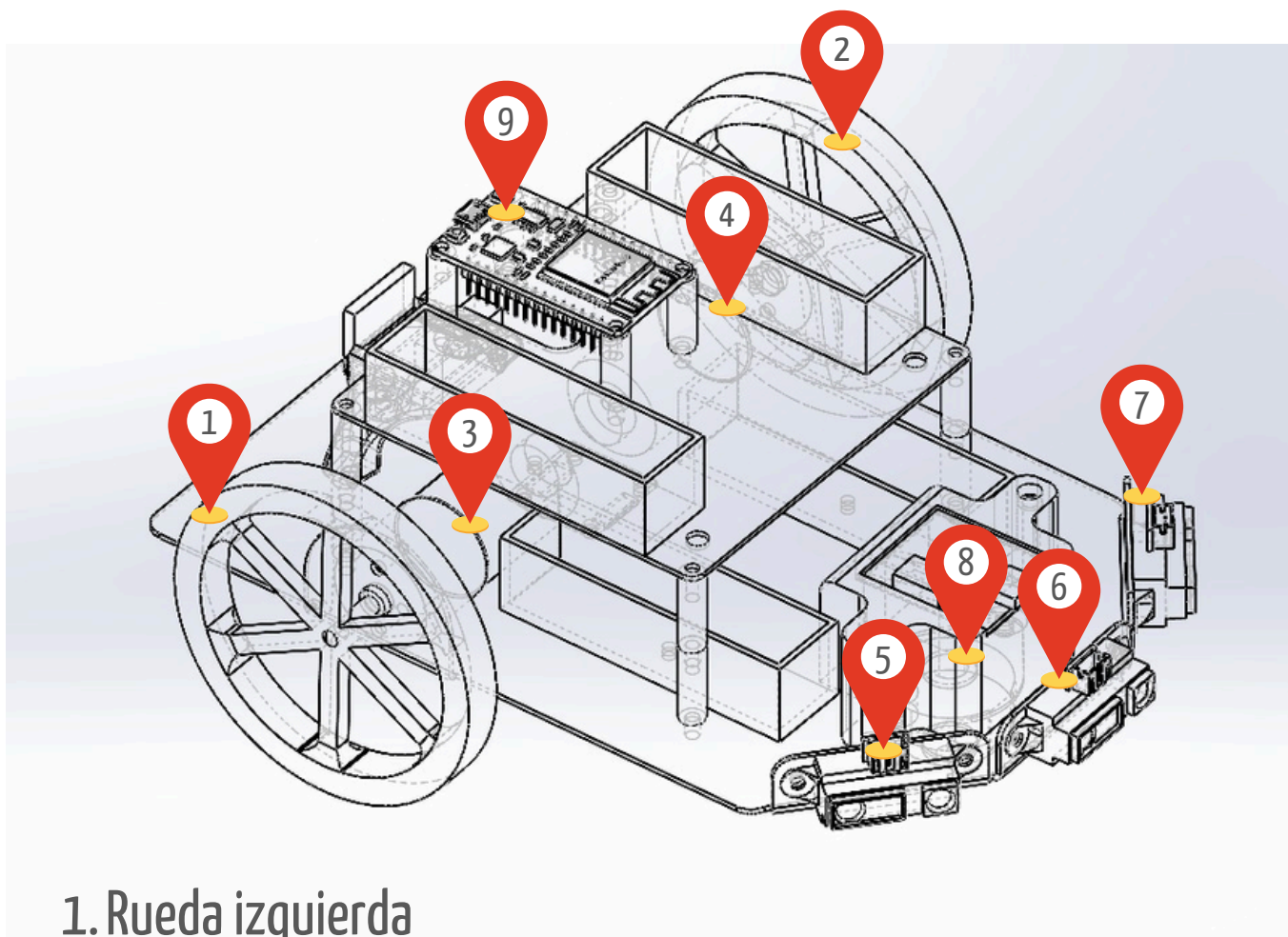
Características mecánicas del robot:

- Peso: 1 Kg
- Radio de rueda: 40.25 mm
- Distancia entre ruedas: 160 mm
- Velocidad máxima alcanzable: 330 RPM
- Velocidad mínima de trabajo: 140 RPM

Características electrónicas del robot:

- PLC: ESP32 WROOM 32
- Potencia eléctrica necesaria: WATTS
- Sensores de detección de obstáculos: Infrarrojos tipo SHARP
- Motores: GM25370
- Giroscopio: MPU6050
- Encoders: Incorporados a los motores, de cuadratura

1.GENERALIDADES DEL ROBOT UNICICLO.



1. Rueda izquierda
2. Rueda derecha
3. Motor izquierdo
4. Motor derecho
5. Sensor derecho
6. Sensor central
7. Sensor izquierdo
8. Rueda loca
9. ESP32 WROOM

2.PREPARACIÓN DE ENTORNOS (ARDUINO IDLE)

Instalación de controladores para la ESP32:

Pegar el siguiente enlace en el espacio que se abre:

