UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"MÁQUINA DE ESTADOS FINITOS PARA CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA"

AUTOR: ANDERSON STEEVEN BORRALLOS PÉREZ

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA-ECUADOR
2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD	100429912 – 7		
APELLIDOS Y NOMBRES	BORRALLOS PÉREZ ANDERSON STEEVEN		
DIRECCIÓN	Enrique Terán y Panamericana , San Roque		
EMAIL	asborrallosp@utn.edu.ec - borrallosanderson@gmail.com		
TELÉFONO FIJO	– – –	TELÉFONO MÓVIL	0967061347
DATOS DE LA OBRA			
TÍTULO	“MÁQUINA DE ESTADOS FINITOS PARA CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA”		
AUTOR	ANDERSON STEEVEN BORRALLOS PEREZ		
FECHA	04 JULIO 2023		
PROGRAMA	PREGRADO		
TÍTULO POR EL QUE OPTA	INGENIERO EN MECATRÓNICA		
DIRECTOR	CARLOS XAVIER ROSERO CHANDI		

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló sin violar derechos de autor de terceros, por lo tanto la obra es original, y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 4 días del mes de julio de 2023

Anderson Steeven Borrallos Pérez
C.I.: 100429912 – 7

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado "MÁQUINA DE ESTADOS FINITOS PARA CONTROL DE UN ROBOT SEGUIDOR DE LÍNEA", presentado por el egresado Anderson Steeven Borrallos Pérez, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, a los 4 días del mes de julio de 2023

Carlos Xavier Rosero
DIRECTOR DE TESIS

Agradecimiento

Quiero expresar mi profundo agradecimiento a mis padres, Juan Miguel Palacios y Gladis Azucena Pérez Díaz, por brindarme su apoyo moral e incondicional a lo largo de mi trayectoria universitaria. Ellos han sido mi pilar fundamental y su constante respaldo siempre será invaluable. A mi abuelito Augusto Pérez y a mis hermanos, Richard y Alan, por su apoyo inquebrantable en este importante camino que he recorrido.

Un agradecimiento especial a mi director de tesis, Carlos Xavier Rosero Chandi, quien ha sido un guía excepcional. Su paciencia, compromiso y profundos conocimientos han sido fundamentales para el desarrollo y éxito de este proyecto.

Por último, deseo expresar mi gratitud a la Universidad Técnica del Norte, que me ha dado la oportunidad de adentrarme en el maravilloso mundo de la ingeniería.

Anderson

Dedicatoria

Dedico este trabajo a mi familia, a mis amigos Javier, Alexander y a todas las personas, que de alguna u otra manera han formado parte de mi vida, durante este proceso universitario.

A Alan Borrallos por ser mi inspiración en esta profesión.

A María Espinosa por ser un apoyo y darme palabras de aliento.

Por último, dedico este trabajo a todos los ingenieros y mentes brillantes que, con su trabajo y dedicación, inspiran y demuestran el poder de la ingeniería para transformar el mundo.

Anderson

Resumen

La tecnología de robots móviles autónomos se ha actualizado para operar en áreas con escasa infraestructura ferroviaria, especialmente en industrias. Los robots seguidores de línea se utilizan en servicios de atención médica y otras aplicaciones de transporte o movimiento, por tanto, se busca mejorar su eficiencia en el control. Estos robots presentan problemas en cuanto a su escalabilidad y capacidad para corregir errores en una trayectoria marcada. Además, carecen de orden y análisis en su funcionamiento. Por esta razón, se utilizan sistemas de control en lazo cerrado en sus sensores o actuadores para controlar su desplazamiento y velocidad de forma simultánea y asegurar un rendimiento óptimo del robot en diferentes situaciones de trabajo. Con este fin, se desarrolló una metodología basada en una técnica robusta llamada máquina de estados finitos (FSM) que permitió separar el comportamiento del robot en diferentes situaciones mediante la definición de estados, eventos y acciones de control. Asimismo, se incluyó un algoritmo que utiliza cinco estados: ”Parado”, ”Avance”, ”Giros”, ”Retroceso” y ”Fin”, junto al método de modulación por ancho de pulso (PWM) para controlar la velocidad y ejecutar una acción de control específica para cada estado. La arquitectura del robot que se utiliza es diferencial y se compone de cuatro sensores infrarrojos y dos actuadores. Por último, se desarrolló un código de programación organizado que facilitó la depuración de errores de lógica y la puesta en marcha del robot. En las pruebas realizadas, se utiliza dos escenarios uno normal y uno complejo que tienen un inicio y un fin. Al analizar las pistas por tramos, se obtuvieron diversos resultados en términos de estabilidad, fluidez, tiempo de recorrido y velocidad. Estas pruebas demostraron un control altamente exitoso del robot seguidor de línea.

Palabras clave: robot, FSM, PWM, estados, programación.

Abstract

Autonomous mobile robot technology has been upgraded to operate in areas with poor rail infrastructure, especially in industries. Line follower robots are used in medical care services and other transport or movement applications, therefore, it is sought to improve their control efficiency. These robots present problems in terms of their scalability and ability to correct errors in a marked trajectory. In addition, they lack order and analysis in their operation. For this reason, closed-loop control systems are used in its sensors or actuators to control its displacement and speed simultaneously and ensure optimal performance of the robot in different work situations. To this end, a methodology based on a robust technique called finite state machine (FSM) was developed, which allowed to separate the robot's behavior in different situations by defining states, events and control actions. Likewise, an algorithm that uses five states was included: ”Stopped”, ”Forward”, ”Turns”, ”Reverse” and ”End”, together with the pulse width modulation method. (PWM) to control the speed and execute a specific control action for each state. The robot architecture used is differential and consists of four infrared sensors and two actuators. Finally, an organized programming code was developed that facilitated the debugging of logic errors and the start-up of the robot. In the tests carried out, two scenarios are used, one normal and one complex, which have a beginning and an end. When analyzing the tracks by sections, various results were obtained in terms of stability, fluidity, travel time and speed. These tests demonstrated a highly successful control of the line following robot.

Keywords: robot, FSM, PWM, states, programming.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1. Planteamiento del Problema

La tecnología de robots móviles autónomos que siguen trayectorias se ha actualizado para permitir el movimiento o transporte en áreas sin infraestructura ferroviaria principalmente en industrias. Estos robots se utilizan para servicios de atención médica y otras aplicaciones, por eso es importante controlarlos de una manera más eficiente. Los robots seguidores de línea pueden ser difíciles de escalar y corregir errores, y no tienen orden ni análisis. Hay muchas situaciones de trabajo en las que puede usar sistemas de control de lazo cerrado como en sus sensores o actuadores para garantizar la estabilidad [1].

Algunos estudios realizados acerca del comportamiento de los robots seguidores de línea presentan diversos problemas en cuanto al control autónomo que tienen en las trayectorias. Controlar el desplazamiento y la velocidad a la par de mantener la estabilidad, son retos que se pueden lograr a través de técnicas robustas ante la incertidumbre [2].

Una técnica robusta es la máquina de estados finitos (FSM, Finite State Machine) que permite separar el comportamiento de un robot en diferentes situaciones, ante las cuales se ejecutan diferentes acciones de control [3]. Es así que se propone controlar un robot seguidor de línea

mediante una FSM que permita a) analizar el comportamiento del sistema y ejecutar una acción de control para cada estado, b) identificar posibles fallas en sensores y actuadores y establecer acciones a prueba de fallos, y c) desarrollar código de programa ordenado que facilite la depuración de errores de lógica.

1.2. Objetivos

1.2.1. Objetivo General

- Desarrollar una máquina de estados finitos para un robot seguidor de líinea controlado en lazo cerrado.

1.2.2. Objetivos Específicos

- Diseñar la máquina de estados finitos considerando hardware, fuentes de incertidumbre y situaciones de funcionamiento del robot.
- Establecer los lazos de control para cada estado de funcionamiento considerando la dinámica del robot.
- Evaluar las estrategias de control implementadas sobre un robot móvil real.

1.3. Justificación

La aplicación de FSM dentro de la ingeniería de control permite estructurar el comportamiento de un sistema y establecer estrategias de control para cada situación. Esto conduce a una programación estructurada y directa ante cada situación.

En la ingeniería mecatrónica el control de sistemas se asocia tradicionalmente al control de prototipos electrónicos y/o robóticos, y máquinas automatizadas que requieren diferentes estados y procesos de funcionamiento. Se utilizan FSM para automatizar dispositivos individuales, robots autómatas, así como máquinas complejas.

A través de una FSM se adiciona confiabilidad a los sistemas de control al mantener un rendimiento aceptable incluso ante fallas. Un robot seguidor de línea que muestre un comportamiento adecuado a cada etapa de funcionamiento dentro de su trayectoria aumentará su confiabilidad y rendimiento.

1.4. Alcance

Se desarrollará una máquina de estados finitos de uso exclusivo en este robot seguidor de línea. Sin embargo, se considera que podría ser aplicada de manera general en otros robots de la misma naturaleza.

Se analizarán los diferentes escenarios en los que trabaja el robot para diseñar la FSM y las condiciones de guarda y de salto de cada estado. Posteriormente, para cada estado se diseñará una estrategia de control continuo en base a la dinámica del robot. El rendimiento del control discreto (FSM) y del control continuo (estrategias en cada estado), funcionando de manera híbrida, se probará en primera instancia a través de software matemático. Posteriormente, se validará el diseño a través de la implementación sobre un robot real.

Capítulo 2

Revisión Literaria

2.1. Estado del Arte

Un estudio realizado por un grupo de estudiantes de la Universidad de Patras [4], se ha evidenciado que las máquinas de estados finitos (FSM) detectan incertidumbres de medición que causan los sensores infrarrojos a la hora de establecer la lectura de datos correspondiente en la trayectoria del robot seguidor de línea, para eso se presenta una FMS que estima la posición del robot en relación a la ruta deseada detectando mediciones ruidosas en los sensores IR y cambiando la propuesta entre un controlador PD de ganancia variable y un controlador de lazo abierto para controlar el robot cuando pierda la pista, lo que hace este método una buena opción para trabajar con el robot en tiempo real.

Así mismo, un trabajo desarrollado por R. Balogh y D. Obdrzalek [5] ayuda al entendimiento del uso de una FSM en la robótica, especialmente para implementar sistemas de control de

nivel de entrada en donde se presentan y se muestra un esquema de implementación en varios ejemplos de tareas reales en donde se usa las FSM y tiene como objetivo mostrar que puede ser una buena herramienta para implementar el sistema de un robot, además de ser potente y fácil de utilizar.

Por otro lado, una investigación presenta las FSM como una metodología que describe el comportamiento del sistema usando tres cosas, a saber: estado, evento y acción; en un programa, el sistema estaría en un estado activo. El sistema puede cambiar o moverse a otro estado si recibe cierta entrada o evento. Se implementa una FSM basada en control PID (Proporcional-Integrativo-Derivativo) para que el robot siga la trayectoria de la línea. Se usa 6 sensores infrarrojos que están posicionados estratégicamente de forma casi lineal frente al chasis para establecer los eventos con lógica binaria en el que se puede encontrar el robot y así ajustar la velocidad adecuada por modulación en ancho de pulso (PMW) que se envía al driver de los motores para que exista un avance controlado. El resultado de la prueba muestra que el algoritmo diseñado puede funcionar bien y puede usarse como un algoritmo basado para el robot de este trabajo [6]. De igual forma en [2] se realiza una comparación entre diferentes algoritmos de control: un algoritmo PID, control por estados, control proporcional destacándose el primero con gran desempeño en cuanto al tiempo de recorrido, velocidad, fluidez y estabilidad en la trayectoria.

Por último, en [7] la navegación de robots móviles que requieren resolver problemas de trampas locales en la estrategia tradicional, proponen un algoritmo para robots móviles basado en un anillo de sonar, para realizar la navegación en un entorno desconocido y complejo. Se presenta un mecanismo de evasión de obstáculos basado en control de lógica difusa y seguimiento

de paredes sobre la base de modelos construidos a partir de datos de sonar. Usando FSM, el estado de navegación del robot móvil cambia en función de la información del entorno, y se elige la estrategia correspondiente para realizar la tarea de navegación.

2.2. Tecnologías Empleadas

En este capítulo, se presentan de manera breve algunas definiciones importantes de tecnologías relevantes para la construcción de un robot seguidor de línea. Además, se aborda algunos conocimientos para el diseño adecuado de una máquina de estados finitos (FSM).

2.2.1. Microcontroladores para los Robots Móviles

Los microcontroladores son dispositivos electrónicos integrados capaces de realizar tareas específicas, como controlar sistemas embebidos, procesar señales y ejecutar programas de control en tiempo real. Estos se utilizan en una amplia gama de aplicaciones, desde electrodomésticos hasta maquinaria industrial. Los microcontroladores tienen la capacidad de controlar prácticamente cualquier dispositivo compatible con sus entradas y salidas digitales/análogas, lo que los convierte en la elección perfecta para trabajar con prototipos robóticos, como es el caso de los robots seguidores de línea.

2.2.2. Robots Móviles Autónomos

Un robot móvil autónomo es un dispositivo que tiene la capacidad de navegar de forma independiente por un espacio. En su configuración más común, cuenta con un accionamiento diferencial, que consiste en dos motores instalados lateralmente en su armazón y accionados de forma independiente. Cada motor se encarga del giro de una de las ruedas motrices, mientras que un tercer punto de contacto es necesario para garantizar la estabilidad del robot, implementando una rueda libre en la parte trasera del modelo. La robótica móvil se puede categorizar en dos formas principales, teniendo en cuenta su característica de autonomía: los robots móviles guiados automáticamente (AGV, por sus siglas en inglés, Automated Guided Vehicle) y los robots móviles autónomos (AMR, por sus siglas en inglés, Autonomous Mobile Robot) [8].

2.2.3. Robot Seguidor de Línea

Es un robot móvil que tiene como objetivo seguir una trayectoria marcada por una pista de color negro sobre una superficie blanca (o cualquier otro color que contraste adecuadamente). Para lograrlo, se hace hincapié en la detección de la pista mediante sensores infrarrojos (IR) y en el uso de motores para su movimiento [9]. En la figura 2.1 se muestra un ejemplo de este tipo de robot y se observa su funcionamiento.

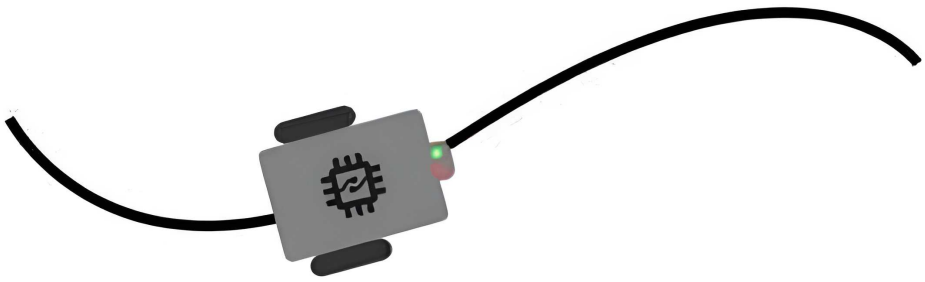


Figura 2.1: Funcionamiento de un Robot Seguidor de Línea [9].

2.2.4. Hardware de un Robot Seguidor de Línea

A continuación, se muestra de forma concisa la arquitectura que conforma un robot seguidor de línea.

2.2.4.1. Fuente de Alimentación

El suministro de energía eléctrica es un elemento crucial para el funcionamiento del robot. Es responsable de alimentar tanto al microcontrolador como a todos los dispositivos del robot. Las tensiones de trabajo necesarias pueden variar dependiendo del fabricante, pero generalmente oscilan entre 5 y 12 voltios en corriente continua. Garantizar un suministro eléctrico adecuado es fundamental para un rendimiento óptimo del robot seguidor de línea.

2.2.4.2. Tarjeta Programadora

El algoritmo de una máquina de estados finitos (FSM) se basa principalmente en la programación de estados. Para llevar a cabo esta programación, se utilizan tarjetas de programación,

que son dispositivos diseñados para conectar elementos de hardware, como sensores y actuadores, a un microcontrolador. Estas tarjetas suelen programarse en código abierto, utilizando un software y lenguaje específico. La conexión entre el ordenador y la tarjeta puede establecerse mediante una conexión inalámbrica o serial [10].

Existen diversas opciones de tarjetas de programación, siendo Arduino y Raspberry Pi ejemplos destacados. En el caso de un robot seguidor de trayectorias, el uso de un Arduino Uno resulta idóneo.

2.2.4.3. Sensor Infrarrojo

Estos sensores, conocidos como sensores ópticos, se componen principalmente de un diodo LED que emite luz infrarroja (3-4) y un fototransistor (1-2) encargado de detectar dicha luz. El fototransistor permite el paso de corriente entre el colector y el emisor según la luminosidad del entorno [2]. En el caso de un robot seguidor de línea, se utilizan estos sensores para detectar y seguir una trayectoria marcada en una superficie negra o blanca obteniendo valores analógicos o digitales según sea su configuración. La figura 2.2 muestra la estructura interna de un sensor infrarrojo.

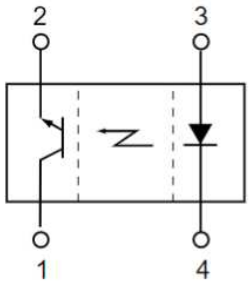


Figura 2.2: Estructura de un sensor infrarrojo [2].

2.2.4.4. Digitalizador Análogo

Es un tipo de convertidor A/D (analógico-digital). En el caso de los robots seguidores de línea, se utiliza un circuito operacional como el LM339N, este posee 4 comparadores. La señal analógica recibida de los sensores infrarrojos se comparara con otro voltaje de referencia y dependiendo de su configuración se obtiene en su salida una señal digital. Esta señal digital facilita su interpretación en la programación de control [2].

2.2.4.5. Driver para Motores

La velocidad de los motores se controla mediante estos drivers, se utilizan para suministrar la potencia adecuada a través de señales de modulación por ancho de pulso (PWM). Estas señales son generadas por el microcontrolador y permiten ajustar la potencia de forma analógica, utilizando valores que van de 0 a 255 en la programación, esto debido a que el microcontrolador se basa en 8 bits de resolución. En la construcción de una máquina de estados finitos (FSM), este elemento adquiere una importancia fundamental, ya que el algoritmo se aplica directamente a los actuadores [6]. Algunos drivers ya vienen estandarizadas como la placa Motor Shield L298P que es compatible con Arduino. Sus características son: operación de trabajo de 5v - 12v, controla dos motores de corriente continua o un motor de pasos, la potencia es de 2 a 4 Amperios.[1]

[1]El robot a utilizar ya cuenta con esta placa estandarizada ”Motor Shield L298P” y un Arduino Uno en su arquitectura.

2.2.4.6. Motores DC

Un motor de corriente continua, es un tipo de actuador diseñado específicamente para proporcionar velocidad de rotación de un eje. Este motor permite transformar esta energía mecánica suministrando energía eléctrica DC. Está compuesto por algunos elementos como un estator, rotor y bobinas, y en algunos casos junto a una variedad de componentes adicionales, como controladores y engranajes para tener diferentes configuraciones de velocidad-torque [8].

