### 

### 

### 

### **Informe Tecnico**

### **Seguidor de Linea**

### 

### 

### 

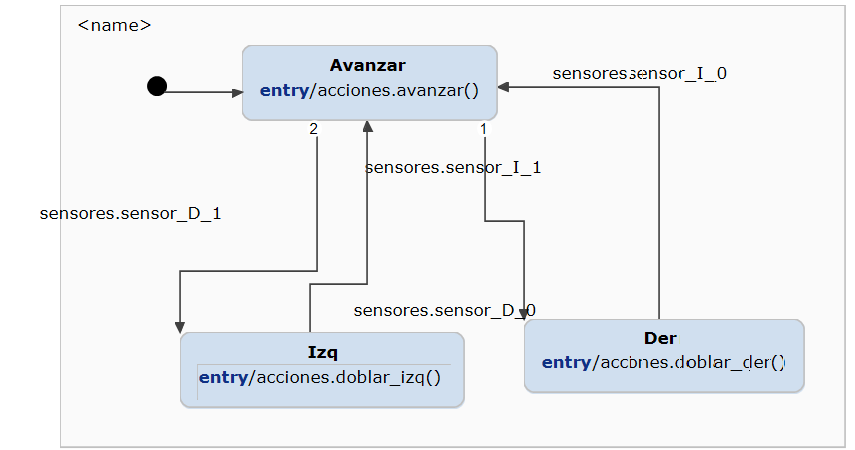
### Nombre: Facundo Palacios

Asignatura: Sistemas Embebidos

Fecha: 16/10/2025

### 

### **a. Diagrama del Modelo de Control**



El sistema de control se implementó utilizando una máquina de estados finitos (Statechart) basada en el formalismo de Harel. El modelo define tres estados operativos principales para gobernar el comportamiento del vehículo en la pista.

### **b. Explicación de los Estados y Transiciones**

El comportamiento del robot se modela con tres estados mutuamente excluyentes, y las transiciones entre ellos son provocadas por los eventos generados a partir de la lectura de los dos sensores infrarrojos.

* **Estado Avanzar**:
  + **Acción**: Este es el estado por defecto. El vehículo mueve ambos motores hacia adelante a la misma velocidad (VELOCIDAD\_AVANCE) para seguir una trayectoria rectilínea.
  + **Condición**: Permanece en este estado mientras ambos sensores se encuentran detectando la línea.
* **Estado Izq**:
  + **Acción**: El motor izquierdo gira en retroceso mientras que el motor derecho gira hacia adelante. Esto provoca un giro cerrado hacia la izquierda para corregir la trayectoria.
  + **Transición**: El sistema entra en este estado cuando el sensor derecho no detecta la línea negra y el izquierdo si. Esto indica que el vehículo necesita corregir su rumbo girando hacia la izquierda. Vuelve al estado Avanzar cuando el sensor derecho vuelve a ver la línea.
* **Estado Der**:
  + **Acción**: El motor derecho gira en retroceso y el motor izquierdo hacia adelante, generando un giro cerrado hacia la derecha.
  + **Transición**: Se activa cuando el sensor izquierdo no detecta la línea negra. Esto significa que el vehículo debe girar hacia la derecha. Regresa al estado Avanzar una vez que el sensor izquierdo vuelve a detectar la línea negra.

### **c. Arquitectura de Hardware**

El hardware se compone de una unidad central de procesamiento, sensores para la percepción del entorno y actuadores para el movimiento del vehículo.

* **Unidad de Control Central**: El cerebro del sistema es la placa de desarrollo **STM32F407G-DISC1**. Esta plataforma se encarga de ejecutar toda la lógica de control, leer los datos de los sensores y enviar los comandos a los motores.
* **Sistema de Percepción**: Consiste en **dos sensores infrarrojos (IR) digitales**. Su función es detectar la línea en la pista. Los sensores se conectan directamente a los pines GPIO del microcontrolador y devuelven un estado lógico bajo (0) al detectar la línea negra y un estado alto (1) sobre el fondo blanco.
* **Sistema de Actuación**: Está formado por **dos motores de corriente continua (DC)** responsables del movimiento y un **driver L298N** para su control. El vehículo utiliza un esquema de **control diferencial**, donde la velocidad y dirección de cada rueda se gestionan de forma independiente para generar movimientos rectos o curvilíneos. El driver L298N actúa como una interfaz de potencia entre el STM32 y los motores, recibiendo tres señales por cada motor:
  + **Dos señales digitales (M0 y M1)** para controlar la dirección de giro (avance, retroceso o frenado).
  + **Una señal PWM (Modulación por Ancho de Pulso)** para regular la velocidad de forma precisa.
* **Fuente de Alimentación**: Una **batería externa** proporciona la energía necesaria para todos los componentes del vehículo.