https://espressif.github.io/arduino-esp32/package_esp32_index.json



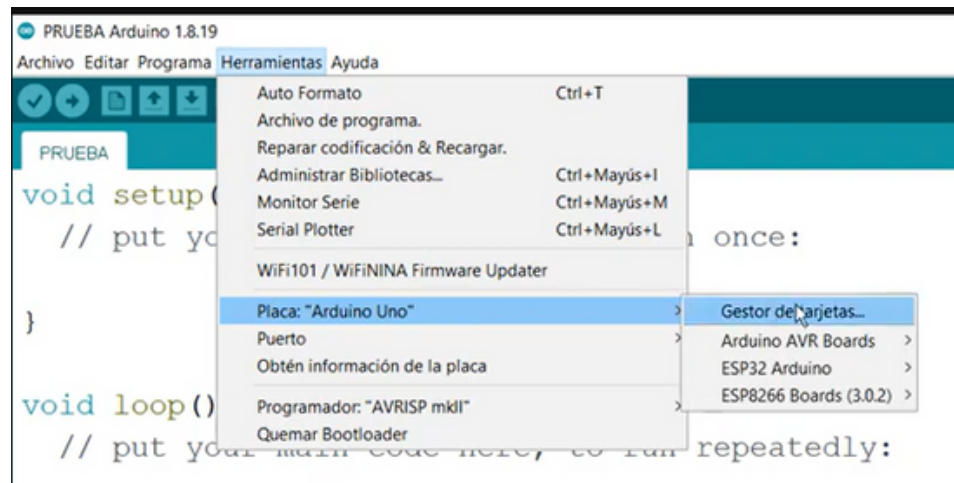
Posteriormente hacer clic en OK



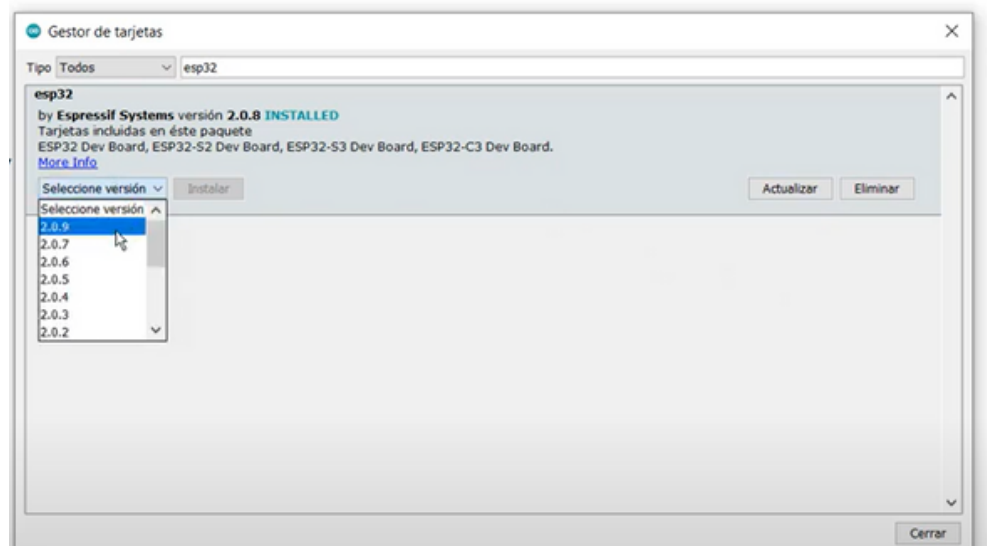
2. PREPARACIÓN DE ENTORNOS (ARDUINO IDLE)

Instalación de controladores para la ESP32:

Luego, en el menú desplegable, se ingresa al gestor de tarjetas del IDLE de Arduino



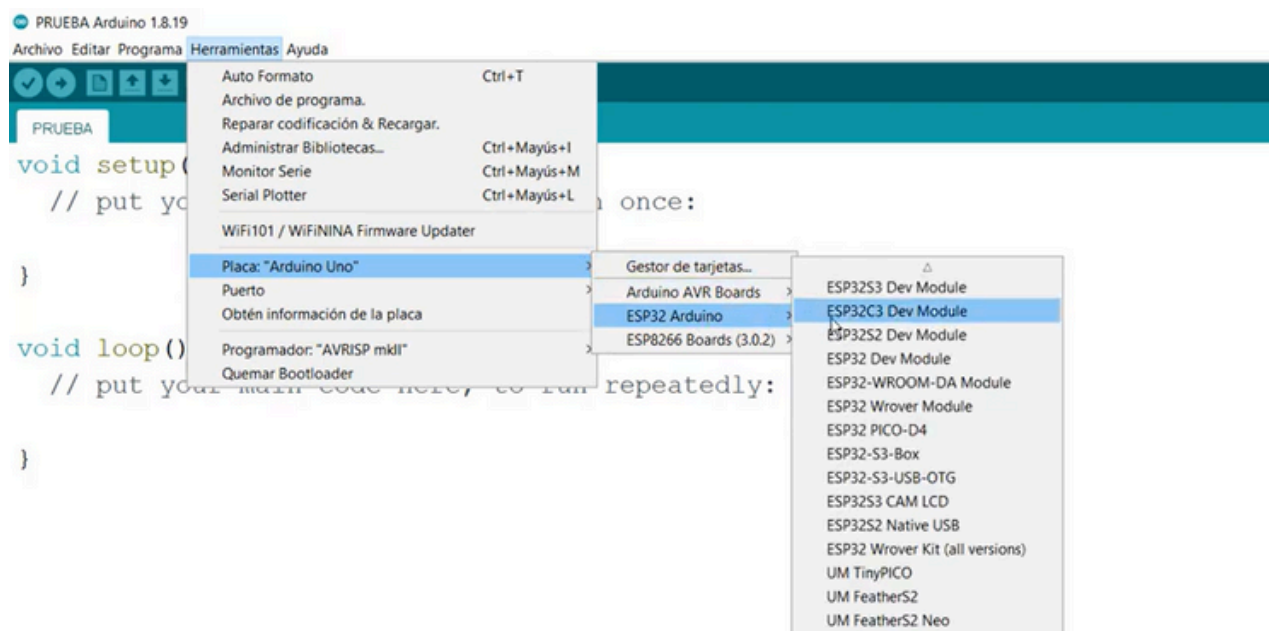
Se busca la ESP32, se selecciona la ultima versión y se da clic en instalar



2.PREPARACIÓN DE ENTORNOS (ARDUINO IDLE)

Instalación de controladores para la ESP32:

Luego, se selecciona la placa ESP32 según la referencia de la lista desplegable, como se muestra a continuación:

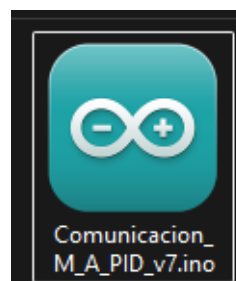
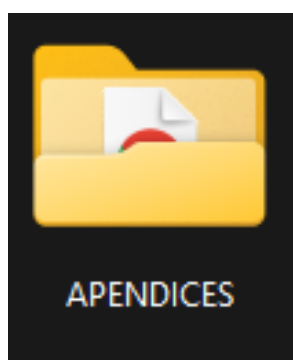


una vez seleccionada la placa ya se puede usar el entorno de Arduino IDE para realizar la programación