2.2.5. Formas de Control de Robots Seguidores de Línea

Para controlar robots móviles autónomos existen diversas técnicas y enfoques. Una de ellas es el control por estados secuenciales (fsm), que se basa en situaciones específicas del robot y su entorno, utilizando transiciones entre estados para lograr un control adaptativo. Otra técnica común es el controlador PID (Proporcional, Integral, Derivativo), que ajusta los errores del sistema utilizando tres parámetros clave para obtener un control preciso y estable. También se encuentra el control por error proporcional, que se enfoca únicamente en el error proporcional para realizar correcciones. Cada enfoque tiene sus ventajas y se puede elegir según las necesidades y características del robot en cuestión. [8]. Además de estos, también existe métodos más sofisticados usando visión artificial.

2.2.6. Modulación por Ancho de Pulso (PWM)

La modulación por ancho de pulso (PWM) se utiliza para el control de velocidad en motores DC. Se forman a partir de señales eléctricas pulsantes con una frecuencia constante la cual es producida por salidas digitales únicas de un microcontrolador. El driver de motores depende mucho de este voltaje para suministrar la velocidad correspondiente a la que rotará el eje del motor. Es decir que mientras más voltaje analógico exista, más rápida será la velocidad de los motores [11].

En el caso del microcontrolador Atmega328p, su procesador cuenta una resolución de 8 bits. Esto significa, que tiene un rango máximo de 255 en número decimal para representar la señal de 5 voltios. La tabla 2.1 y la figura 2.3 presentan algunas configuraciones disponibles en relación al voltaje de salida y el porcentaje de trabajo correspondiente. Estas configuraciones permiten ajustar la velocidad de los motores de acuerdo con las necesidades específicas del sistema.

Tabla 2.1: Tabla comparativa entre el valor analógico con respecto al % de trabajo

Voltaje(v)	Valor Analógico	Porcentaje de Trabajo
5	255	100 %
4	204	80 %
3	153	60 %
2	102	40 %
1	51	20 %
0	0	0 %

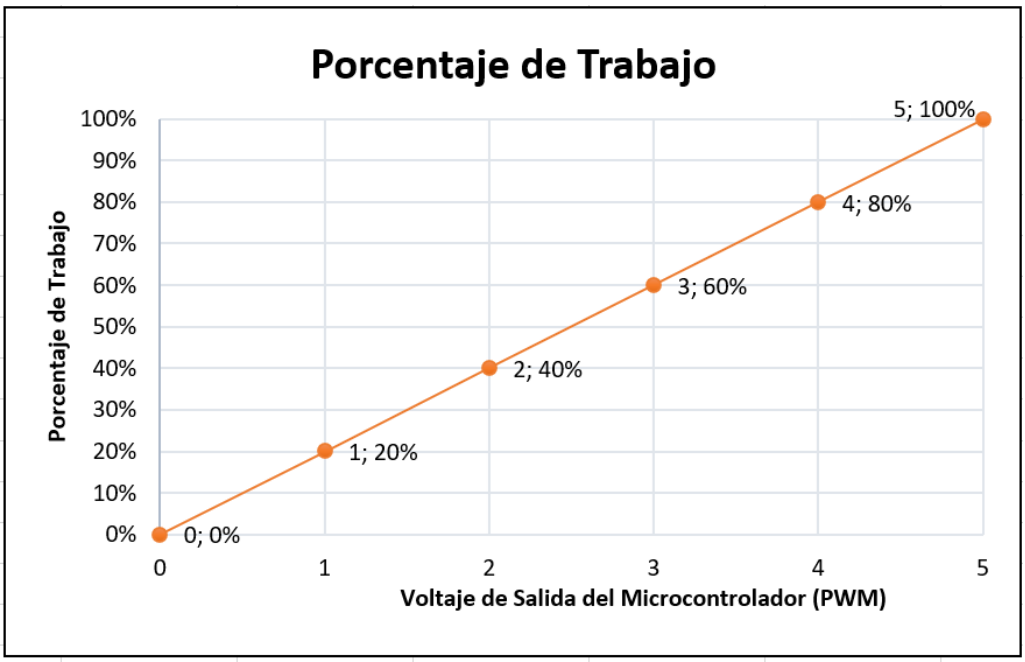


Figura 2.3: Gráfica comparativa.

2.2.7. Máquinas de Estados Finitos

Una Máquina de Estados Finitos (FSM) es una herramienta útil para el diseño de sistemas de control. Esta permite modelar sistemas mediante la definición de un conjunto finito de estados, eventos y transiciones entre dichos estados. Al considerar una máquina de estados para el control de un robot autónomo, se puede concebir en su capacidad para detectar cambios en su entorno y reaccionar en consecuencia. [4].

Esta FSM se la interpreta de la siguiente manera:

$$M = (S, \sum, A, sk), \tag{2.1}$$

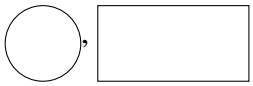
donde

$S = \{S_1, S_2, ..., S_m\}$, son los estados que posee el sistema. $\sum$, corresponde al número de etiquetas correspondiente a un alfabeto infinito. A, son las transiciones del sistema. y sk, corresponde al estado inicial del sistema.

2.2.7.1. Estructura de una FSM

Las máquinas de estados finitos poseen la siguiente estructura general básica:

Tabla 2.2: Estructura de las máquinas de estado finitas

Elemento	Símbolo	Definición
Estado inicial		Estado de inicio
Estados		Representa la situación actual del sistema.
Eventos	condition	Son las condiciones de guarda y salto de cada estado
Transiciones		Son las direcciones de cambio de estado

2.2.7.2. Funcionamiento y Creación de la FSM

La creación de una FSM implica los siguientes pasos: identificar el número de entradas y salidas del sistema, definir el conjunto de estados y determinar las transiciones entre ellos. A continuación, se crea un diagrama de estados para visualizar el sistema de manera clara. La operación de la FSM se describe de la siguiente manera: cuando se recibe una señal de entrada, se genera una excitación. A partir de ahí, se procesan sucesivamente los diferentes estados en

función de las condiciones (eventos) y las transiciones de cambio de estado. Como resultado de este proceso, se genera una salida que corresponde a la respuesta a la excitación recibida [12].

Estas FSM se las utiliza para realizar un proceso en corto tiempo, optimizando las secuencias de trabajo que posea el dispositivo robótico y permitiendo una buena síntesis del sistema. A su vez, se obtiene una buena lógica de programación que ayude al funcionamiento idóneo del mismo [13]. Una FSM en un robot móvil autónomo puede permitir lo siguiente:

– Facilitar la capacidad del robot para responder a diversas entradas.

– Permitir al robot adaptarse y responder a cambios en su entorno.

– Representar interacciones y dependencias complejas entre las entradas y salidas del robot.

– Generar múltiples acciones basadas en un mismo evento de entrada.

2.2.7.3. Diagrama de Estados

Es fundamental realizar un diagrama de estados, antes de diseñar una máquina de estados finitos (FSM) en software, ya que esto contribuye a una mejor comprensión de cómo implementarla en base a los parámetros de la FSM. Así, se crea un modelo del comportamiento del sistema que considera tanto el entorno como los criterios específicos del mismo [14]. En la figura 2.4 se presenta un ejemplo sencillo de un digrama FSM, donde los 5 estados se representan mediante círculos, las transiciones se muestran con flechas y los eventos se identifican mediante letras.

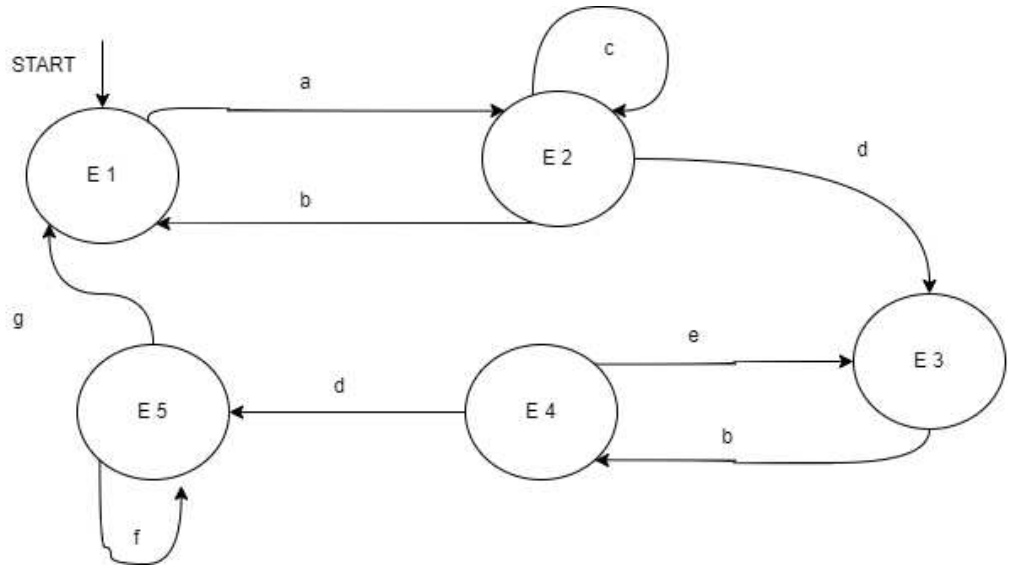


Figura 2.4: Diagrama de cinco estados

2.2.7.4. Tipos de FSM

Al diseñar sistemas de control basados en una Máquina de Estados Finitos (FSM), es crucial considerar las definiciones de dos tipos de FSM que permiten establecer distintas aplicaciones según la configuración de las entradas, salidas y estados. Estos tipos de FSM desempeñan un papel fundamental en el diseño de sistemas de control eficientes, ya que proporcionan un marco estructurado para la representación y gestión del comportamiento secuencial. A través de esta comprensión, se logra un diseño sólido y adaptado a las necesidades específicas de control en diferentes aplicaciones [15].

Máquina de Mealy Es una máquina que produce una salida dependiendo de su estado actual y una entrada de excitación. La máquina de Mealy es la más utilizada debido a su versatilidad y funcionamiento se puede observar en la figura 2.5 cómo es su estructura interna. [15].

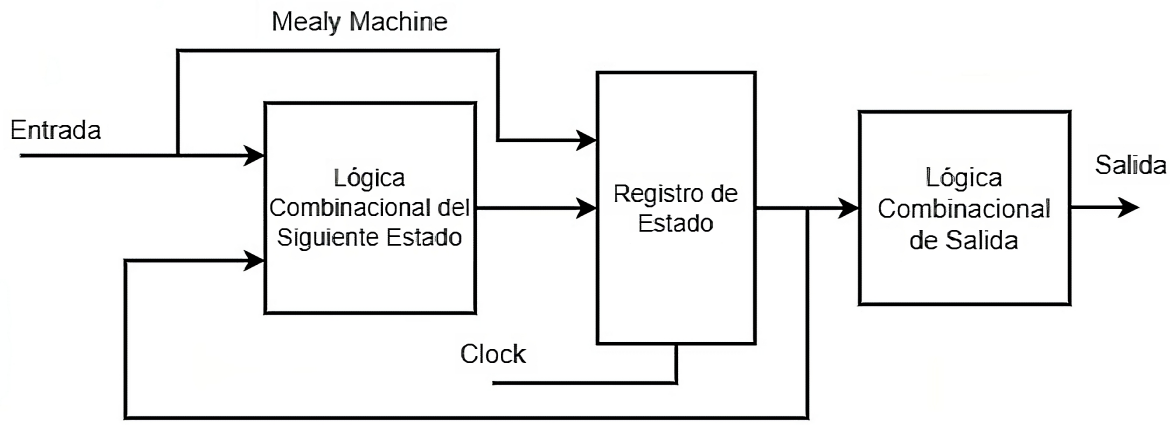


Figura 2.5: Diagrama de bloques general de la máquina de Mealy.

Máquina de Moore Es una máquina que depende solo de su estado actual para producir una salida [15]. La estructura interna de una máquina de Moore se representa de la siguiente manera, figura 2.6.

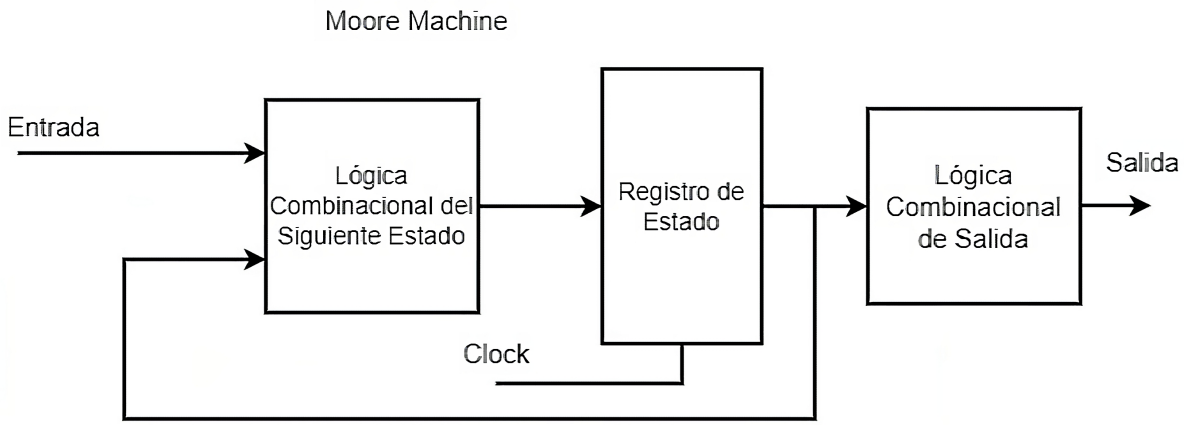


Figura 2.6: Diagrama de bloques general de la máquina de Moore

2.2.7.5. Software matemático para implementación de una FMS

Las FMS requieren de una previa simulación antes de su implementación sobre el prototipo. Matlab es una buena opción a la hora de trabajar con simulaciones utilizando el entorno Simulink. Para la creación de las máquinas de estados, Simulink consta con muchas librerias dentro de su interfaz como Stateflow.

Simulink El entorno de Simulink en Matlab ofrece la posibilidad de realizar diseños de sistemas dinámicos de forma gráfica mediante bloques configurables. Esta herramienta es utilizada en el campo de la robótica para simular y probar modelos conceptuales antes de implementarlos en el mundo real, lo que ayuda a reducir errores y optimizar el rendimiento. La plataforma de Simulink no solo permite la simulación de sistemas, sino que también proporciona la capacidad de generar código en varios lenguajes de programación, como C/C++, Python y Java. Esto significa que los algoritmos de control pueden ser creados y desarrollados en estos lenguajes, lo que brinda flexibilidad y facilidad para implementar soluciones de control en diferentes entornos [13].

La capacidad de generar código en varios lenguajes de programación es especialmente útil cuando se trabaja con robots y sistemas embebidos, ya que permite la integración con el hardware y la implementación de algoritmos de control en tiempo real.

Stateflow Stateflow es una herramienta esencial en Matlab para diseñar, modelar y simular máquinas de estados finitos en sistemas dinámicos. Su enfoque gráfico interactivo y su amplia variedad de librerías facilitan la representación y análisis de diagramas de flujo, diagramas de

estado y tablas de verdad. En la figura 2.7 se observa las diferentes opciones de trabajo que esta herramienta nos ofrece. Stateflow utiliza bloques gráficos interactivos que representan los elementos de una FSM. Estos bloques facilitan la creación de diagramas de estados intuitivos y comprensibles, lo que ayuda a visualizar y analizar el comportamiento de un sistema [13].

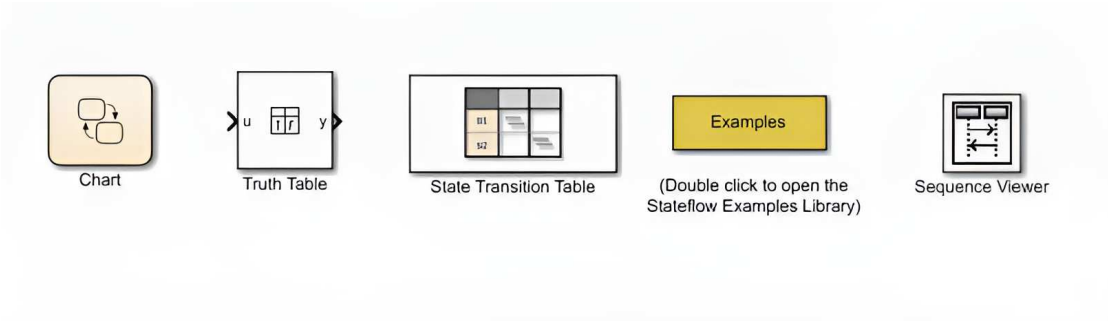


Figura 2.7: Librerías de Stateflow

La simulación de una máquina de estados finitos en Stateflow permite probar y verificar el comportamiento del sistema antes de su implementación en el prototipo real. Esto reduce el riesgo de errores y ayuda a optimizar el rendimiento del sistema.

Una característica primordial que ofrece esta herramienta, es generar un código en C portable.

Capítulo 3

Desarrollo

En este capítulo, se describe el proceso de diseño para crear una máquina de estados finitos (FSM) basada en un algoritmo de control, específicamente diseñada para un robot seguidor de línea.

3.1. Requerimientos del sistema

El sistema debe cumplir con una serie de requerimientos para asegurar su correcto funcionamiento en el contexto de una aplicación de transporte con trayectoria continua. Los escenarios de prueba deben contar con un punto de inicio y un punto final claramente definidos y sin cruces, lo que permitirá establecer una ruta precisa y consistente. Para abordar este objetivo, se propone la implementación de cinco estados de trabajo que se ajusten a las necesidades específicas de la aplicación, priorizando el control y la estabilidad del sistema en lugar de la velocidad. Por último la disposición de los sensores debe ser lineal, con cuatro sensores colocados a una medida

constante entre ellos.

3.2. Entradas y Salidas

Para iniciar en la creación de una Máquina de Estado Finito (FSM), es necesario definir las diferentes variables de entrada y salida del sistema. En la figura 3.1, se pueden observar estas variables claramente. En este caso, el sistema cuenta con un pulsador (SW) para activar el sistema, 4 sensores infrarrojos que están diseñados para detectar diferentes eventos a lo largo de la trayectoria del robot. Estos sensores trabajan con lógica binaria simple gracias al digitalizador análogo, es decir, cuando estos detectan una superficie negra, su valor es ”0”, mientras que si detectan una superficie blanca, su valor es ”1”. Es importante destacar que estos valores son independientes.

Por otro lado, las variables de salida corresponden a los actuadores del sistema. En este caso, se utilizan dos motores que permiten el avance del robot. Estos actuadores se controlan en base a las variables de entrada mencionadas anteriormente.Todas estas variables, tanto de entrada como de salida, juegan un papel fundamental en la definición del algoritmo de control que tiene el robot en su respectiva programación.

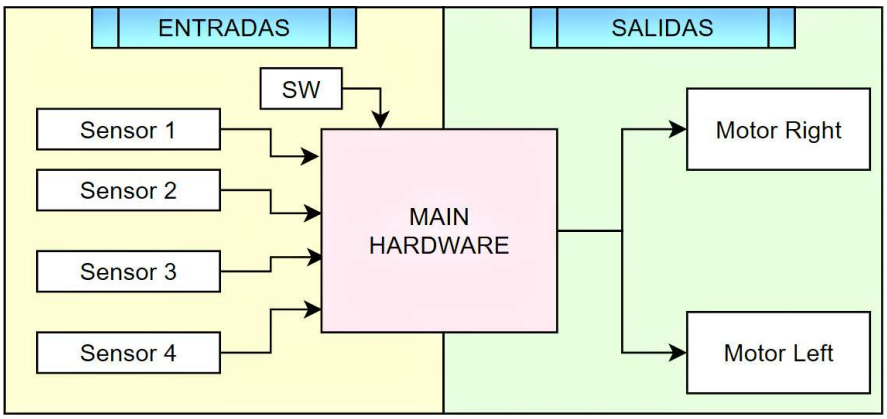


Figura 3.1: Entradas y salidas del sistema.