3. CONFIGURACIÓN DEL SISTEMA DE COMUNICACIÓN

Configuración del programa en la ESP32

En la carpeta “APENDICES” busque y abra el archivo
“Comunicacion_M_A_PID_v7.ino” en el idle de arduino.



```

Comunicacion_M_A_PID_v7 | Arduino IDE 2.3.2
Archivo Editar Sketch Herramientas Ayuda
Seleccionar Placa

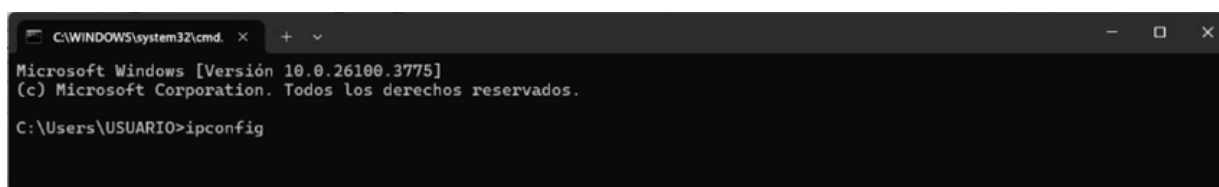
Comunicacion_M_A_PID_v7.ino
1 #include <Arduino.h>
2 #include <WiFi.h> // Librería para manejar la conexión WiFi
3 #include "ESP32Encoder.h" // Librería para encoders cuadratura
4 #define LED_INTEGRADO 2 // LED azul integrado
5
6 ////////////////////////////////////////////////// CONFIGURACION RED WIFI ///////////////////////////////////
7 const char* ssid = "Sebastián G"; // Nombre de la red Wi-Fi
8 const char* password = "sjgranados"; // Contraseña de la red Wi-Fi
9 // Objeto cliente WiFi
10 WiFiClient client;
11 // Configuración del servidor MATLAB
12 const char* serverAddress = "192.168.51.3"; // Dirección IP del PC (MATLAB)
13 const uint16_t serverPort = 25000; // Puerto del servidor MATLAB
14
15 ////////////////////////////////////////////////// CONEXIÓN WIFI ///////////////////////////////////
16 // Variables globales adicionales
17 unsigned long ultimaConexion = 0;
18 const unsigned long IntervaloConexion = 5000; // 5 segundos
19 ////////////////////////////////////////////////// COMUNICACION SERIAL ///////////////////////////////////
20
21 // Tamaño esperado de la trama: 2 floats = 8 bytes
22 const int BytesEnvio = 8;
23 const int BytesRegreso = 8;
24 uint8_t recvBuffer[BytesRegreso]; // Buffer para datos recibidos
25 uint8_t sendBuffer[BytesEnvio]; // Buffer para datos a enviar
26
27 ////////////////////////////////////////////////// Variables control PI ///////////////////////////////////
28 double Setpoint1 = 0.0, Setpoint2 = 0.0; // Setpoints recibidos
29 double uR = 0.0, uL = 0.0; // Velocidades de cada rueda
30 double Input1 = 0.0, Input2 = 0.0; // Velocidad Media (Lin y Ang)
31 double kpU = 450, kiU = 2.0, kdU = 1.2; // Ganancias para velocidad lineal
32 double kpW = 58, kiW = 0.108, kdW = 0.1; // Ganancias para velocidad angular
33 double PTerm1 = 0.0, PTerm2 = 0.0; // Errores Proporcionales del control
34 double ITerm1 = 0.0, ITerm2 = 0.0; // Errores Integrales del control
35 double DTerm1 = 0.0, DTerm2 = 0.0; // Errores Derivativos del control
36 double outMin = -255, outMax = 255; // Límites de PWM
37 int pwm1 = 0, pwm2 = 0; // Señales PWM para los motores
38 double lastError1 = 0, lastError2 = 0;
39
40 unsigned long lastTime = 0; // Tiempo anterior
41 unsigned long sampleTime = 120; // Tiempo de muestreo
42
43 // Constantes de conversión y geometría
44 const double valueConst1 = 1.886842, valueConst2 = 0.07604, wheelDistance = 0.155; // Vel Angular [rad/s], Vel Lineal [cm/s]
45 //const int R = 3330; // Resolución del encoder R = mm*s*r.
46

```


3. CONFIGURACIÓN DEL SISTEMA DE COMUNICACIÓN

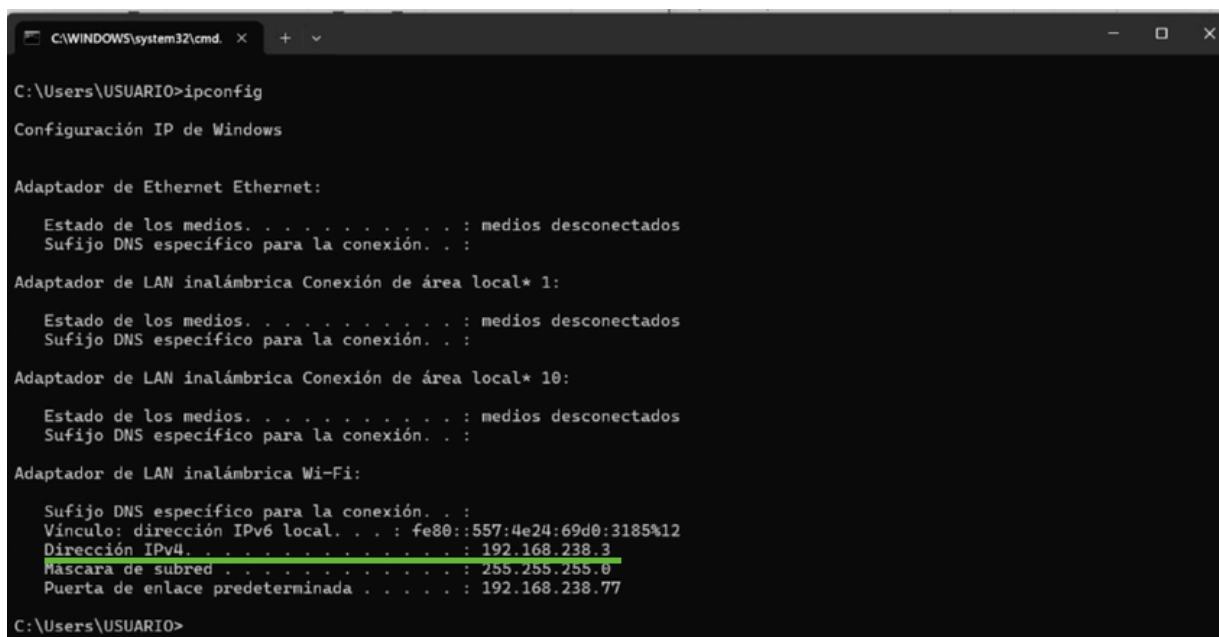
Configuración del programa en la ESP32

Se ejecuta el simbolo del sistema en el ordenador en el que se va a hacer la implementación, y ejecute el comando “IPCONFIG”, posteriormente se presiona “ENTER”



```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Versión 10.0.26100.3775]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\USUARIO>ipconfig
```