3.3. Estados del Robot

El robot se divide en 5 estados de trabajo, los cuales se representan de manera clara en un diagrama que muestra las transiciones entre ellos. Para comprender mejor el sistema, se puede hacer referencia a la figura 3.2.

Los estados son llamados de la siguiente manera:

– Estado ”Parado”: En este estado, el robot se encuentra encendido, pero en reposo, y listo para comenzar su funcionamiento.

– Estado ”Avance”: Al presionar el botón de ”Start”, el robot iniciará su movimiento hacia adelante.

– Estado ”Giro”: Este estado se activa cuando el robot se enfrenta a curvas más cerradas en su recorrido,

– Estado ”Retroceso”: Si el robot sale de la trayectoria establecida, entrará en el estado de retroceso. En esta situación, el robot invertirá su dirección y se moverá hacia atrás para corregir

su posición, realizando pequeños movimientos serpenteo.

– Estado ”Fin”: Este estado se alcanza cuando el robot ha llegado a su destino final. Indica que el recorrido se ha completado exitosamente y que el robot ha cumplido su tarea.

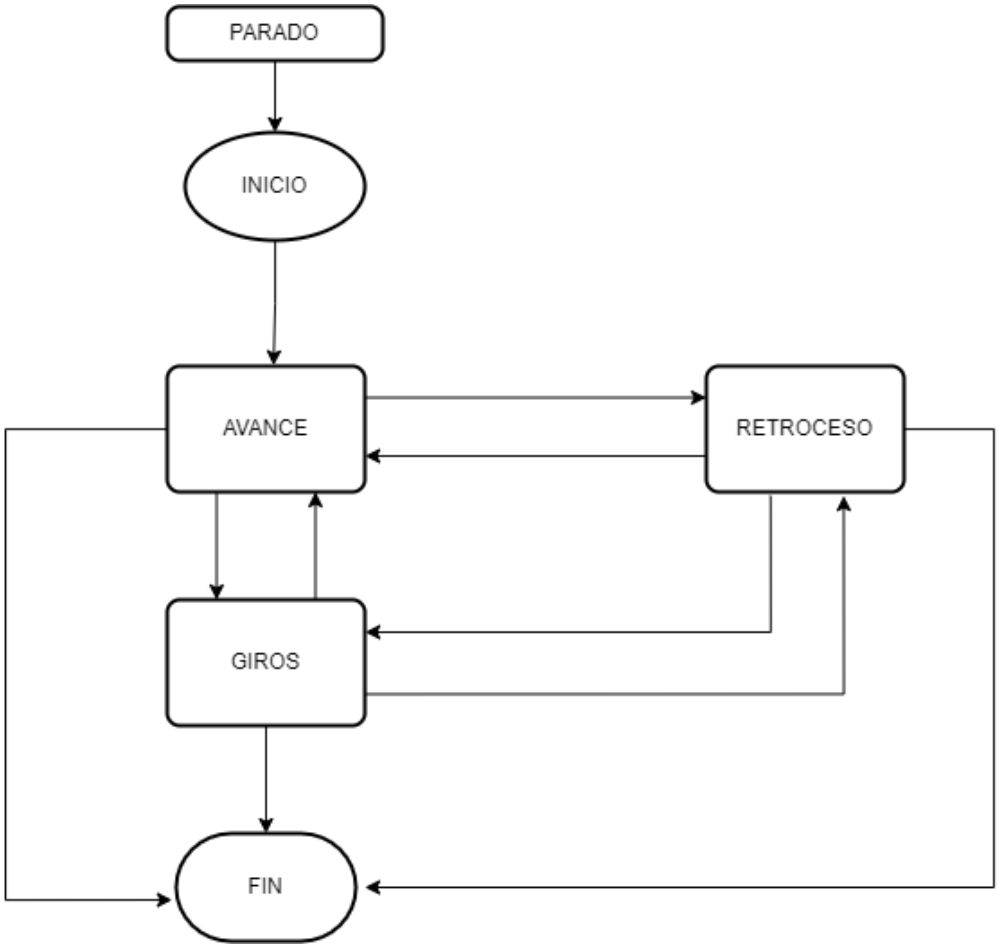


Figura 3.2: Estados de trabajo.

3.4. Algoritmo FSM

En la figura 3.3 se observa el algoritmo que se implementa en el robot seguidor de línea con sus respectivos 5 estados, 9 eventos y 9 acciones de control. Este algoritmo FSM se utiliza con el propósito de mejorar la posición del robot y lograr un control más estable.

ALGORITMO DE LA MÁQUINA DE ESTADOS FINITOS			
ESTADO	**EVENTO**		**ACCIÓN**
PARADO	Inactivo		Stop
AVANCE	1	1100	REV I > REV R
	2	1110	REV I >> REV R
	3	1001	REV I = REV R
	4	0111	REV I << REV R
	5	0011	REV I < REV R
GIROS	6	0001	MOTOR I = OFF MOTOR R = ON
	7	1000	MOTOR I = ON MOTOR R = OFF
RETROCESO	8	1111	MOTOR I Y R= ON
FIN	9	0000	MOTOR I Y R = OFF

Figura 3.3: Algoritmo FSM.

El estado inicial es **”Parado”**, donde el robot se encuentra encendido, en posición de espera hasta recibir una señal proveniente de un botón de ”start”. Esta señal desencadena el cambio de estado, preparándolo para el siguiente.

La figura 3.4 muestra claramente el funcionamiento del estado **”Avance”**, el cual se activa en diferentes eventos y desencadena acciones específicas.

El evento 1 ocurre cuando dos sensores se encuentran fuera del lado izquierdo de la trayectoria (superficie blanca), mientras que los otros dos sensores permanecen en la superficie negra. En este caso, se inicia la acción correspondiente que implica aumentar el porcentaje de trabajo o el valor de PWM en el motor izquierdo y disminuirlo en el motor derecho **(a)**.

El evento 2 se activa cuando tres sensores se encuentran fuera de la trayectoria en el lado izquierdo, y el sensor restante permanece en la línea negra. En esta situación, se realiza la acción correspondiente que implica aumentar el valor de PWM del motor izquierdo, pero disminuir aún más el motor derecho **(b)**.

El evento 3, conocido como el ”ideal”, ocurre cuando los dos sensores centrales están perfectamente alineados con la línea de seguimiento. En este caso, ambos motores deben tener el mismo porcentaje de trabajo **(c)**.

El evento 4 se activa cuando un sensor se encuentra en la línea de seguimiento y tres sensores en el lado derecho están sobre la superficie blanca. En esta situación, se aumenta el porcentaje de trabajo en el motor derecho y se disminuye considerablemente en el motor izquierdo **(d)**.

Por último, el **El evento 5** se activa cuando dos sensores están en la línea y los otros dos en el lado derecho sobre la superficie blanca. En este caso, se incrementa el porcentaje de trabajo en el motor derecho y se reduce en el motor izquierdo **(e)**.

Cada evento específico desencadena una acción de control precisa, para mantener alineado a la trayectoria de color negro, buscando siempre la máxima fidelidad en su seguimiento con el evento ideal, identificado como número 3.

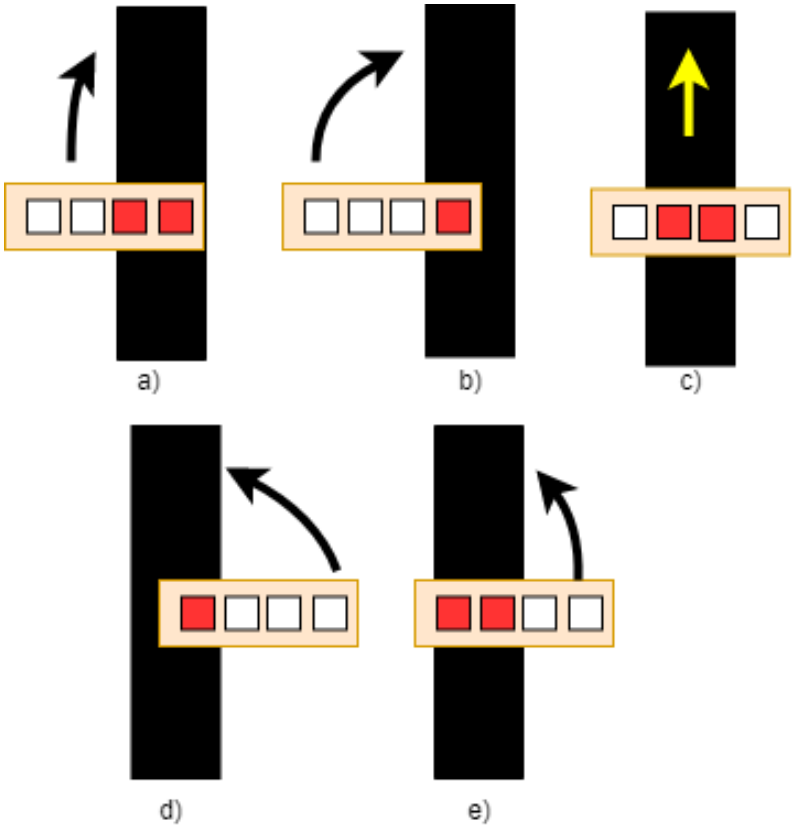


Figura 3.4: Eventos del estado ”Avance”.

El estado **"Giros"** incluye dos eventos específicos, los cuales se representan en la figura 3.5:

Evento número 6: Se activa cuando tres sensores del lado izquierdo están en la línea de seguimiento, mientras que un sensor se encuentra fuera de ella. En esta situación, se realiza la acción de encender el motor derecho y apagar el motor izquierdo, lo que resulta en un giro hacia la izquierda **(f)**.

Evento número 7: Se activa cuando los tres sensores del lado derecho están sobre la línea negra, mientras que el sensor restante se encuentra en la superficie blanca. En esta ocasión, se ejecuta la acción de encender el motor izquierdo y apagar el motor derecho, permitiendo que el robot gire hacia la derecha **(g)**.

Estas acciones de control específicas en los eventos 6 y 7 del estado "Giros" permiten al robot realizar giros precisos hacia la izquierda o la derecha, respectivamente.

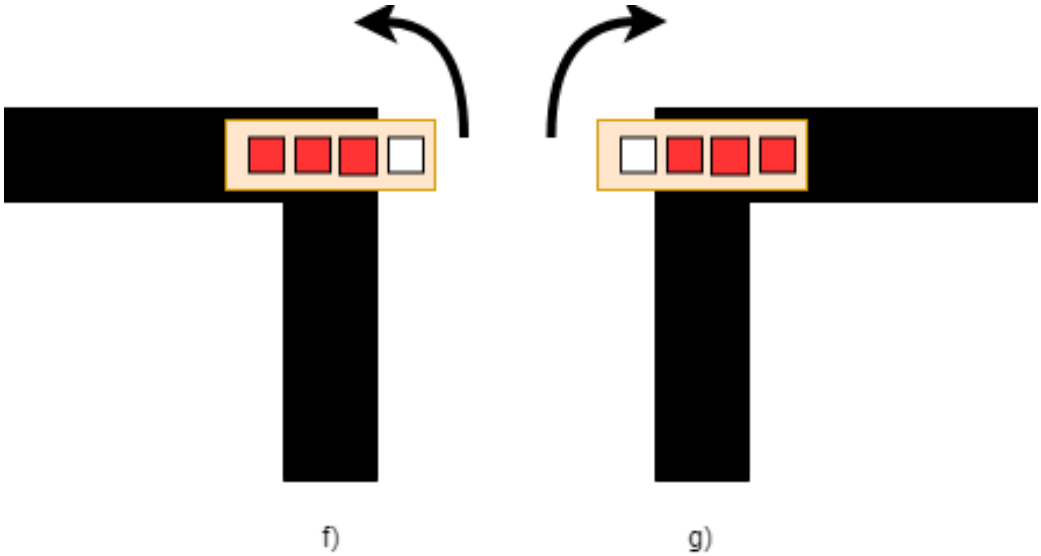


Figura 3.5: Eventos del estado "Giros".

El estado ”Retroceso” se activa cuando todos los sensores detectan la superficie blanca. En este estado, se realiza la acción de invertir el sentido de giro de los motores y mantenerlos encendidos, como se muestra en el literal h) de la figura 3.6. Este estado se considera como una medida de seguridad para evitar salirse de la trayectoria establecida.

Por otro lado, el estado ”Fin” indica el final de la trayectoria y se representa en el literal i) de la figura 3.6. Se activa cuando todos los sensores detectan la superficie negra. En este estado, la acción consiste en detener los motores, poniéndolos en modo ”OFF”.

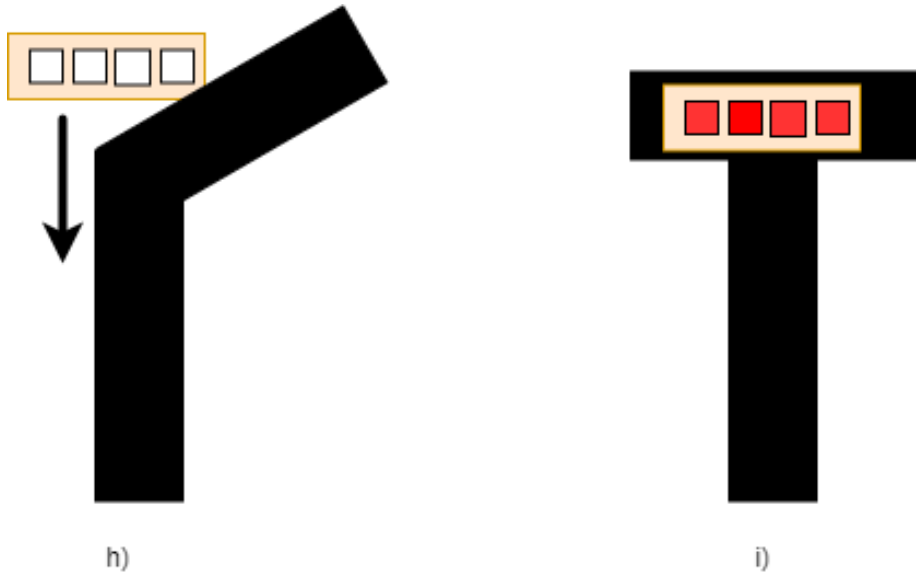


Figura 3.6: Estados ”Retroceso” y ”Fin”.

3.5. Diagrama de Estados

Se crea un diagrama de estados para el robot seguidor de línea, el cual tiene en cuenta las condiciones de guarda y salto entre cada estado, junto con las acciones de control correspondientes. Este diagrama refleja fielmente el algoritmo presentado en la sección anterior, y se presenta en la figura 3.7. El diagrama de estados proporciona una representación visual clara y concisa de las interacciones entre los diferentes estados del robot, permitiendo una comprensión más completa del funcionamiento del sistema.

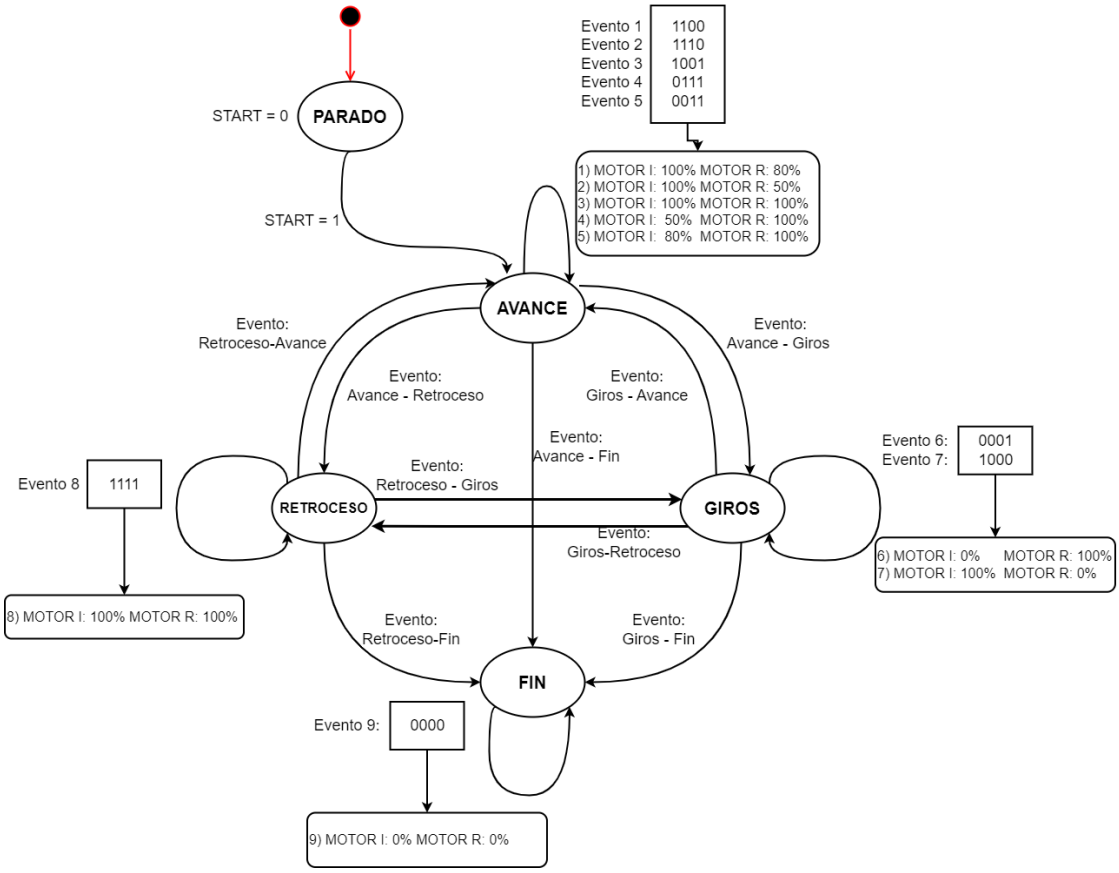


Figura 3.7: Diagrama FSM del robot seguidor de línea.

En el diagrama anterior, se observa que cuando el robot está en funcionamiento, los estados ”Avance”, ”Giros” y ”Retroceso” tienen transiciones tanto de entrada como de salida entre ellos. Estas transiciones permiten un progreso seguro del robot a lo largo de la trayectoria, asegurando un seguimiento adecuado y evitando desviaciones no deseadas.

Sin embargo, es importante destacar que el estado ”Fin” tiene un papel especial en el diagrama. Este estado indica que la trayectoria ha llegado a su fin y no se activarán transiciones hacia los estados anteriores hasta que se reinicie el robot con el botón de ”start”. Esto garantiza que el robot no reanude la trayectoria desde un punto anterior sin una nueva señal de inicio, evitando posibles errores o inconsistencias en su desempeño.

3.6. Simulación de la Máquina de Estados

La FSM que se menciona en el diagrama de estados finitos de la figura 3.7, requiere una simulación previa en un software en donde se analice lo que sucede en cada estado y en los eventos correspondientes, además que acciones de control toma el sistema. En la figura 3.8 muestra el diseño FSM en Matlab, utilizando la herramienta stateflow en el entorno simulink.

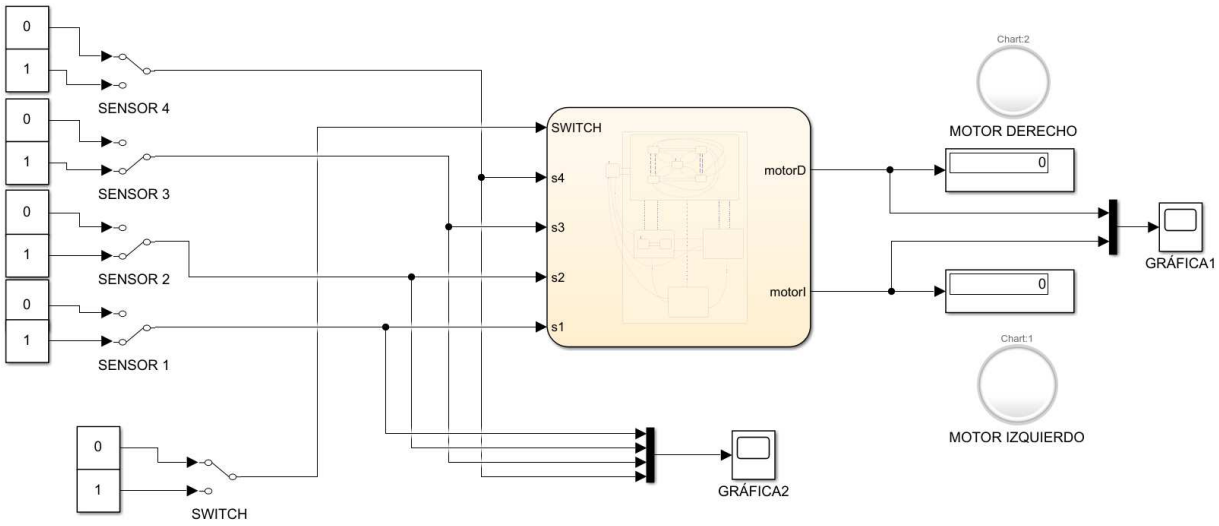


Figura 3.8: Diseño de la FSM.

En el diseño, en el lado izquierdo, se utilizan constantes de ”0” y ”1” en las entradas para simular la lógica de estado cuando los sensores infrarrojos detectan una de las dos superficies y adquieren la señal correspondiente. Estas entradas se dirigen hacia un ”manual switch” que representa el cambio manual de la señal digital. A su vez, estas entradas ingresan al diagrama de estados en la parte central. Por último, se utiliza una señal digital llamada ”switch” para encender el sistema.

En la parte derecha del sistema, las salidas se representan mediante dos ”display” que muestran los valores numéricos del PWM de cada motor. Además, se utilizan dos ”charts” que permiten visualizar gráficamente los valores PWM de los motores. Estos gráficos se presentan con tres colores distintos para indicar los niveles de intensidad:

El color verde representa un valor de PWM de 255.

El color azul representa un valor de PWM de 200.

El color rojo representa un valor de PWM de 114.

Esta codificación de colores facilita la interpretación visual de los niveles de intensidad y proporciona una representación clara de los valores PWM para la velocidad de los motores en el sistema.

En la parte central del sistema se encuentra el diseño del diagrama de estados finitos. Este diagrama se lo puede encontrar en el siguiente repositorio: `https://github.com/Anderson2399/FSM.git`. El archivo se denomina ”fsm.pdf”.

Para comprobar el funcionamiento de la máquina de estados finitos, se lleva a cabo una simulación del programa durante 5 segundos. Como resultado de esta simulación, se obtienen las siguientes gráficas:

3.6.1. Estado ”Parado”

No realiza alguna acción.

3.6.2. Estado ”Avance”

Evento 1 En la figura 3.9 se observa una salida en valores PWM de 200 el motor derecho y 255 el motor izquierdo.

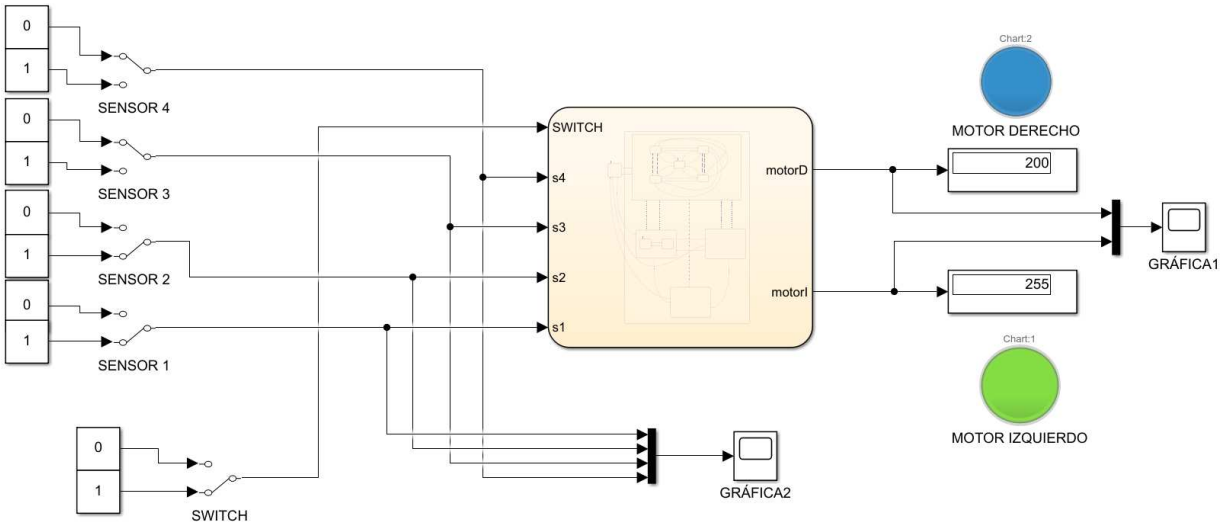


Figura 3.9: Evento 1.

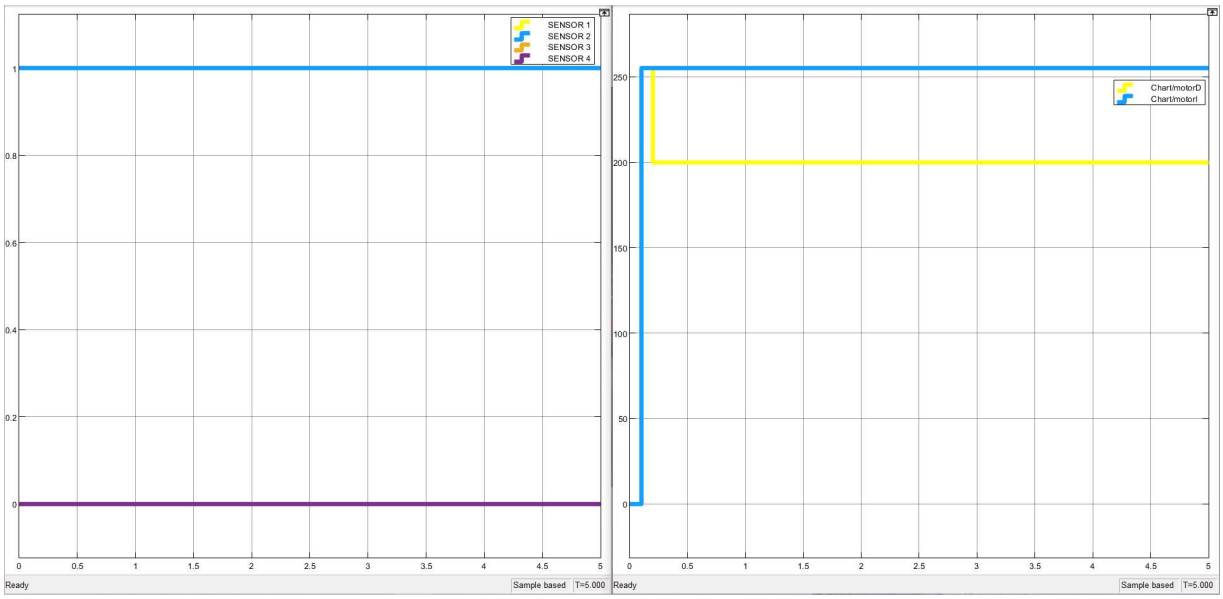


Figura 3.10: Gráficas de simulación evento 1.

En la figura 3.10 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1 y S2 = 1 ; S3 y S4 = 0, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 255 (Azul), motor derecho 200 (Amarillo).

Evento 2 En la figura 3.11 se observa una salida en valores PWM de 114 el motor derecho y 255 el motor izquierdo.

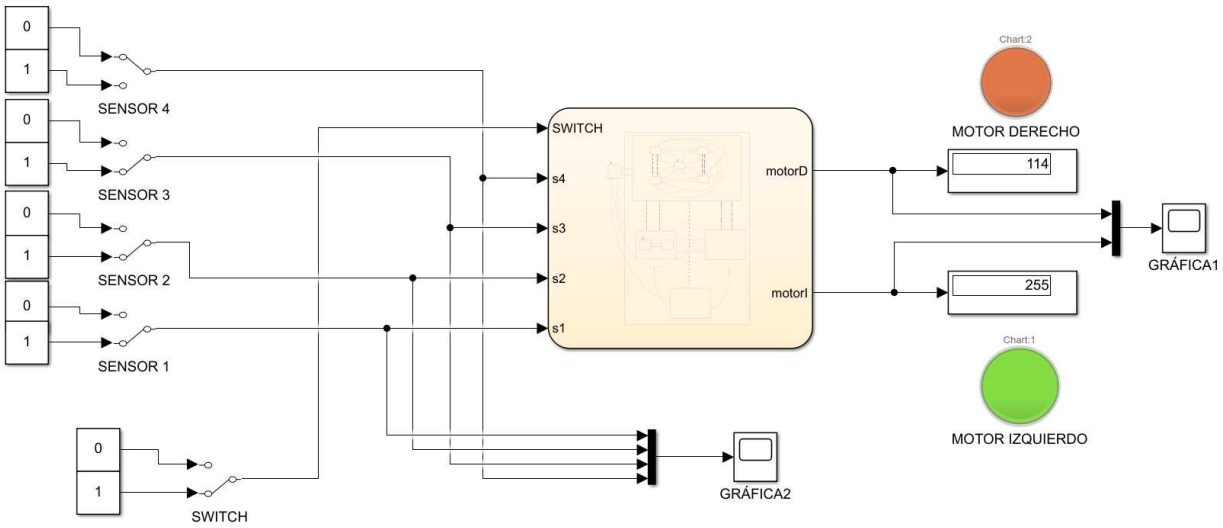


Figura 3.11: Evento 2.

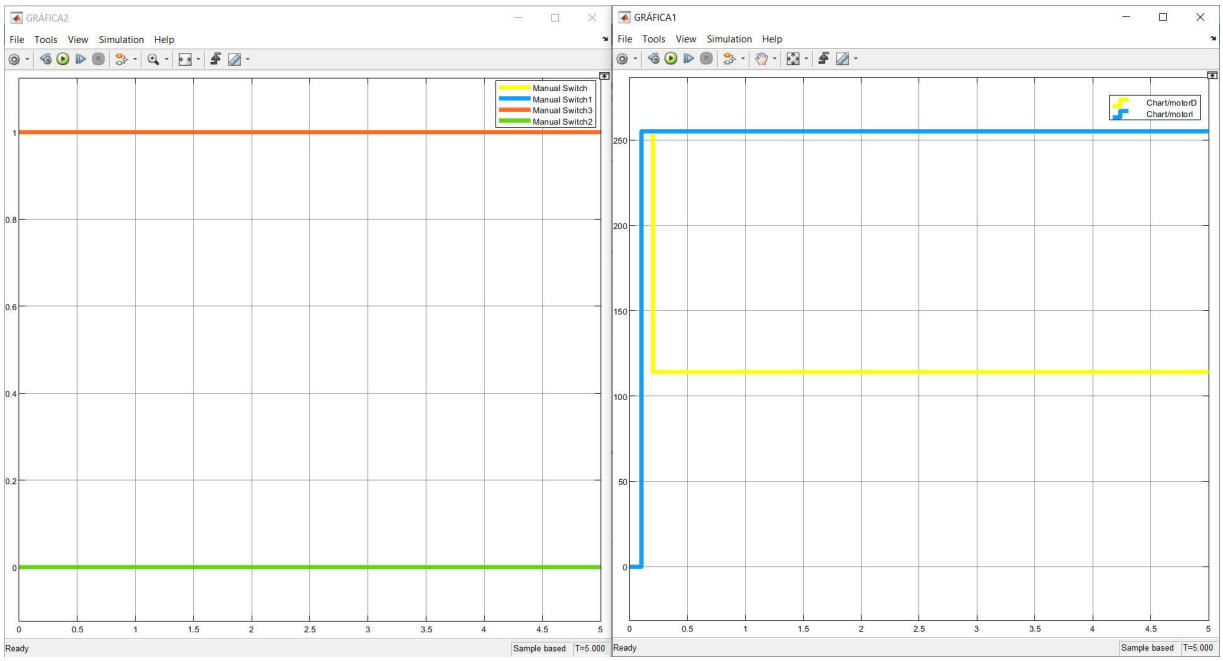


Figura 3.12: Gráficas de simulación evento 2.

En la figura 3.12 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1, S2 y S3 = 1 y S4 = 0, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 255 (Azul), motor derecho 114 (Amarillo).

Evento 3 En la figura 3.13 se observa una salida en valores PWM de 255 el motor derecho y 255 el motor izquierdo.

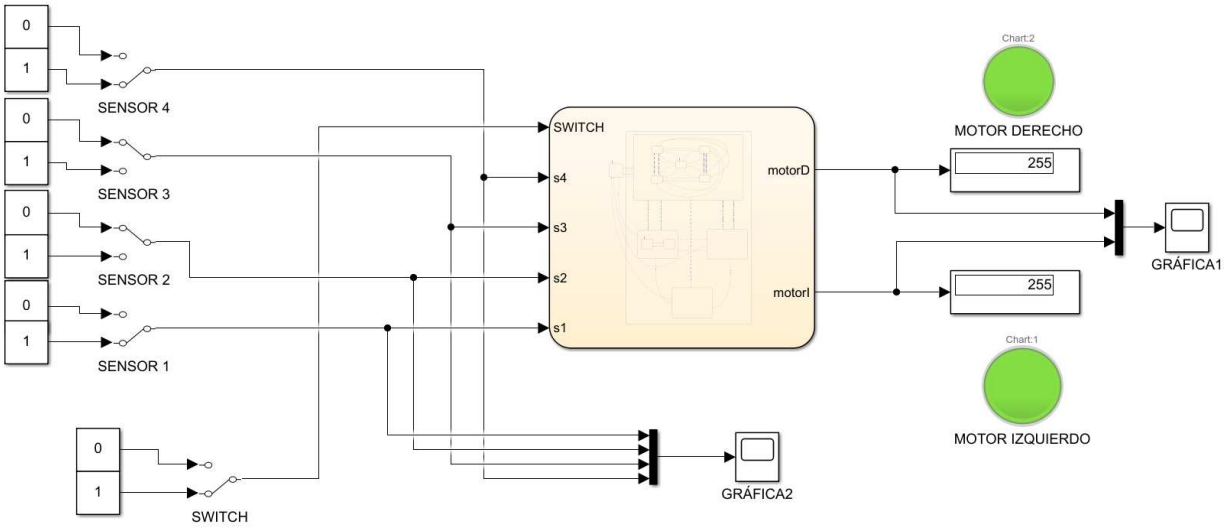


Figura 3.13: Evento 3.

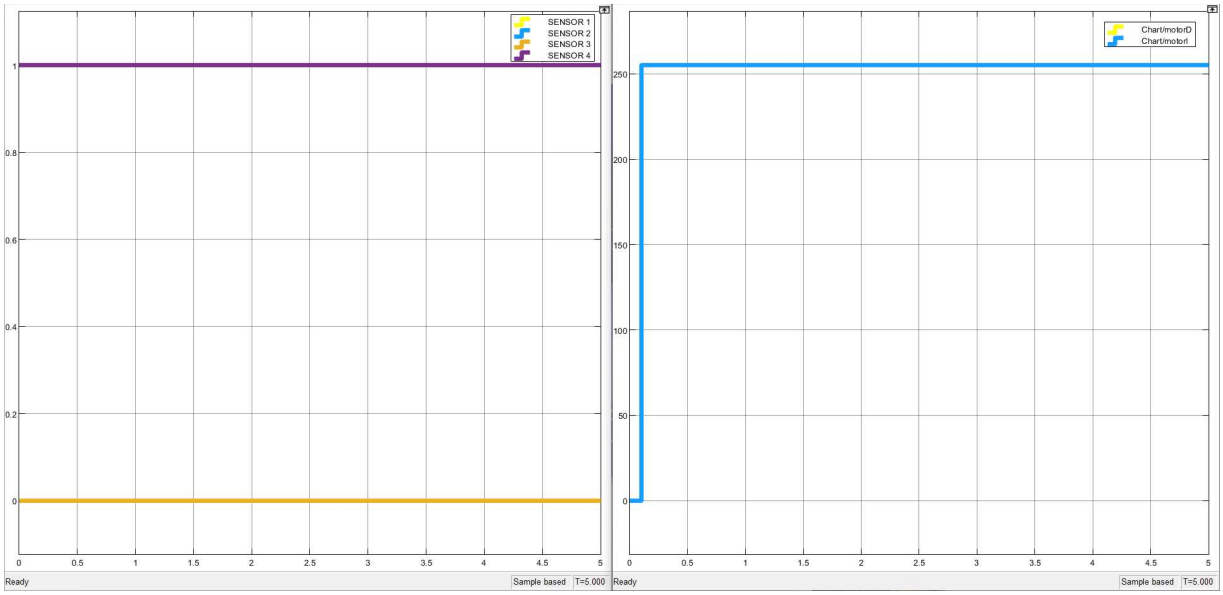


Figura 3.14: Gráficas de simulación evento 3.

En la figura 3.14 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1 = 1; S2 y S3 = 0; S4 = 1, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 255 (Azul), motor derecho 255 (Amarillo).

Evento 4 En la figura 3.15 se observa una salida en valores PWM de 255 el motor derecho y 114 el motor izquierdo.

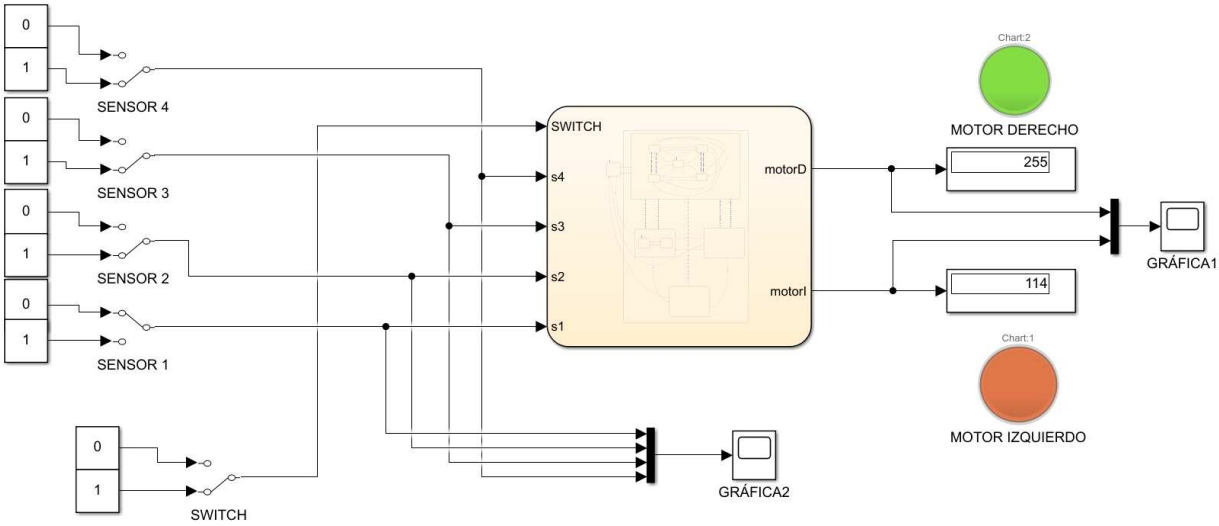


Figura 3.15: Evento 4.