Posteriormente se captura el valor de la dirección IPV24, como se indica en la siguiente imagen:



```
C:\WINDOWS\system32\cmd. x + v
C:\Users\USUARIO>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Conexión de área local* 1:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Conexión de área local* 10:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . :
    Vínculo: dirección IPv6 local. . . : fe80::557:4e24:69d0:3185%12
    Dirección IPv4. . . . . : 192.168.238.3
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . : 192.168.238.77

C:\Users\USUARIO>
```

3. CONFIGURACIÓN DEL SISTEMA DE COMUNICACIÓN

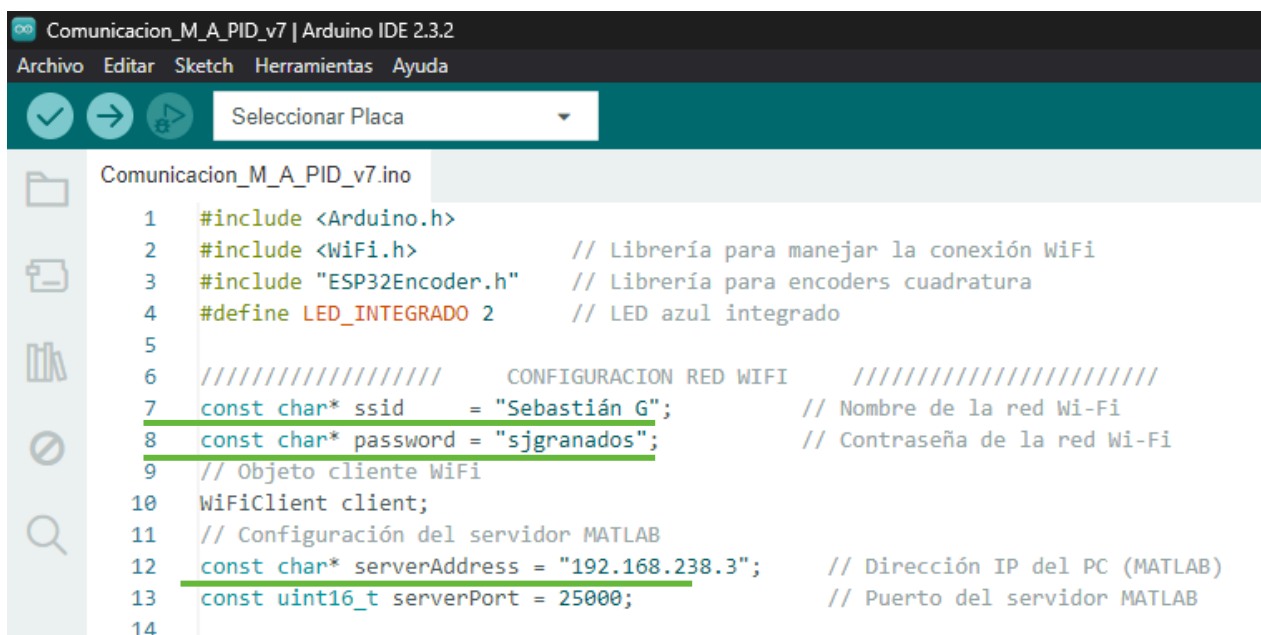
Configuración del programa en la ESP32

En las líneas 7 y 8 escriba los datos de la red wifi a la que esté conectada el dispositivo, también en la línea 12, por favor poner la IP del paso anterior.

LN7: “Nombre de la red”

LN8: “Contraseña de la red”

LN12: “IPV24”



```

Comunicacion_M_A_PID_v7 | Arduino IDE 2.3.2
Archivo  Editar  Sketch  Herramientas  Ayuda

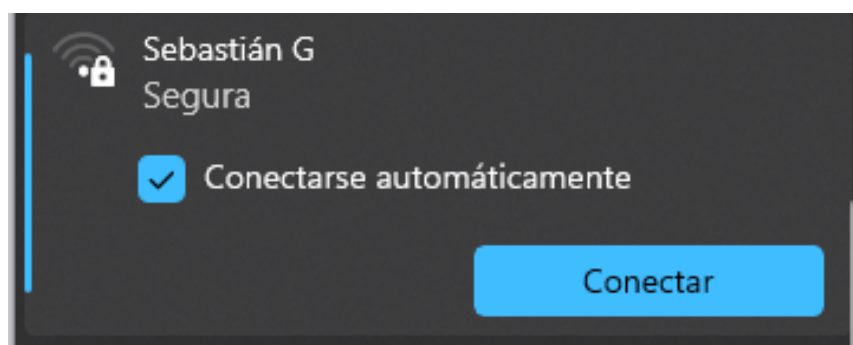
✓ → ⚙ Selecciones Placa

Comunicacion_M_A_PID_v7.ino
1  #include <Arduino.h>
2  #include <WiFi.h>           // Librería para manejar la conexión WiFi
3  #include "ESP32Encoder.h"   // Librería para encoders cuadratura
4  #define LED_INTEGRADO 2     // LED azul integrado
5
6  ////////////////////////////////////////////////// CONFIGURACION RED WIFI ///////////////////////////////////
7  const char* ssid = "Sebastián G";           // Nombre de la red Wi-Fi
8  const char* password = "sjgranados";        // Contraseña de la red Wi-Fi
9  // Objeto cliente WiFi
10 WiFiClient client;
11 // Configuración del servidor MATLAB
12 const char* serverAddress = "192.168.238.3"; // Dirección IP del PC (MATLAB)
13 const uint16_t serverPort = 25000;          // Puerto del servidor MATLAB
14
  
```

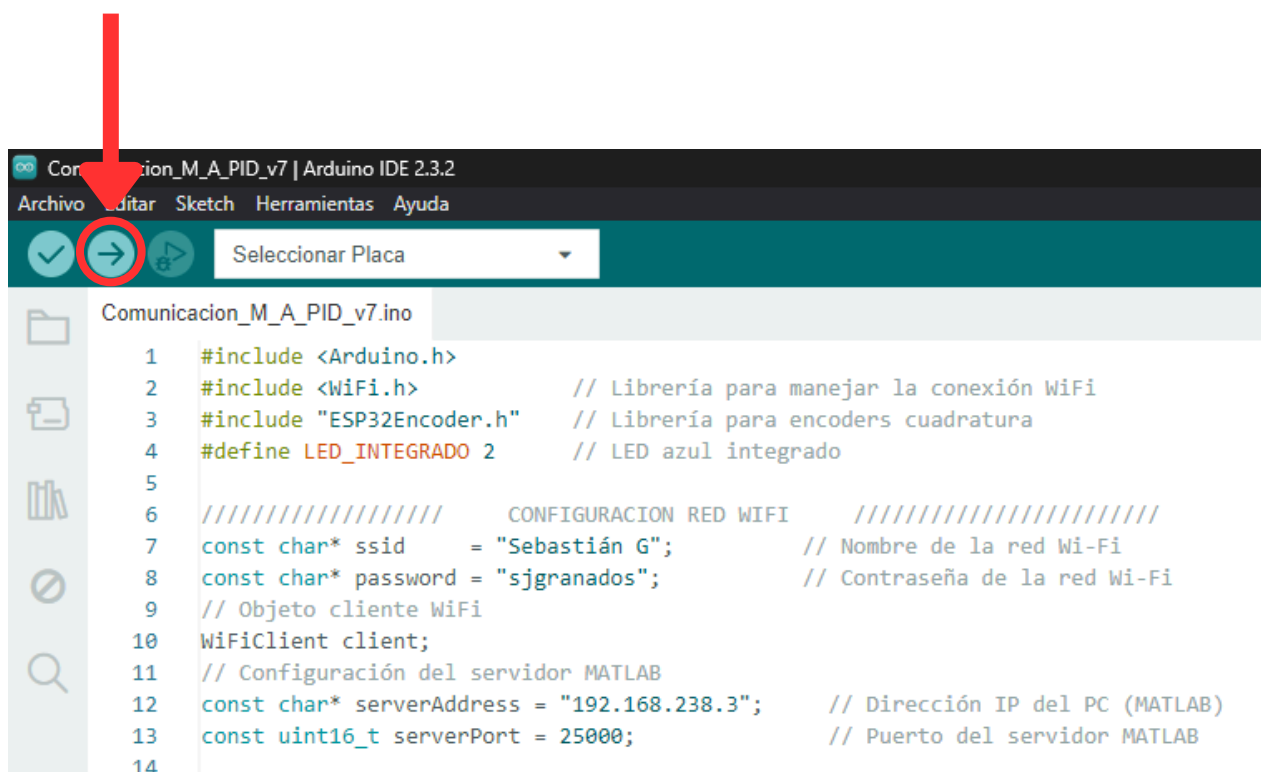
3. CONFIGURACIÓN DEL SISTEMA DE COMUNICACIÓN

Configuración del programa en la ESP32

Es importante resaltar que el PC debe estar conectado a la misma red Wifi que el robot



Finalmente, se graba el programa en la placa de control



4, CONFIGURACIÓN DEL SISTEMA CONTROLADOR BASADO EN ADRC.

Configuración del programa en MATLAB

En la carpeta “APENDICES” busque y abra el archivo
“IMPLEMENTACION_ADRC.mlx”



Es importante resaltar que el PC debe estar conectado a la misma red Wifi que el robot, adicionalmente se relaciona el numero de puerto para que coincida con el que se relaciona en la configuración wifi de la placa en arduino.

```

72 %% TCP/IP Configuración
73 disp('Iniciando configuración del servidor TCP/IP...');
74 port = 25000;
75 tcpObj = tcpserver("0.0.0.0", port, 'Timeout', 10);
76 dataSize = 8;
77
78 disp(['Servidor TCP iniciado en puerto ' num2str(port) ', esperando conexión...']);
79 while ~tcpObj.Connected
80     pause(1);
81 end
82 disp('Cliente conectado. ');
83 timeoutLimit = 5;
84 ultimoPaquete = tic;
85

```

5. ESTABLECIMIENTO DE TRAYECTORIAS.

Parametrización de trayectorias

El sistema de seguimiento de trayectorias implementado en este robot requiere que las trayectorias estén expresadas en función del tiempo para poder generar referencias dinámicas en cada instante de control. Por ello, es necesario convertir cualquier expresión de trayectoria dada en forma cartesiana, es decir, de la forma

$$y = f(x),$$

A una forma paramétrica dependiente del tiempo:

$$x(t), y(t).$$

La parametrización permite definir una trayectoria continua en el plano, donde cada punto de la trayectoria está asociado a un instante de tiempo específico., usualmente el tiempo t , y se describe tanto la coordenada x como la coordenada y como funciones explícitas de t . Esto es indispensable para que el controlador del robot pueda calcular errores de posición en función del tiempo y generar señales de control adecuadas.

5. ESTABLECIMIENTO DE TRAYECTORIAS.

Parametrización de trayectorias

En las líneas “” del código se presentan las variables

$$hxd_ADRC(t) =$$

$$hyd_ADRC(t) =$$

las cuales definen la posición para “x” y “y” en espacio para cada instante “t” de tiempo, a través de las ecuaciones parametrizadas, a continuación se exponen 3 ejemplos de ecuación parametrizada:

```
% %1.Trayectoria deseada (Ejemplo sencillo: Escalon)
hxd_ADRC = 0.1 * t_ADRC; % Trayectoria deseada en x
hyd_ADRC = 0.00001 * t_ADRC + 0.5 ; % Trayectoria deseada en y
```

```
% 2.Trayectoria deseada (Ejemplo sencillo: trayectoria circular)
hxd_ADRC = 1 * cos(0.15 * t_ADRC); % Trayectoria deseada en x
hyd_ADRC = 1 * sin(0.15 * t_ADRC); % Trayectoria deseada en y
```

```
% 3.Trayectoria deseada (Ejemplo sencillo: trayectoria senoide)
hxd_ADRC = 0.1 * t_ADRC; % Trayectoria deseada en x
hyd_ADRC = 0.4 * sin(0.3 * t_ADRC); % Trayectoria deseada en y
```