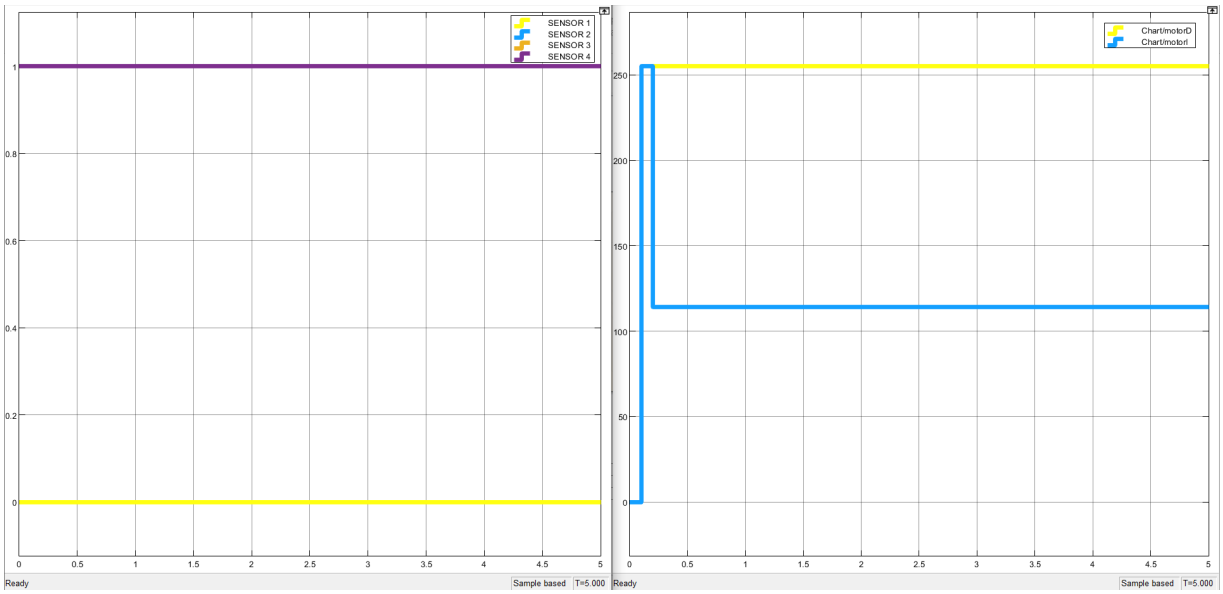


Figura 3.16: Gráficas de simulación evento 4.

En la figura 3.16 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1 = 0; S2, S3 y S4 = 1, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 114 (Azul), motor derecho 255 (Amarillo).

Evento 5 En la figura 3.17 se observa una salida en valores PWM de 255 el motor derecho y 200 el motor izquierdo.

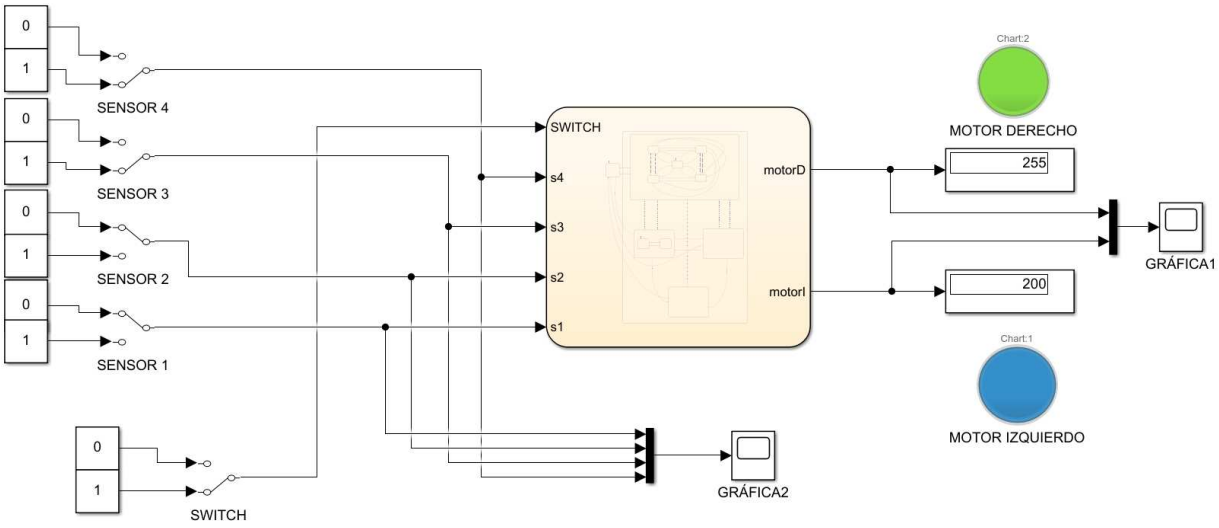


Figura 3.17: Evento 5.

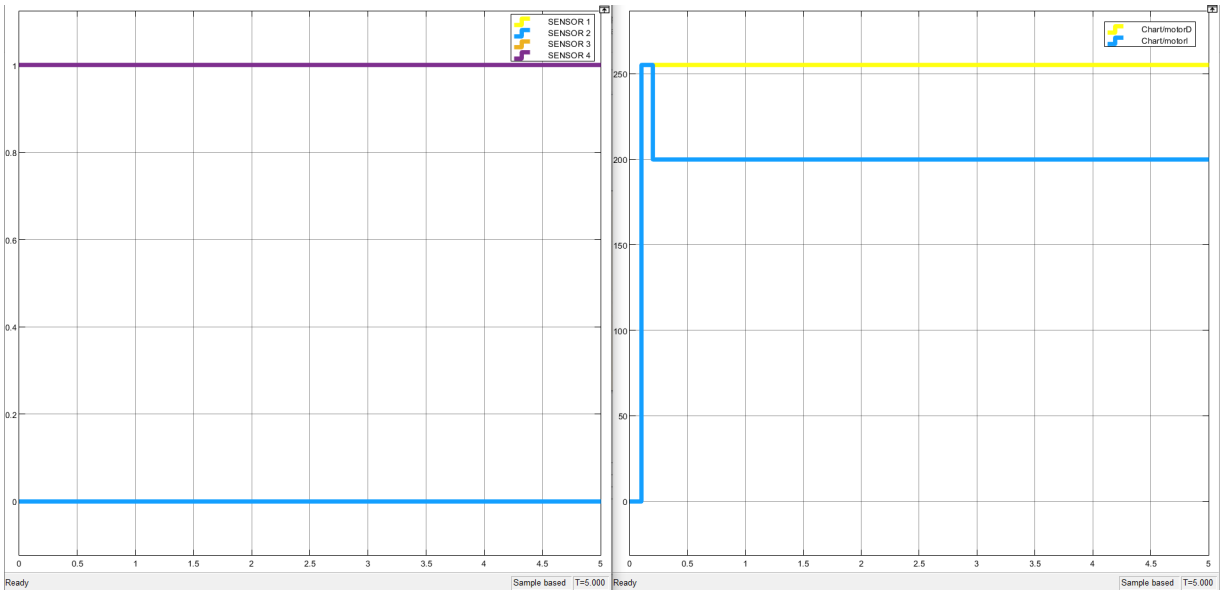


Figura 3.18: Gráficas de simulación evento 5.

En la figura 3.18 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1 y S2 = 0; S3 y S4 = 1, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 200 (Azul), motor derecho 255 (Amarillo).

3.6.3. Estado ”Giros”

Evento 6 En la figura 3.19 se observa una salida en valores PWM de 255 el motor derecho y 0 el motor izquierdo.

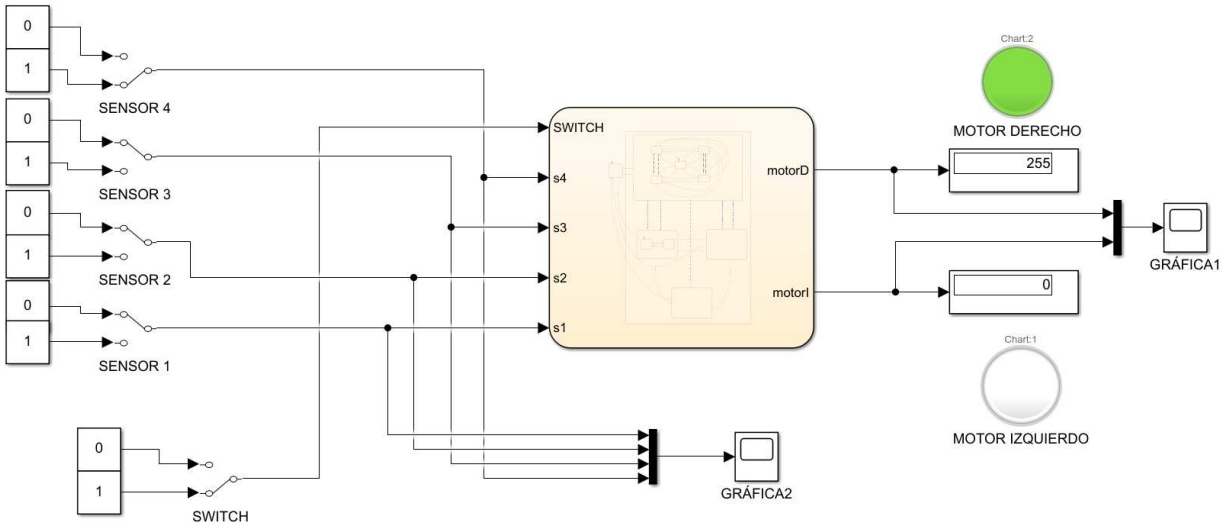


Figura 3.19: Evento 6.

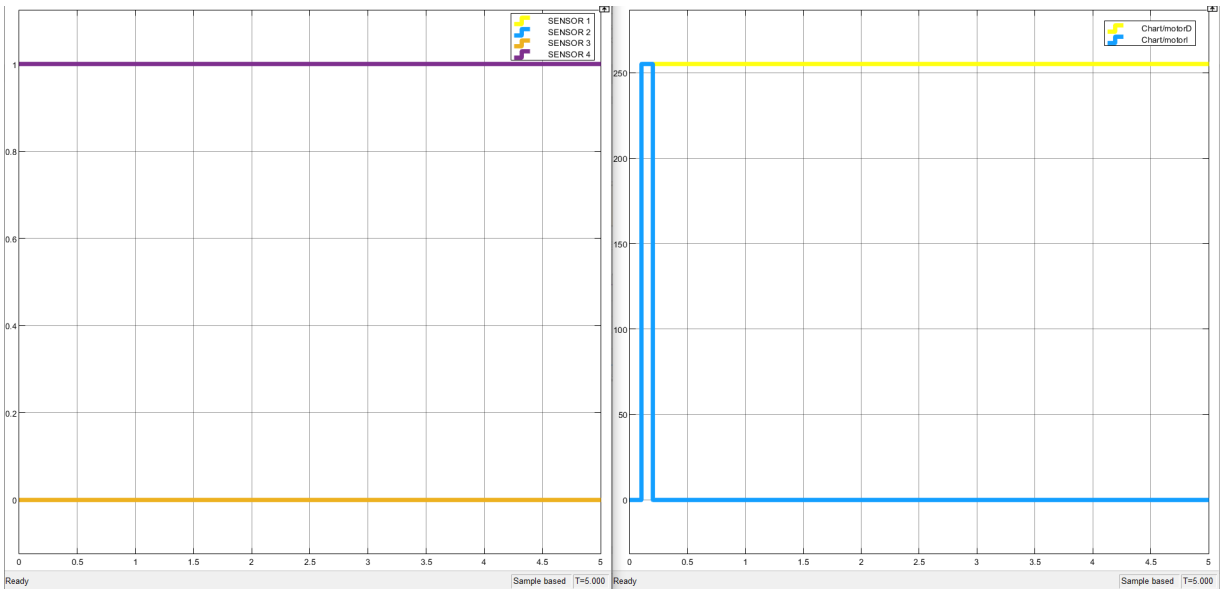


Figura 3.20: Gráficas de simulación evento 6.

En la figura 3.20 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1, S2 y S3=0 y S4 = 1, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 0 (Azul), motor derecho 255 (Amarillo).

Evento 7 En la figura 3.21 se observa una salida en valores PWM de 0 el motor derecho y 255 el motor izquierdo.

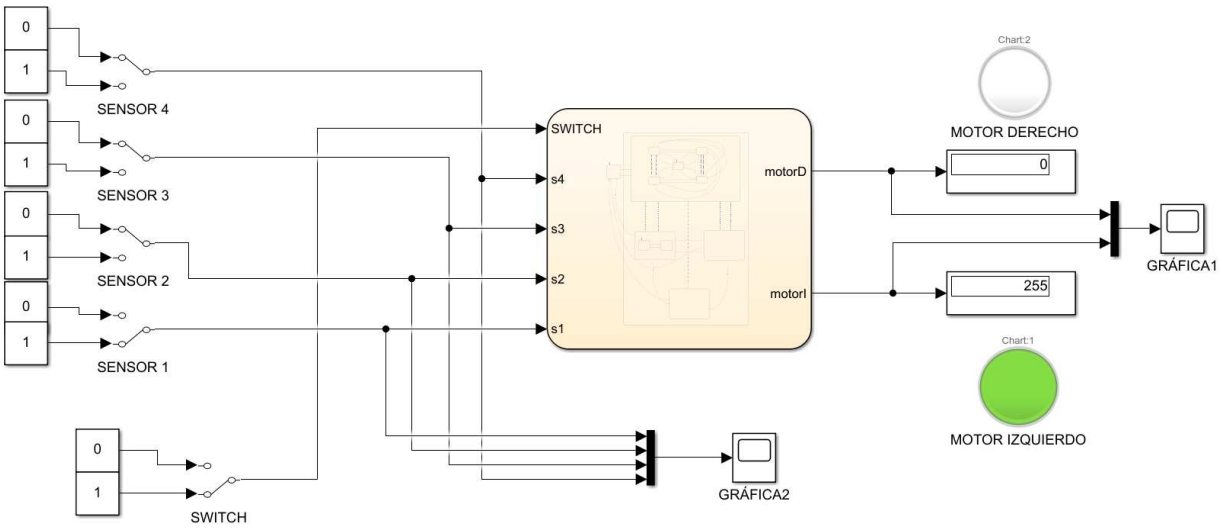


Figura 3.21: Evento 7.

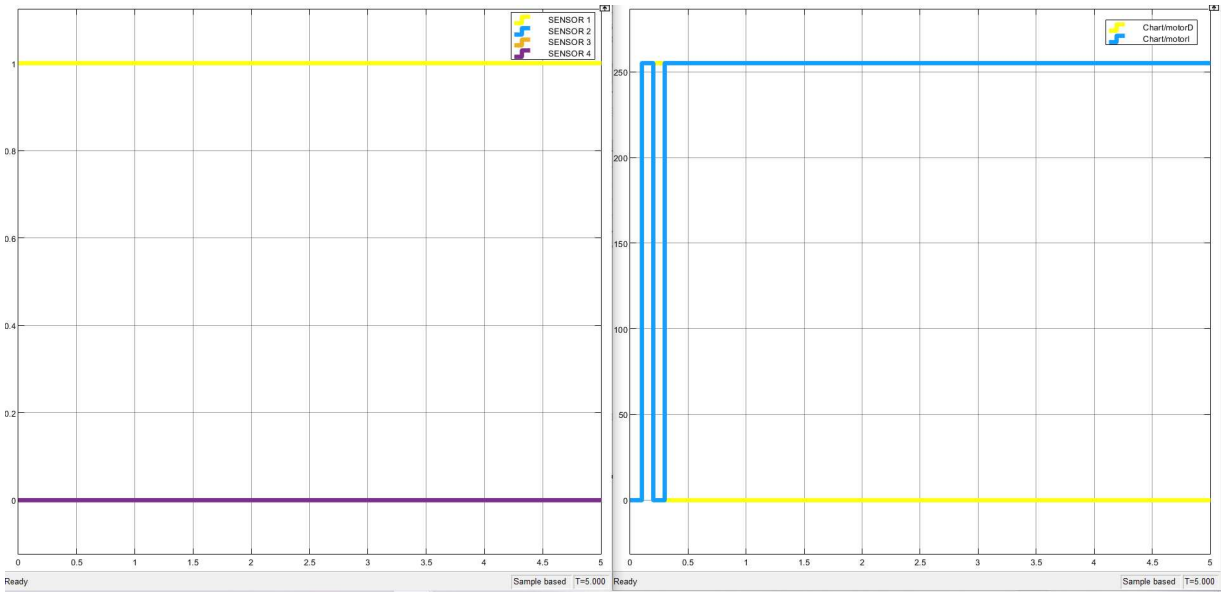


Figura 3.22: Gráficas de simulación evento 7.

En la figura 3.22 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1 = 1; S2, S3 y S4 = 0, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 255 (Azul), motor derecho 0 (Amarillo).

3.6.4. Estado ”Retroceso”

Evento 8 En la figura 3.23 se observa una salida en valores PWM de 255 el motor derecho y 255 el motor izquierdo.

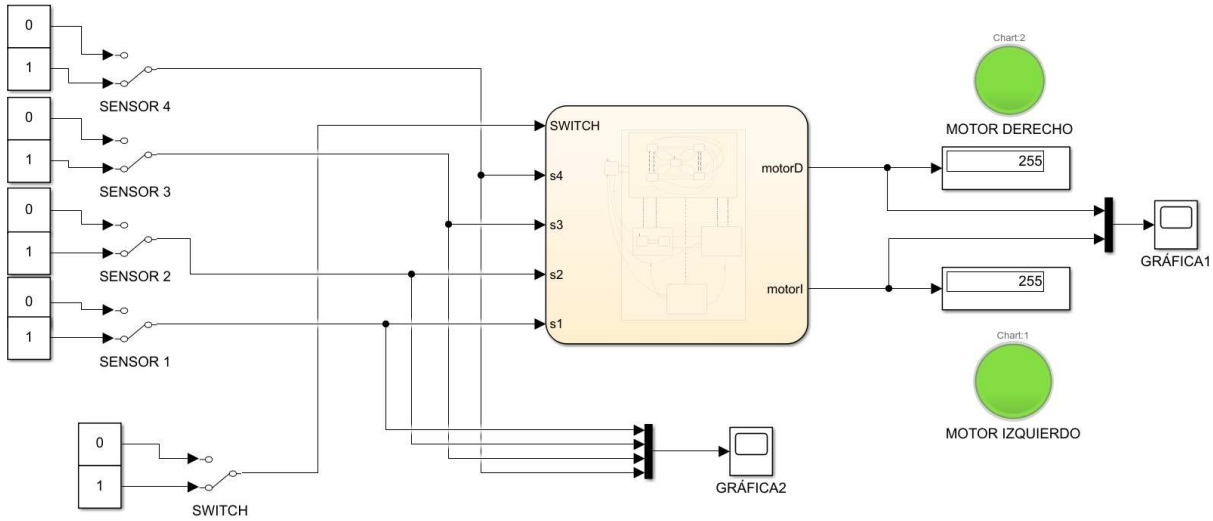


Figura 3.23: Evento 8.