5. ESTABLECIMIENTO DE TRAYECTORIAS.

Parametrización de trayectorias

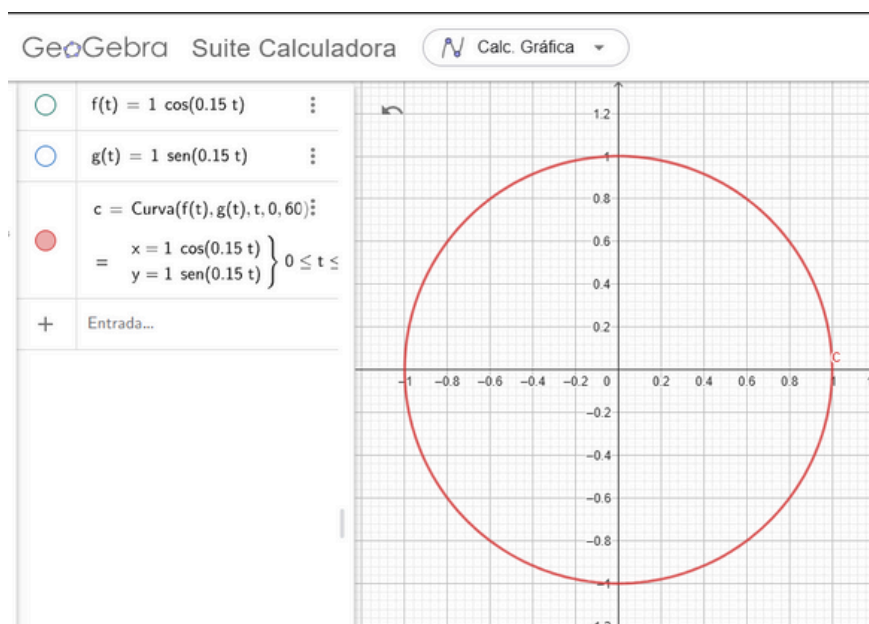
En las líneas “” del código se presentan las variables:

$$hxd_ADRC(t) =$$

$$hyd_ADRC(t) =$$

las cuales gobiernan el sistema en las coordenadas x, y.

Se sugiere utilizar una herramienta computacional como geogebra para visualizar las ecuaciones antes de implementarlas en el programa, es importante mencionar que el programa interpreta cada unidad con una magnitud de 1m.



6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Sintonización del controlador.

Es importante mencionar que el controlador gobierna velocidad lineal y velocidad angular de forma independiente bajo la misma estructura y de forma simultanea, según las siguientes ecuaciones.

$$v = k_0 v_d + k_1 \int v dt + k_2 \frac{de_x}{dt} + k_3 dx$$

$$\omega = k_0 e_\theta + k_1 \int \omega dt + k_2 \frac{de_y}{dt} + k_3 dy$$

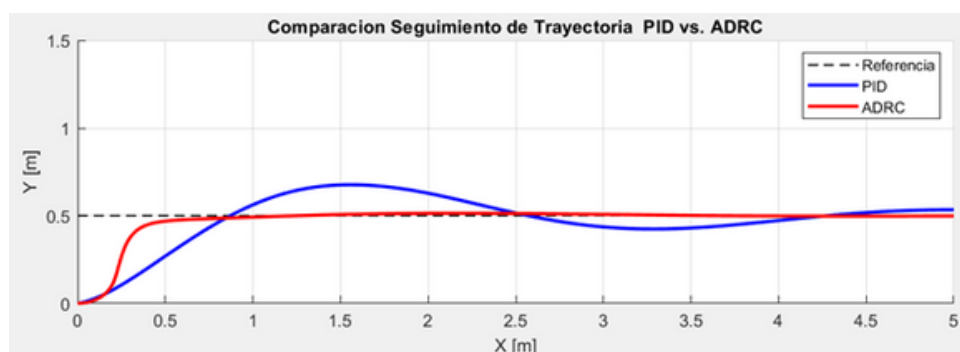
6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Sintonización del controlador.

Trayectorias lineales

Para trayectorias rectas, el error es principalmente en dirección transversal. El sistema es poco exigente dinámicamente. Se recomienda:

- K_p moderado: suficiente para corregir rápidamente el error.
- K_i bajo o incluso cero si no hay error en estado estacionario.
- K_d bajo: no hay grandes cambios de curvatura.
- N entre 10 y 20: filtra el ruido sin retrasar demasiado la respuesta.



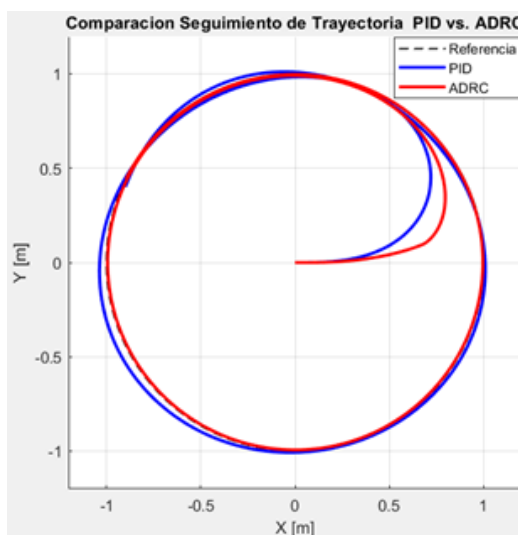
6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Sintonización del controlador.

Trayectorias circulares

La trayectoria exige un cambio continuo en la dirección, generando errores angulares y de curvatura.

- K_p mayor que en la trayectoria lineal, para aumentar la respuesta ante errores crecientes.
- K_i moderado: útil para compensar errores acumulados por desajustes del modelo.
- K_d mayor que en línea recta: ayuda a predecir los cambios de dirección.
- N entre 15 y 30: necesario para atenuar el ruido sin perder capacidad de predicción.