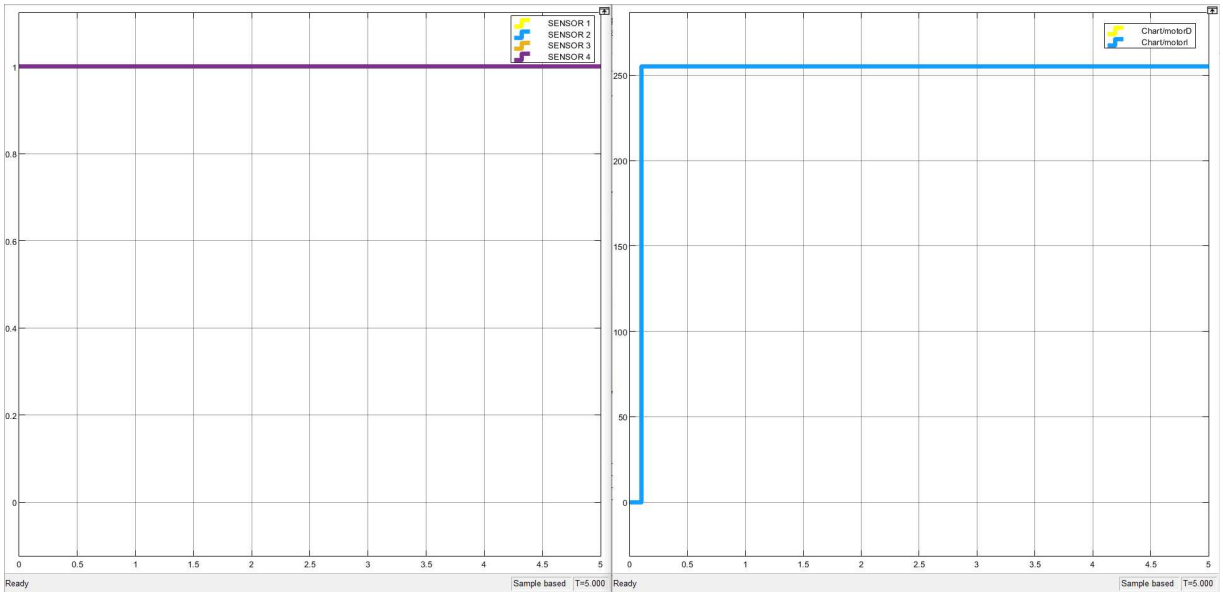


Figura 3.24: Gráficas de simulación evento 8.

En la figura 3.24 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1, S2, S3 y S4 = 1, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 255 (Azul), motor derecho 255 (Amarillo). Cabe destacar que el sentido de giro de los motores y el movimiento de serpenteo es ajustado en la respectiva programación.

3.6.5. Estado "Fin"

Evento 9 En la figura 3.25 se observa una salida en valores PWM de 0 el motor derecho y 0 el motor izquierdo.

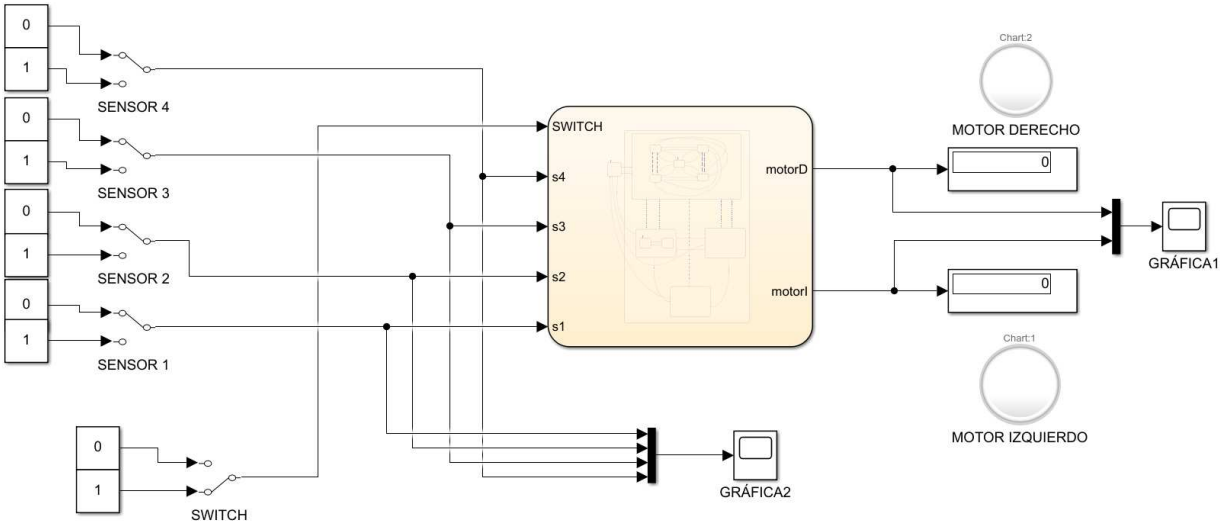


Figura 3.25: Evento 9.

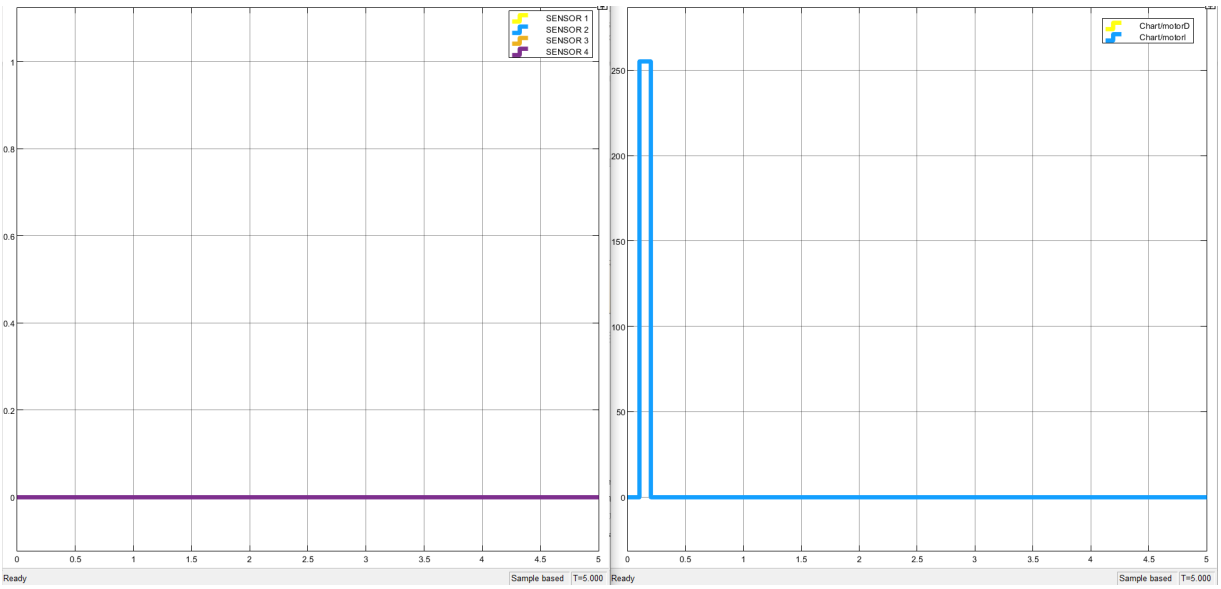


Figura 3.26: Gráficas de simulación evento 9.

En la figura 3.26 se observa en la gráfica de la izquierda la señal de los 4 sensores con la siguiente valoración: S1, S2, S3 y S4 = 0, y en la gráfica de la derecha la señal de las 2 salidas en valores PWM: motor izquierdo 0 (Azul), motor derecho 0 (Amarillo). Este evento demuestra que se ha concluido el trayecto.

Capítulo 4

Resultados y Análisis

En este capítulo habla acerca de la implementación de la máquina de estados finitos (FSM) que se desarrolla en el capítulo anterior. Se utiliza un robot seguidor de línea real y un código de programación que evalúa el desempeño de cada estado, variando los valores PWM y analizando el resultado por tramos estratégicos, en dos pistas definidas por el autor.

4.1. Arquitectura del Robot a utilizar

Se utiliza un robot seguidor de línea de 113 mm de ancho y 210 mm de largo, con eje diferencial que tiene un peso de 1 lb. Este robot cuenta con una serie de elementos en su hardware para realizar diversas pruebas, como se indica en la vista general de la figura 4.1. El robot tiene un indicador en el centro de los actuadores, y otro centrado en la parte trasera a 130 mm desde el eje de los actuadores, estos marcan la desviación de la trayectoria cuando entra en funcionamiento sobre la pista.

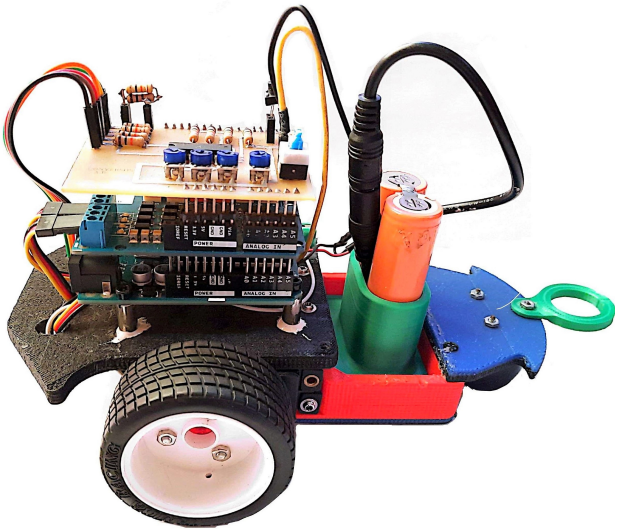


Figura 4.1: Robot a utilizar.

4.1.1. Elementos

En esta sección se presenta brevemente los materiales que se utilizan para la construcción del robot. Asimismo en la figura 4.2 se observa la comunicación entre los diferentes elementos electrónicos.

- Sensores QRD-1114

- Motores modelo HS 422

- Arduino Uno

- Motor Shield L298P

- Batería ICR18650

- Comparador LM339N

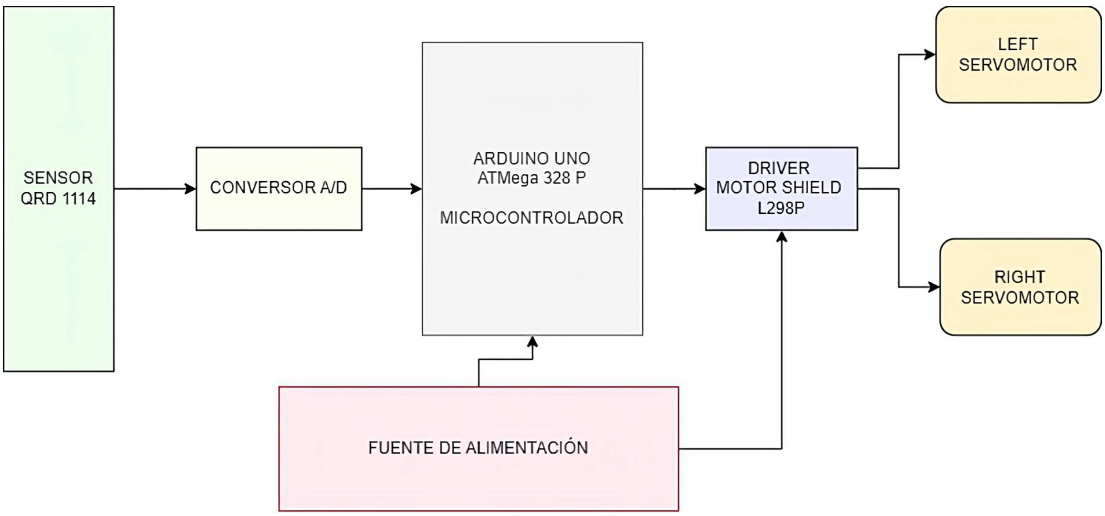


Figura 4.2: Bloques de Comunicación.

4.1.2. Configuración Electrónica

Consiste en implementar un diagrama electrónico que permita observar la conexión entre los diferentes elementos del robot.

Sensores Se realiza la siguiente configuración, que representa la interconexión de los cuatro sensores infrarrojos de forma lineal y se obtiene sus respectivas salidas analógicas y entradas para el suministro eléctrico, tal como se muestra en la figura 4.3. Las conexiones al digitalizador análogo se realizan en orden de izquierda a derecha, según su etiquetado. (véase en el anexo B sus diagramas esquemáticos)

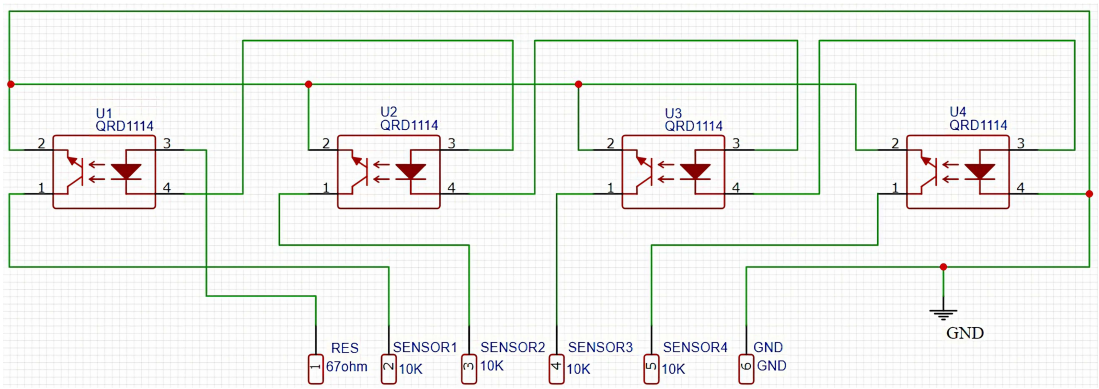


Figura 4.3: Conexión de sensores.

Estos sensores infrarrojos están situados en la parte inferior del robot, a un centímetro por encima del eje de los actuadores de manera lineal y poseen una separación de 3 mm entre cada uno, como lo muestra la siguiente figura 4.4.

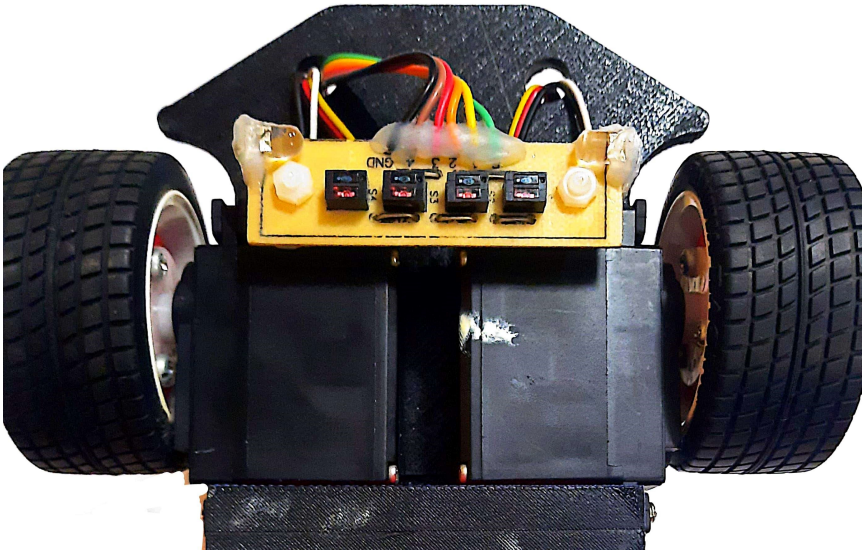


Figura 4.4: Posición de sensores infrarrojos.

Digitalizador Análogo La configuración del digitalizador Análogo es fundamental para establecer una conexión electrónica digital entre el microcontrolador y los sensores. Las etiquetas de los sensores que se muestran en la figura 4.3 se conectan a las etiquetas y a la configuración del conversor que utiliza el comparador LM339N que muestra la figura 4.5. (véase en el anexo C sus diagramas esquemáticos)

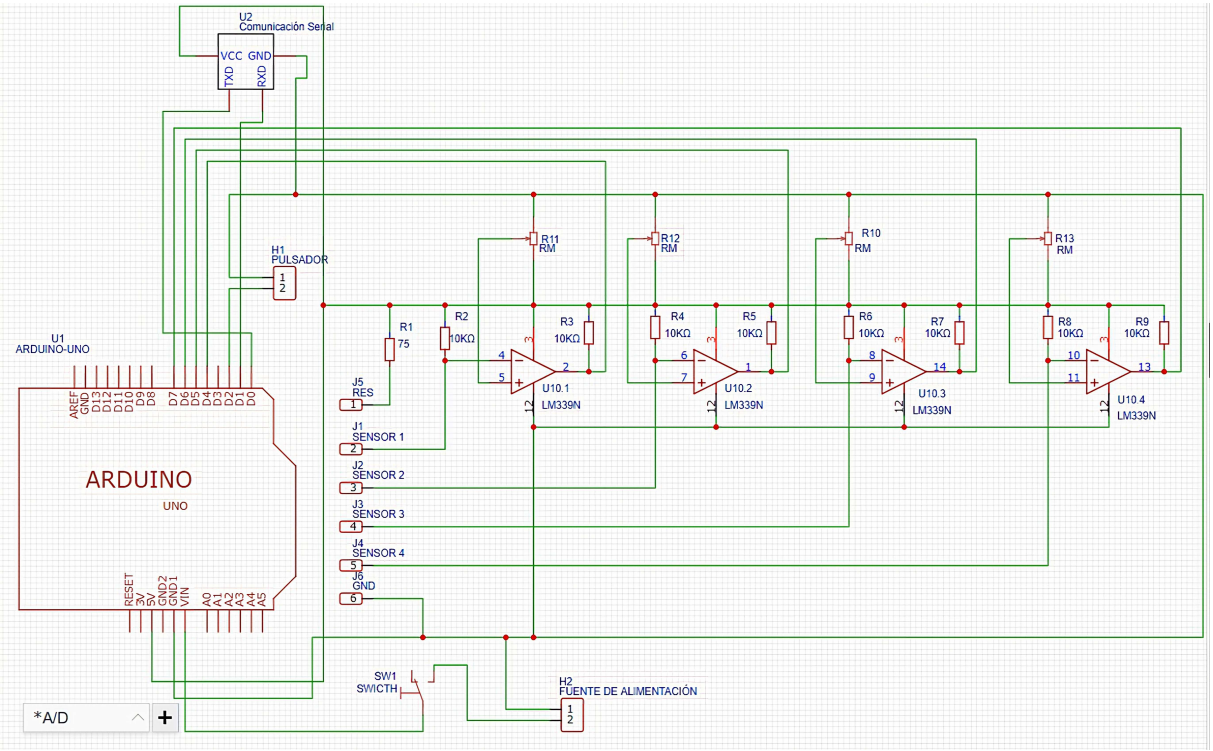


Figura 4.5: Diagrama electrónico del digitalizador análogo.

En la siguiente figura 4.6 se observa cómo se conecta el digitalizador análogo al microcontrolador y a la tarjeta motor shield L298P.

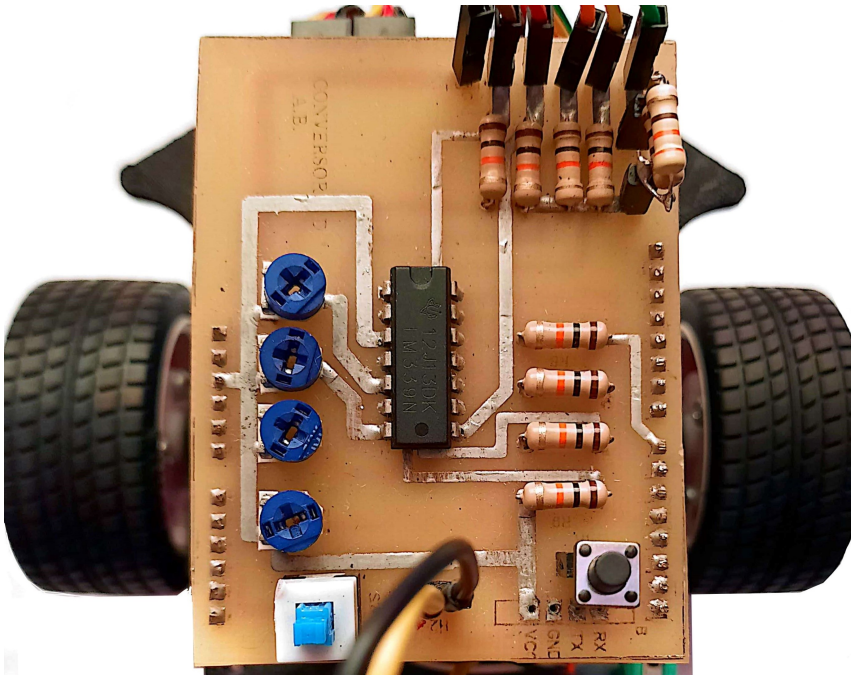


Figura 4.6: Posición de la placa conversora A/D.

Motor Shield L298P Para este caso en particular, solo es necesario conectar esta placa estandarizada a la tarjeta de Arduino.

4.2. Escenario 1 ”Pista Normal”

Para realizar las pruebas de control se realiza una pista normal de 4 metros de recorrido y 18 mm de ancho como se muestra en la figura 4.7, en temperatura normal e iluminación adecuada en la cual se pone a prueba el robot seguidor de línea. Esta pista consta de diferentes cambios de trayectoria como: curvas abiertas, rectas, inclinadas.

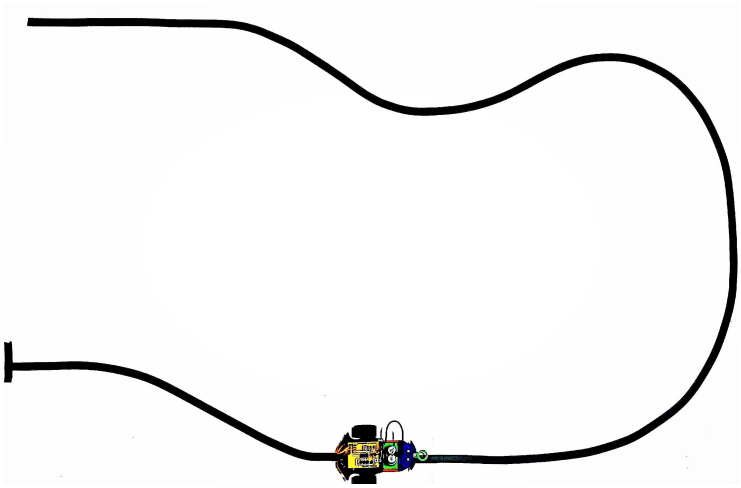


Figura 4.7: Pista normal de cuatro metros.

4.3. Escenario 2 ”Curvas”

La pista número dos tiene como objetivo analizar el comportamiento del robot en situaciones mas complejas como en curvas cerradas y se proyecta en la figura 4.8

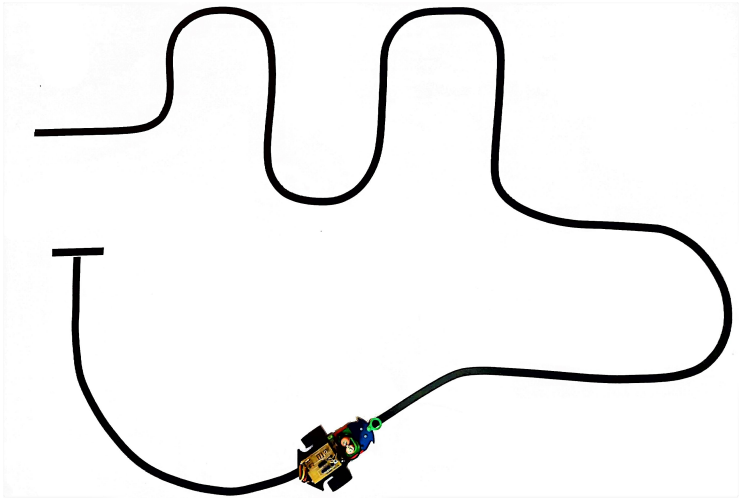


Figura 4.8: Pista de curvas cerradas de 5 metros.

4.4. Parámetros de Pruebas

Para realizar las diferentes pruebas de rendimiento de la máquina de estados finitos, se presenta los resultados tomando en cuenta diversos parámetros como: la velocidad, el tiempo, la fluidez y la estabilidad del robot seguidor de línea para completar su recorrido.

El control del robot se analiza con diferentes medidas y técnicas como:

Desviación de la trayectoria: para evaluar la estabilidad del robot seguidor de línea se observa la desviación de la trayectoria respecto a la línea negra de referencia. Esta medida se puede calcular como la distancia promedio entre el centro del robot y la línea negra a lo largo del recorrido. Si esta desviación es pequeña y constante, se dice que el robot tiene buena estabilidad.

Variabilidad de la velocidad: otro aspecto importante para la estabilidad de un robot seguidor de línea es la consistencia en la velocidad de desplazamiento. Si el robot varía mucho su velocidad a lo largo del recorrido, puede ser indicativo de problemas de control o de ajuste de los sensores. Una forma de medir esta variabilidad es calcular la desviación estándar de las velocidades registradas en diferentes tramos de la trayectoria.

Fluidez: se analiza la capacidad del robot seguidor de línea para corregir errores teniendo en cuenta la frecuencia y la magnitud de las oscilaciones que se producen alrededor de la línea negra. Si el robot realiza correcciones suaves y precisas, es probable que tenga buena estabilidad. Si, por el contrario, el robot oscila demasiado o hace correcciones bruscas, puede indicar problemas de ajuste de los controladores o de los sensores.

Tiempo de respuesta: esta medida es importante para evaluar la estabilidad de un robot se-

guidor de línea que presenta ante los cambios en la trayectoria o en las condiciones del entorno. Si el robot responde rápidamente y de forma consistente a estos cambios, es probable que tenga buena estabilidad. Si, por el contrario, el robot tarda demasiado en detectar o corregir desviaciones, puede indicar problemas de control o de sensibilidad de los sensores.

4.4.1. Prueba FSM Matlab

Si se desea llevar a cabo una prueba en tiempo real, se requiere implementar la máquina de estados finitos que se desarrolla en el capítulo 3. Esto se puede lograr sustituyendo los bloques de entradas y salidas utilizando la librería de Arduino en Simulink, asignando los nombres de las variables correspondientes como se muestra en la figura 4.9. De esta manera, se puede obtener una evaluación en tiempo real entre el prototipo y la interfaz, obteniendo las gráficas respectivas de los resultados del funcionamiento sobre la pista hecha por el autor, esto sin la necesidad de un código de programación.

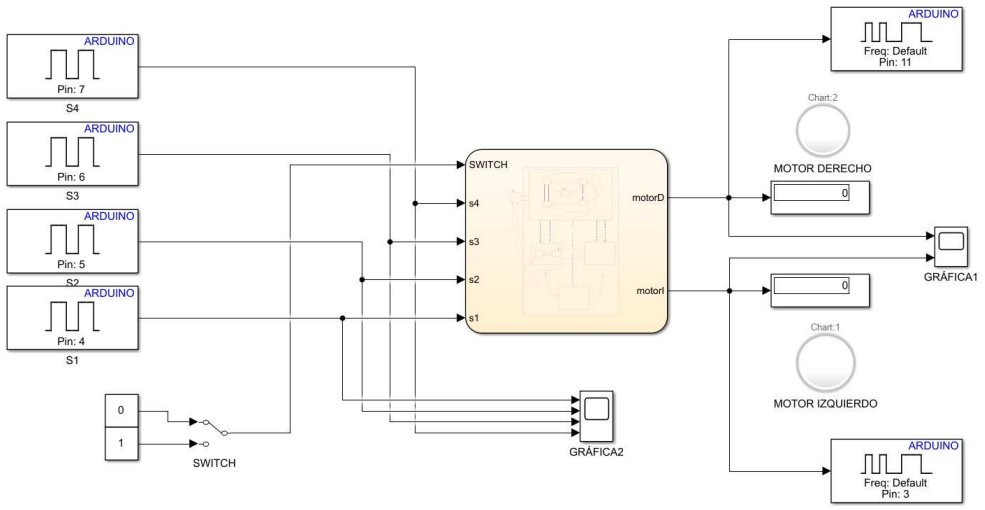


Figura 4.9: FSM en tiempo real.

Durante las pruebas, se utilizan dos escenarios diferentes (ver Figuras 4.7 y 4.8) y se emplea un código de programación equivalente a la máquina de estados. Se utiliza valores de ancho de pulso (PWM) que corresponden a la programación de la FSM detallada en el anexo A. En concreto, para el estado ”Avance” se utilizan los valores de 60, 100, 150, 200 y 255 según corresponda a cada evento, mientras que para el estado ”Giros” se emplean los valores de 0 y 255. Por último, para el estado ”Retroceso” se utiliza un valor de 150.

4.4.2. Prueba 1

En la prueba de la figura 4.10, se usa el escenario número uni ”Pista Normal” y la trayectoria que realiza el robot se presenta con el indicador central del robot.

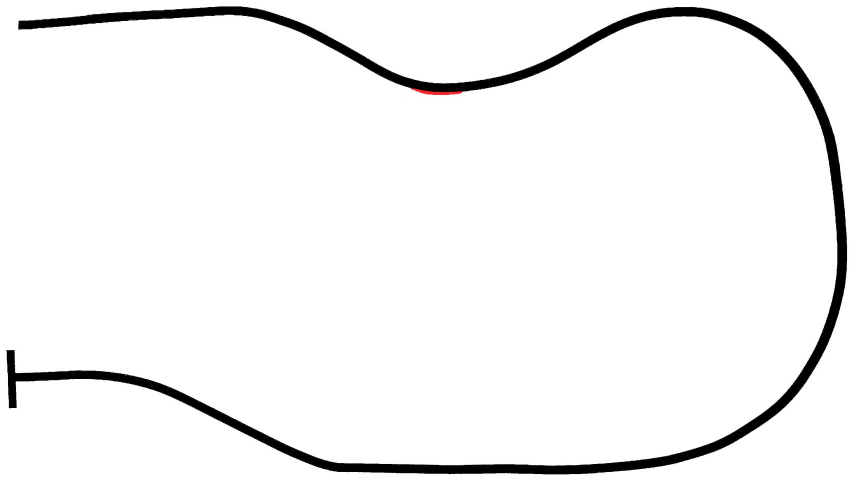


Figura 4.10: Prueba 1 ”indicador central”.

El indicador central permite apreciar que existe un control muy estable a la trayectoria (no existe desviaciones).

Se usa el escenario número uno nuevamente, pero esta vez con el indicador trasero del robot (a 130 mm del eje), se lo secciona en diferentes tramos. El objetivo de esto es analizar la fidelidad que tiene el recorrido del robot con respecto a la línea negra de la pista en casos extremos de inestabilidad. Los tramos son los siguientes y se aprecia en la figura 4.11:

- ◇ Tramo uno: Recta inicial (52 cm).
- ◇ Tramo dos: Curva en ”S” (95 cm).
- ◇ Tramo tres: Curva abierta (113 cm).
- ◇ Tramo cuatro: Recta continua (60 cm).
- ◇ Tramo cinco: Inclinada y curva fin (80 cm).

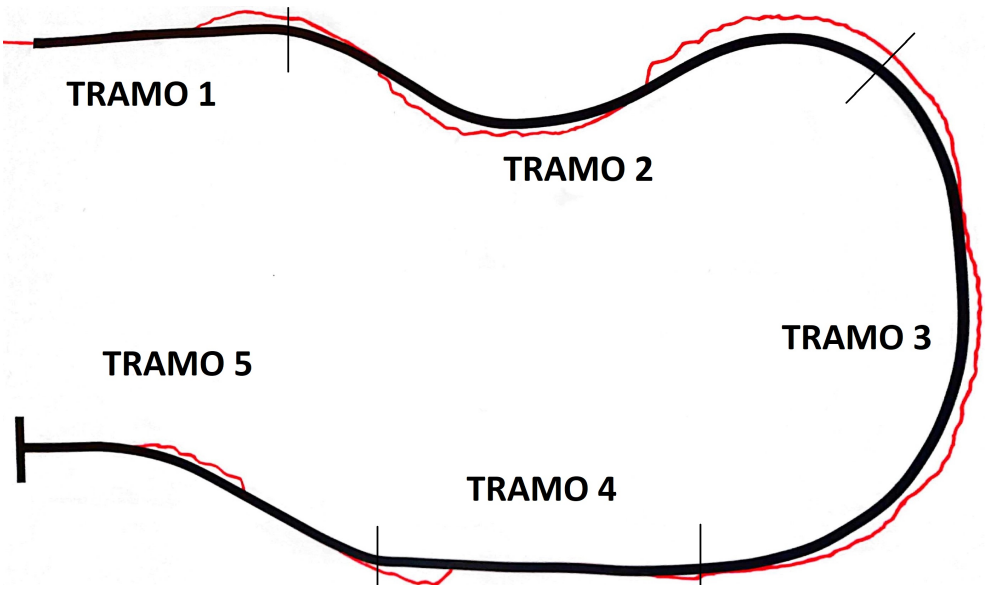


Figura 4.11: Prueba 1 ” indicador a 130 mm”

1. Primer Tramo: Permanece en el estado ”Parado”, luego con el pulsador el robot avanza en línea recta, su fidelidad es alta en la trayectoria, tiene una velocidad constante y utiliza el evento ideal del estado ”Avance” de la FSM al final se prepara para cambiar al otro tramo girando hacia la derecha.

2. Segundo Tramo: Para tomar la curva en ”s” el robot utiliza el estado ”Avance”, el estado ”Giros” y ”retroceso”, posee una fidelidad admisible, tiene algunas oscilaciones y tiene buena estabilidad.

3. Tercer Tramo: En la curva abierta tiene buena fidelidad y estabilidad, presenta oscilaciones, su control es continuo y utiliza los estados de ”Avance” y ”Giros”.

4. Cuarto Tramo: En la recta continua utiliza el estado ”Avance” teniendo una velocidad constante, y mantiene una trayectoria recta sin salirse de la línea como en el tramo 1.

5. Quinto Tramo: Se presenta el estado ”Giros” y ”Avance” presenta alta estabilidad y fidelidad con oscilaciones normales en el cambio de estado y finaliza con el estado ”Fin”,

Los diferentes datos de cada tramo se presenta en la tabla 4.1.

Tabla 4.1: Resultados prueba 1

Criterios a Evaluar	Tramo 1	Tramo 2	Tramo 3	Tramo 4	Tramo 5
Tiempo	3.49 s	9.31 s	8.61 s	4.10 s	5.97 s
Velocidad	14.89 cm/s	10.20 cm/s	13.12 cm/s	14.63 cm/s	13.40 cm/s
Fluidez	MUY ALTA	BAJA	MEDIA	MUY ALTA	MEDIA
Estabilidad	MUY ALTA	MEDIA	ALTA	MUY ALTA	ALTA

4.4.3. Prueba 2

En la prueba de la figura 4.12, se usa el escenario número dos ”Curvas” y la trayectoria que realiza el robot se presenta con el indicador central del robot.

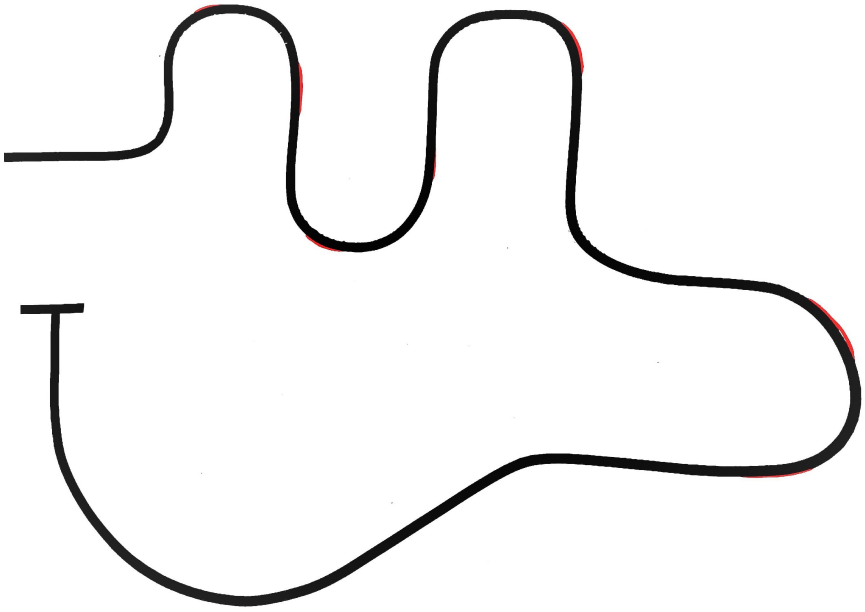


Figura 4.12: Prueba 2 ”indicador central”.

Se aprecia que existe un control muy estable y fiel a la trayectoria.

Al igual que el caso anterior, se utiliza el escenario 2, pero esta vez con el indicador trasero del robot y se lo secciona en diferentes tramos detallados a continuación en la figura 4.13. La realización de esto tiene como finalidad analizar la fidelidad en casos extremos de falla, que tiene el recorrido del robot con respecto a la línea negra de la pista, cuando trabaja con curvas cerradas.

◇ Tramo uno: Recta inicial (28 cm).

◇ Tramo dos: Curvas cerradas cortas (219 cm).

◇ Tramo tres: Curva cerrada larga (120 cm).

◇ Tramo cuatro: Inclinada y curva abierta (133 cm).

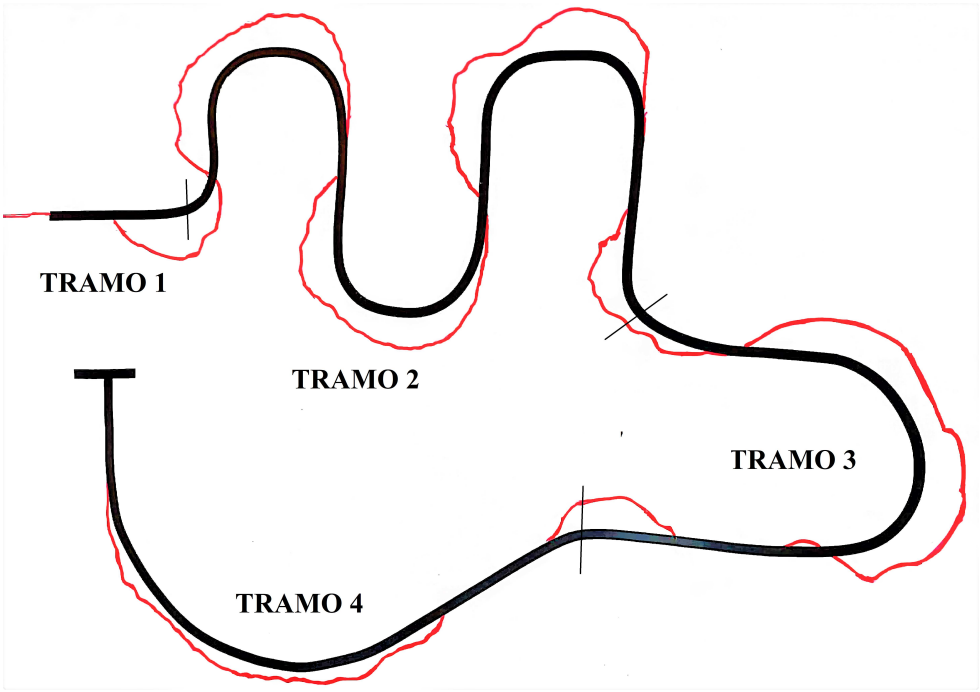


Figura 4.13: Prueba 2 ”indicador a 130 mm”.