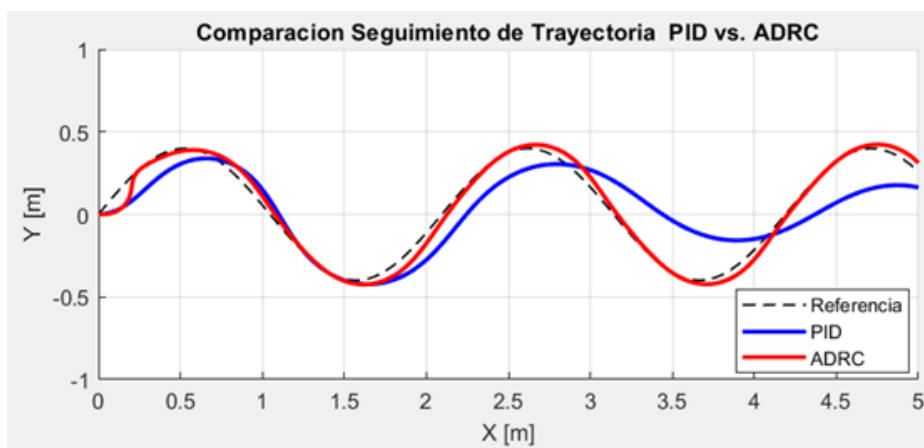
6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Sintonización del controlador.

Trayectorias senoidales

Este tipo de trayectoria impone cambios frecuentes en la dirección del robot, por lo que se requiere una respuesta dinámica más ágil.

- K_p alto: para respuesta rápida.
- K_i : puede inducir oscilaciones si se exagera.
- K_d alto: mejora la anticipación de cambios de dirección.
- N entre 20 y 50: el derivativo filtra el ruido pero debe ser suficientemente rápido para seguir las oscilaciones.



6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Proceso iterativo de ajuste

El proceso de sintonización se realiza experimentalmente por prueba y error, siguiendo estos pasos:

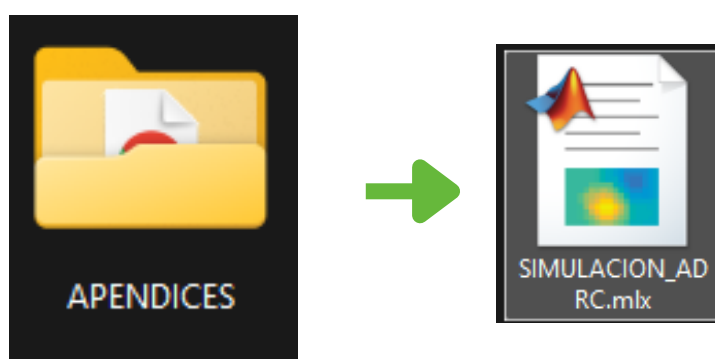
- Comenzar con $K_i = 0$, $K_d = 0$, y ajustar K_p hasta obtener un seguimiento sin oscilaciones.
 - Aumentar gradualmente K_d , observando si mejora la respuesta en curvas o si introduce ruido.
 - Ajustar el valor de N para suavizar.
 - Incorporar K_i si existe un error de seguimiento persistente.
- Finalmente, refinar los tres parámetros con pequeñas variaciones y validar en cada tipo de trayectoria.

Se recomienda observar gráficamente el error de seguimiento $e(t)$, así como la trayectoria real frente a la deseada. .

6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Simulación del controlador

es recomendable simular el sistema antes de implementarlo, para esto es posible utilizar el archivo “SIMULACIÓN_ADRC” que se encuentra grabado en la carpeta “MANUAL”



En las líneas 178 a 188 se definen las ecuaciones parametricas, segun el tipo de trayectoria a seguir, solo puede haber una trayectoria activa a la vez, las otras deben estar comentadas con %

```

178 % 1.Trayectoria deseada (Ejemplo sencillo: Escalon)
179 % hxd_ADRC = 0.1 * t_ADRC; % Trayectoria deseada en x
180 % hyd_ADRC = 0.00001 * t_ADRC + 0.5 ; % Trayectoria deseada en y
181
182 % 2.Trayectoria deseada (Ejemplo sencillo: trayectoria circular)
183 hxd_ADRC = 1 * cos(0.15 * t_ADRC); % Trayectoria deseada en x
184 hyd_ADRC = 1 * sin(0.15 * t_ADRC); % Trayectoria deseada en y
185
186 % 3.Trayectoria deseada (Ejemplo sencillo: trayectoria senoide)
187 % hxd_ADRC = 0.1 * t_ADRC; % Trayectoria deseada en x
188 % hyd_ADRC = 0.4 * sin( 0.3 * t_ADRC); % Trayectoria deseada en y

```

6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Simulación del controlador.

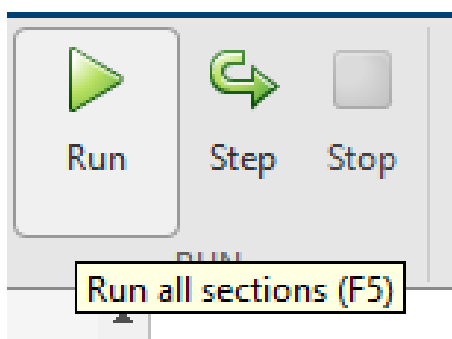
Posteriormente en las líneas de la 190 a la 212 se establecen los parametros de control para el bloque basado en ADRC, según el tipo de trayectoria

```

190 %% Escalon -> t=60s
191 % Kp =0.1;      % Ganancia proporcional
192 % Ti = 0.04;    % Tiempo integral
193 % Td = 0.0028;  % Tiempo derivativo
194 % N = 7;        % Factor del filtro derivativo
195 % alpha = 0.3;  % Parámetro de filtrado para la derivada (0 < alpha < 1)
196 % beta = 0.95; % Estimación de la perturbación
197
198 %% Parámetros del PID Filtrado (Simulando un control ADRC) - Circulo -> t=40s
199 Kp =0.2;        % Ganancia proporcional
200 Ti = 0.08;      % Tiempo integral
201 Td = 0.001;     % Tiempo derivativo
202 N = 7;          % Factor del filtro derivativo
203 alpha = 0.3;
204 beta = 0.95;
205
206 %% Sinusoide -> t=60s
207 % Kp =0.15;     % Ganancia proporcional
208 % Ti = 0.02;    % Tiempo integral
209 % Td = 0.006;   % Tiempo derivativo
210 % N = 7;        % Factor del filtro derivativo
211 % alpha = 0.35;
212 % beta = 0.95;

```

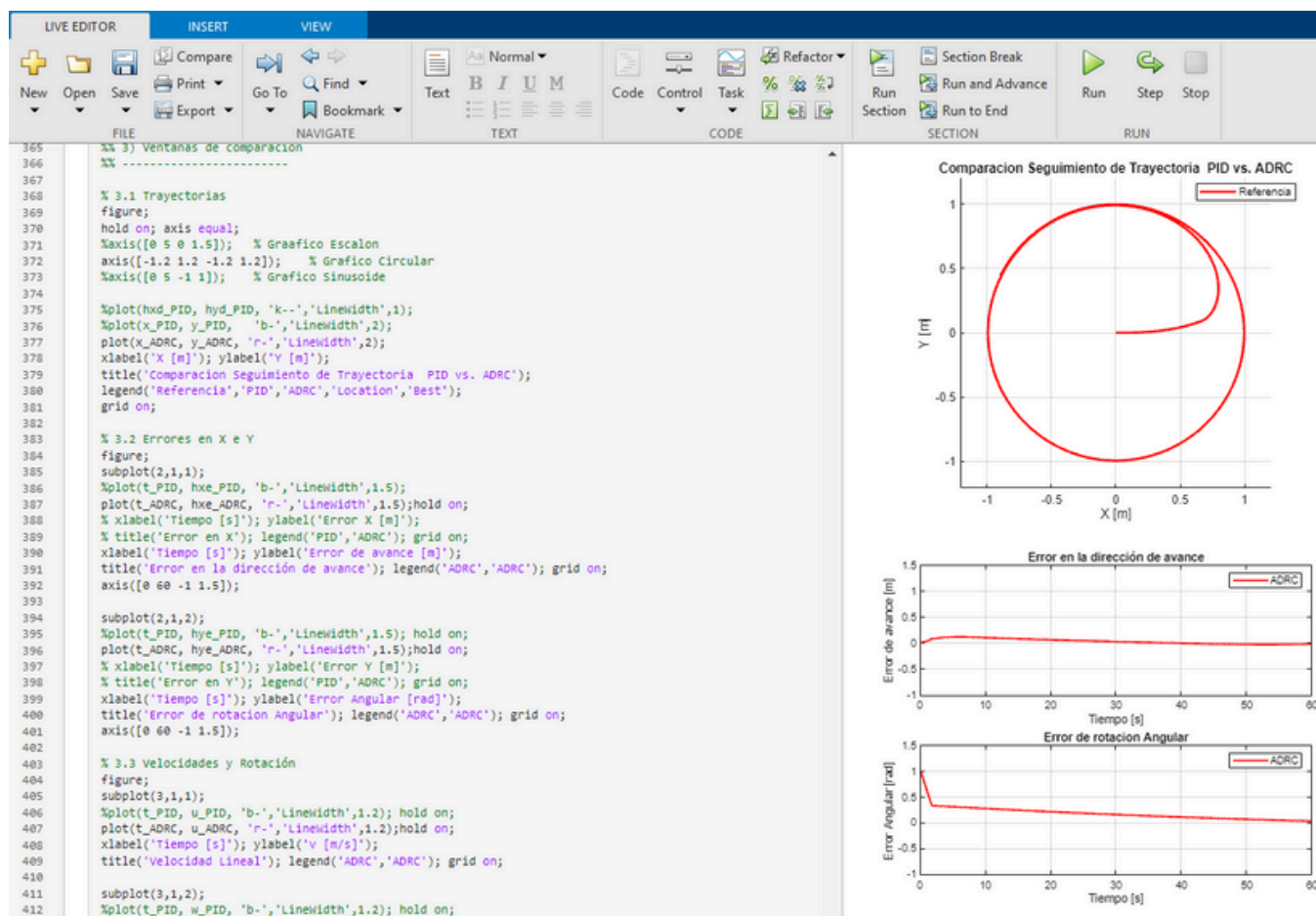
finalmente se ejecuta el script



6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Simulación del controlador.

Es importante verificar la información que entregan las graficas, y si es necesario realizar correcciones en los parametros de control



6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Implementación del controlador .

Una vez confirmada la trayectoria, y garantizada la tendencia de los factores del controlador, busque el archivo

“IMPLEMENTACIÓN_ADRC.mlx” y ejecutelo, a continuación configure la trayectoria deseada en las líneas 14-15 para que se expresen de forma parametrica

```

1      clear; close all; clc;
2      warning off;
3
4      %% Parámetros del Robot
5      tf = 40; % Tiempo total de simulación (s)
6      ts = 0.1; % Paso de tiempo
7      t = 0:ts:tf;
8      Q = length(t) - 1;
9
10     % Estado inicial [ x ; y ; theta] --> (m, m, rad)
11     X = [ 0 ; 0 ; 0 ];
12
13     % 1.Trayectoria deseada (Ejemplo sencillo: Escalon)
14     hxd = 0.1 * t; % Trayectoria deseada en x
15     hyd = 0.00001 * t + 0.5 ; % Trayectoria deseada en y
16

```

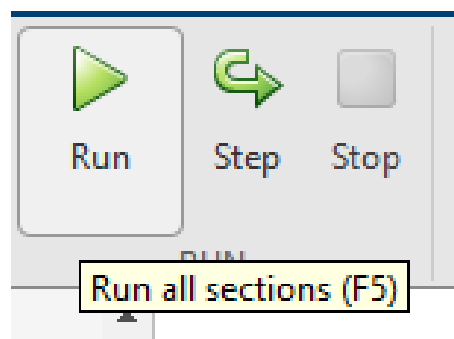

6. PUESTA EN MARCHA DEL SISTEMA DE CONTROL

Implementación del controlador .

Luego establezca los parametros de control del robot, en las lineas de la 18 a la 22

```
18 %% Parámetros del PID Filtrado (Simulando un control ADRC)
19 Kp =0.01;      % Ganancia proporcional
20 Ti = 0.4;      % Tiempo integral
21 Td = 0.002;    % Tiempo derivativo
22 N = 16;       % Factor del filtro derivativo
```

finalmente se ejecuta el script



finalmente se verifica el comportamiento del controlador en el robot movil con ayuda de las gráficas que se generan durante la ejecución, y de ser necesario ajuste los parametros nuevamente teniendo en cuenta el orden descrito en la sección anterior.