1. Primer Tramo: Permanece en el estado ”Parado”, cuando se presiona el pulsador de encendido el robot avanza en línea recta, posee una velocidad constante, adquiere una estabilidad y fluidez muy alta, utiliza el evento ideal del estado ”Avance” de la FSM al final se prepara para cambiar de estado cuando gira a la izquierda.

2. Segundo Tramo: Para tomar la curvas cerradas, el robot utiliza el estado de ”Avance” dando paso a los eventos correspondientes que actúan en este tipo de escenarios, hasta

llegar al estado ”Giros” y por último entra en el estado ”Retroceso” cuando el robot sale de la trayectoria antes de llegar al tramo 3 para reestablecer la conexión en la pista, posee una fidelidad admisible, tiene algunas oscilaciones de velocidad debido al cambio de estado, tiene una fluidez media y posee una estabilidad alta.

3. Tercer Tramo: En la curva cerrada larga, tiene buena fidelidad en la pista, su estabilidad es buena, presenta pocas oscilaciones, su control es continuo y utiliza los estados de ”Avance” y ”Giros” al último prepara para entrar en una inclinación.

4. Cuarto Tramo: Al inicio, se presenta el estado de ”Avance” con su evento ideal, luego entra en el estado ”Giros” y ”Avance” alternándose, presenta alta estabilidad y fidelidad con oscilaciones normales en el cambio de estado y finaliza con el estado ”Fin”,

Los diferentes datos de cada tramo se presenta en la tabla 4.2.

Tabla 4.2: Resultados prueba 2

Criterios a Evaluar	Tramo 1	Tramo 2	Tramo 3	Tramo 4
Tiempo	2.09 s	21.23 s	11.68 s	11.56 s
Velocidad	13.39 cm/s	10.31 cm/s	10.27 cm/s	11.50 cm/s
Fluidez	MUY ALTA	MEDIA	MEDIA	ALTA
Estabilidad	MUY ALTA	ALTA	ALTA	ALTA

Capítulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

- La máquina de estados finitos (FSM) demostró ser una herramienta fundamental para el control del robot seguidor de línea en lazo cerrado, al proporcionar una simulación del sistema sólidamente estructurado y organizado. Basándose en estados de trabajo, esta tecnología permitió al dispositivo tomar decisiones rápidas y precisas en respuesta a las señales captadas por los sensores infrarrojos, lo que a su vez se tradujo en un desplazamiento fluido y preciso a lo largo del recorrido.

- La herramienta Stateflow del entorno Simulink de Matlab facilitó la creación de la FSM de tipo Mealy la misma que constó de 5 estados, 9 eventos y 36 transiciones, basándose en las posibles situaciones de funcionamiento en los que puede estar sometido el robot. En la simulación estado por estado, se consideró 4 entradas que representan la señal digital de

los sensores infrarrojos y 2 salidas que muestra el valor PWM que toman los actuadores a través del driver.

- El uso de la FSM posibilitó establecer los lazos de control adecuados para cada estado de trabajo mediante la identificación de estados, definiendo las transiciones, asociando acciones, diseñando la lógica de control e implementando y probando la simulación en el software.

- El algoritmo FSM implementado en las pruebas del robot seguidor de línea demostró un control altamente exitoso en el escenario ”Normal” de 4 metros, al transitar por tramos lineales y curvas abiertas. Sin embargo, se observó una ligera afectación en su desempeño en curvas cerradas en el escenario ”Curvas” de 5 metros. A pesar de esto, se pudo comprobar que la técnica FSM es eficiente en las pruebas realizadas, obteniendo resultados satisfactorios en todos los tramos.

5.2. Recomendaciones

- Utilizar el software matemático, reemplazar todos los elementos de entrada y salida de la FSM por los bloques de la librería de Arduino y establecer la comunicación, desde una conexión serial con el prototipo. Esto permitirá interactuar con la interfaz en tiempo real.

- Ubicar los sensores estratégicamente. Esto permitirá obtener lecturas más precisas del entorno y a su vez trabajar con más estados en el algoritmo de la FSM, lo cual contribuirá a mejorar su estabilidad y desempeño general.

5.3. Trabajo a Futuro

- Realizar una comparativa entre una FSM y un algoritmo convencional por eventos como ejemplo: control PID, o redes neuronales. O a su vez, implementar a la FSM este tipo de algoritmo, utilizando escenarios con obstáculos o más complejos.

- Mejorar el prototipo de pruebas, especialmente a sus elementos de hardware, con el objetivo de aumentar la versatilidad del algoritmo FSM en velocidad y estabilidad. Esto es especialmente relevante para situaciones en donde este parámetro es crucial, como en competencias o retos.

Bibliografía

[1] L. F. Ortiz Arroyave, “Metodología de Navegación de robots móviles para detección de vanos y discontinuidades en su trayectoria,” p. 126, 2019, [Online]. Available: http://hdl.handle.net/20.500.12622/1396

[2] M. C. Romero, J. Antonio, C. Soto, G. Axel, A. Gutiérrez, and F. R. Rico, “Robótica Sistema de control y arquitectura de un robot seguidor de línea Resumen Arquitectura del Robot Seguidor de Línea,” CULCyT, no. 59, pp. 115–128, 2016, [Online]. Available: http://erevistas.uacj.mx/ojs/index.php/culcyt/article/view/1570/1390

[3] A. Kurt and Ü. Özgüner, “Hierarchical finite state machines for autonomous mobile systems,” Control Eng. Pract., vol. 21, no. 2, pp. 184–194, Feb. 2013, doi: 10.1016/j.conengprac.2012.09.020.

[4] A. Toumpa, A. Kouris, F. Dimeas, and N. Aspragathos, “Control of a line following robot based on FSM estimation,” IFAC-PapersOnLine, vol. 51, no. 22, pp. 542–547, 2018, doi: 10.1016/j.ifacol.2018.11.573.

[5] R. Balogh and D. Obdržálek, “Using Finite State Machines in Introductory Robotics,” Adv. Intell. Syst. Comput., vol. 829, no. April, pp. 85–91, 2019, doi:10,1007/978 – 3 – 319 – 97085 – 1_9.

[6] A. R. Al Tahtawi, Y. Somantri, and E. Haritman, “Design and Implementation of PID Control-based FSM Algorithm on Line Following Robot,” J. Teknol. Rekayasa, vol. 1, no. 1, p. 23, 2017, doi: 10.31544/jtera.v1.i1.2016.23-30.

[7] K. Qian and A. Song, “Autonomous navigation for mobile robot based on a sonar ring and its implementation,” 2012 8th IEEE Int. Symp. Instrum. Control Technol. ISICT 2012 - Proc., pp. 47–50, 2012, doi: 10.1109/ISICT.2012.6291646.

[8] V. Vergara and N. Silvana, “Implementación de sistemas de control PID en robots móviles autónomos usando entornos de simulación,” 2022.

[9] O. F. Gómez and U. E. Gómez, “Kinematic simulation of a line follower robot for the creation of the programming videogame rusty roads in the unity framework,” Inf. Tecnol., vol. 28, no. 5, pp. 55–64, 2017, doi: 10.4067/s0718-07642017000500008.

[10] P. Plaza, M. Blazquez, C. Perez, M. Castro, and S. Martin, “Arduino as an Educational Tool to I ntroduce R obotics,” no. December, pp. 1–8, 2018.

[11] M. W. Fatma and M. I. Hamid, “PWM speed control of dc permanent magnet motor using a PIC18F4550 microcontroller,” IOP Conf. Ser. Mater. Sci. Eng., vol. 602, no. 1, 2019, doi: 10.1088/1757-899X/602/1/012017.

[12] M. Aravena, D. Ruete, N. Campos, and A. Figueroa, Eds., ESTRATEGIAS PARA DESARROLLAR HABILIDADES DEL PENSAMIENTO EN LA EDUCACIÓN SUPERIOR: VARIABLES OCULTAS Y ESTRATEGIAS. Ariadna Ediciones, 2023. doi: 10.26448/ae.9789566095767.62.

[13] E. Leandro and T. Loaiza, "Diseño e implementación de un algoritmo de seguimiento de trayectorias para el minidron Parrot mambo utilizando – Simulink/Stateflow.," 2022.

[14] P. C. Jorgensen, "Finite State Machines," Cr. Model. Test., pp. 55–79, 2017, doi: 10.1201/9781315204970-4.

[15] P. Garapati and S. Musala, "Moore and Mealy Negative Edge detector A VHDL Example for Finite State Machine," pp. 1159–1161, 2020.

Anexos

Anexo A

Programación FSM

Se presenta el código de programación de la máquina de estados finitos para las pruebas.

```
// FINITE STATE MACHINE

const int leftSensorPin = 4;
const int centerLeftSensorPin = 5;
const int centerRightSensorPin = 6;
const int rightSensorPin = 7;
const int leftMotorPin = 3;
const int rightMotorPin = 11;
const int pul = 2;
const int blackThreshold = 0;
const int whiteThreshold = 1;
const int dirB = 13;
const int dirA = 12;
bool systemOn = false;

enum State {

  STATE_STOPPED,
  STATE_ADVANCING,
    EVENT1,
    EVENT2,
    EVENT3,
    EVENT4,
    EVENT5,
```

```
  STATE_TURNING,
    EVENT6,
    EVENT7,
  STATE_REVERSING,
  STATE_END,

};

void setup() {

  pinMode(leftSensorPin, INPUT);
  pinMode(centerLeftSensorPin, INPUT);
  pinMode(centerRightSensorPin, INPUT);
  pinMode(rightSensorPin, INPUT);
  pinMode(leftMotorPin, OUTPUT);
  pinMode(rightMotorPin, OUTPUT);
  pinMode(dirA, OUTPUT);
  pinMode(dirB, OUTPUT);
  pinMode(pul, INPUT_PULLUP);

}

void loop() {
   bool pulState = digitalRead(pul);

  if (pulState == LOW) {
    systemOn = !systemOn;
    delay(200);
  }

   static State state = STATE_STOPPED;

  if (systemOn) {
  int leftSensorValue = digitalRead(leftSensorPin);
  int centerLeftSensorValue = digitalRead(centerLeftSensorPin);
  int centerRightSensorValue = digitalRead(centerRightSensorPin);
  int rightSensorValue = digitalRead(rightSensorPin);
  digitalWrite(dirA, HIGH);
  digitalWrite(dirB, HIGH);

  //CONTROL CONTINUO
```

```
if (leftSensorValue == whiteThreshold &&
centerLeftSensorValue== blackThreshold &&
centerRightSensorValue == blackThreshold &&
rightSensorValue == whiteThreshold) {
state = EVENT1;
}

else if (leftSensorValue == blackThreshold &&
centerLeftSensorValue == blackThreshold &&
centerRightSensorValue == whiteThreshold &&
rightSensorValue == whiteThreshold) {
  state = EVENT2;
}

else if (leftSensorValue == blackThreshold &&
centerLeftSensorValue == whiteThreshold &&
centerRightSensorValue == whiteThreshold &&
rightSensorValue == whiteThreshold) {
  state = EVENT3;
}

else if (leftSensorValue == whiteThreshold &&
centerLeftSensorValue == whiteThreshold &&
centerRightSensorValue == blackThreshold &&
rightSensorValue == blackThreshold) {
  state = EVENT4;
}

else if (leftSensorValue == whiteThreshold &&
centerLeftSensorValue == whiteThreshold &&
centerRightSensorValue == whiteThreshold &&
rightSensorValue == blackThreshold) {
  state = EVENT5;
}

else if (leftSensorValue == blackThreshold &&
centerLeftSensorValue == blackThreshold &&
centerRightSensorValue == blackThreshold &&
rightSensorValue == whiteThreshold) {
  state = EVENT6;
}
```

```
else if (leftSensorValue == whiteThreshold &&
centerLeftSensorValue == blackThreshold &&
centerRightSensorValue == blackThreshold &&
rightSensorValue == blackThreshold) {
  state = EVENT7;
}

else if (leftSensorValue == whiteThreshold &&
centerLeftSensorValue == whiteThreshold &&
centerRightSensorValue == whiteThreshold &&
rightSensorValue == whiteThreshold) {
  state = STATE_REVERSING;
}

else if (leftSensorValue == blackThreshold &&
centerLeftSensorValue == blackThreshold &&
centerRightSensorValue == blackThreshold &&
rightSensorValue == blackThreshold) {
  state = STATE_END;
}

//ESTADOS

switch (state) {

  //////////////ESTADO PARADO\\\\\\\\\\\\\\

  case STATE_STOPPED:
    analogWrite(leftMotorPin, 0);
    analogWrite(rightMotorPin, 0);
    break;

  //////////////ESTADO AVANCE\\\\\\\\\\\\\\

  case EVENT1:
    analogWrite(leftMotorPin, 255);
    analogWrite(rightMotorPin, 255);
    break;

  case EVENT2:
    analogWrite(leftMotorPin, 100);
    analogWrite(rightMotorPin, 200);
```

```
    break;

  case EVENT3:
    analogWrite(leftMotorPin, 60);
    analogWrite(rightMotorPin, 150);
    break;

  case EVENT4:
    analogWrite(leftMotorPin, 200);
    analogWrite(rightMotorPin, 100);
    break;

  case EVENT5:
    analogWrite(leftMotorPin, 150);
    analogWrite(rightMotorPin, 60);
    break;

  //////////////ESTADO GIROS\\\\\\\\\\\\\\

  case EVENT6:
    analogWrite(leftMotorPin, 0);
    analogWrite(rightMotorPin, 255);
    break;

  case EVENT7:
    analogWrite(leftMotorPin, 255);
    analogWrite(rightMotorPin, 0);
    break;

  //////////////ESTADO RETROCESO\\\\\\\\\\\\\\

  case STATE_REVERSING:
    digitalWrite(dirB, LOW);
    digitalWrite(dirA, LOW);
    delay(10);
    analogWrite(leftMotorPin, 150);
    analogWrite(rightMotorPin, 0);
    delay(50);
    analogWrite(leftMotorPin, 0);
    analogWrite(rightMotorPin, 150);
    delay(150);
    digitalWrite(dirB, HIGH);
```

```
        digitalWrite(dirA, HIGH);
        analogWrite(leftMotorPin, 150);
        analogWrite(rightMotorPin, 150);
        break;

      //////////////ESTADO FIN\\\\\\\\\\\\\\

      case STATE_END:

        analogWrite(leftMotorPin, 0);
        analogWrite(rightMotorPin, 0);
        systemOn = false;
        state = STATE_STOPPED;
        break;
    }
  }
}
```

Anexo B

Placa Sensores

Se presenta en las figuras B.1 y B.2 los diagramas esquemáticos que representan de manera gráfica la conexión de los cuatro sensores QRD1114 que van en la placa.

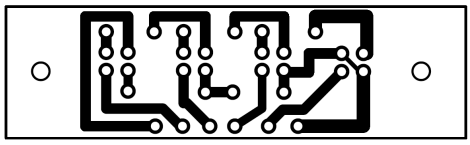


Figura B.1: Esquemático de los sensores.

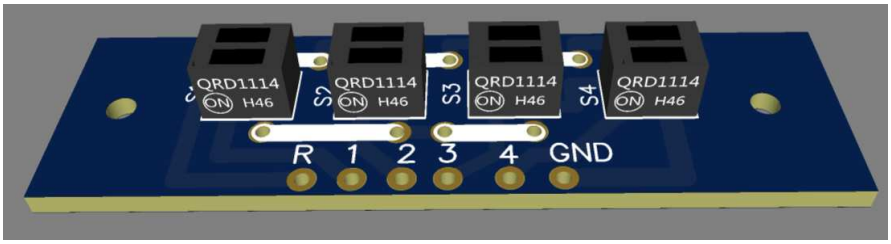


Figura B.2: Diagrama en 3D.

Anexo C

Digitalizador Análogo

Se presenta en las figuras C.1 y C.2 los diagramas esquemáticos de doble cara, y la figura C.3 que representan de manera gráfica la conexión de los componentes de la placa digitalizadora.

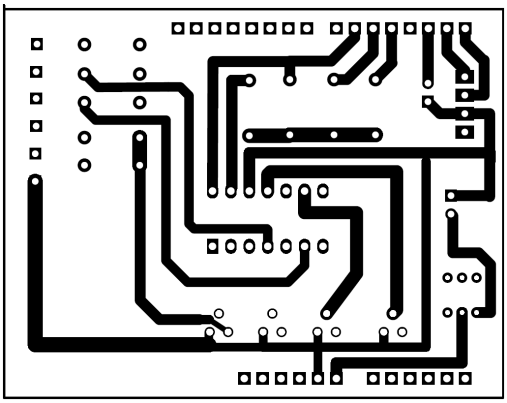


Figura C.1: Esquemático parte 1.

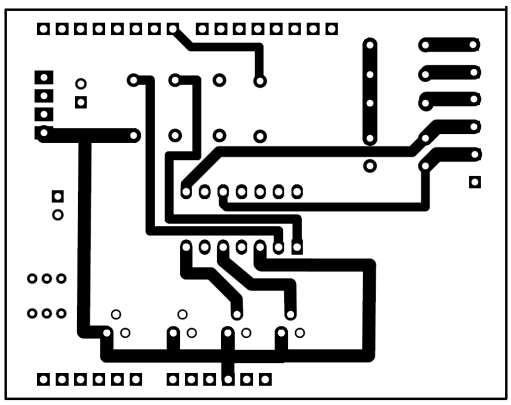


Figura C.2: Esquemático parte 2.

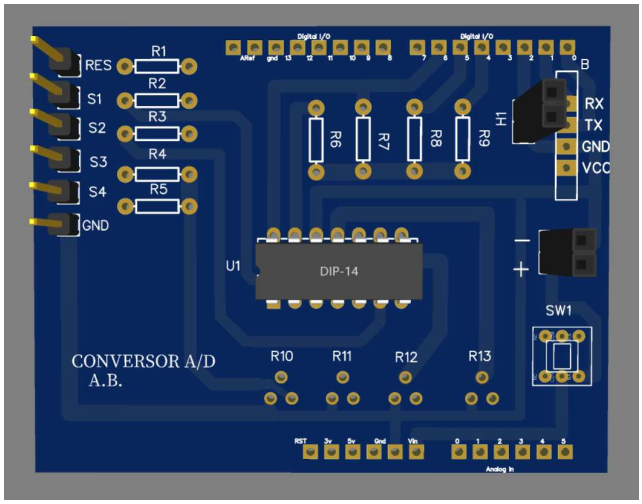


Figura C.3: Diagrama en 3D.