### **d. Problemas Encontrados y Soluciones Aplicadas**

1. **Errores de Compilación Iniciales**: Durante la integración del código generado por Itemis CREATE con el proyecto de STM32CubeIDE, surgieron múltiples errores de compilación.
   * **Solución**: Se corrigieron errores estructurales en el archivo main.c, como llaves de cierre mal ubicadas y llamadas a funciones con nombres incorrectos (ej. statechart\_run\_cycle en lugar de statechart\_trigger\_without\_event). Además, se ajustó la configuración de los periféricos en el código para que coincidiera con la del hardware.
2. **Comportamiento Errático de los Motores**: En las primeras pruebas, los motores no respondían como se esperaba, girando de forma incorrecta o a velocidades inadecuadas.
   * **Solución**: El problema residía en la configuración del Timer y el PWM. Se ajustaron los valores de **Prescaler** y **Period** en MX\_TIM4\_Init para generar una frecuencia de PWM adecuada. Luego, se calibraron experimentalmente los valores de VELOCIDAD\_AVANCE y VELOCIDAD\_GIRO para obtener un movimiento fluido y controlado.
3. **Falsas Detecciones de los Sensores**: La sensibilidad a la luz ambiental causaba que los sensores dieran lecturas incorrectas, haciendo que el robot se desviara sin motivo.
   * **Solución**: Aunque el diseño se hizo robusto, se implementó una pequeña cubierta o visera sobre los sensores para reducir la interferencia de la luz externa. Adicionalmente, se ajustó el HAL\_Delay(1) en el bucle principal para crear un tiempo de muestreo constante y filtrar lecturas espurias.

### **e. Tiempo de Vuelta Promedio**

El tiempo de vuelta promedio obtenido fue de **17 segundos**.

### **f. Observaciones del Comportamiento**

* El vehículo sigue la línea de manera efectiva, pero muestra un ligero movimiento de "zigzag" en las rectas. Esto es inherente al control por estados, ya que corrige la trayectoria solo después de detectar una desviación.
* En las curvas cerradas, el robot realiza giros bruscos sobre su propio eje. Esta estrategia es efectiva para no salirse de la pista, aunque puede no ser la más rápida.
* El rendimiento es consistente bajo condiciones de iluminación controlada. Se observó que cambios drásticos en la luz ambiental pueden afectar la precisión de los sensores.

**g. Propuesta de mejora**

La principal limitación del modelo de tres estados es su simplicidad. No puede manejar situaciones ambiguas o complejas, como curvas de 90 grados, intersecciones o la pérdida total de la línea. Las siguientes mejoras añaden robustez al software para manejar estos escenarios.

**Implementar un Estado de "Pérdida y Búsqueda"**

Esta es la mejora más crítica. Si el robot se sale completamente de la pista, el modelo actual no tiene un plan de recuperación y se detendría o seguiría recto indefinidamente.

* **Nuevo Estado: Perdido**
  + **Transición de Entrada**: Se entra a este estado desde Avanzar si **ambos sensores detectan blanco** por un período de tiempo predefinido (ej. 200 milisegundos). El uso de un temporizador evita que el robot entre en pánico por pequeñas interrupciones en la línea.
  + **Acción**: En este estado, el robot ejecuta una maniobra de búsqueda. Una estrategia efectiva es hacer que gire sobre su propio eje (ej. motor izquierdo hacia atrás, motor derecho hacia adelante) para "escanear" el área en busca de la línea.
  + **Transición de Salida**: Si durante el giro, uno de los sensores detecta la línea negra, el robot transita inmediatamente al estado de corrección correspondiente (Doblar Izquierda o Doblar Derecha).

**Añadir un Estado para Curvas Cerradas o de 90 Grados**

El modelo actual trata todos los desvíos por igual. Un estado específico para curvas más pronunciadas puede mejorar la navegación.

* **Nuevo Estado: Curva Cerrada**
  + **Transición de Entrada**: Se puede entrar a este estado si ambos sensores detectan blanco simultáneamente. Esto es un claro indicador de que el robot se ha encontrado con una línea perpendicular a su trayectoria (como una curva de 90 grados o una 'T').
  + **Acción**: El robot podría realizar una acción específica, como detenerse momentáneamente y luego realizar un giro pivote muy cerrado hasta que uno de los sensores vuelva a detectar negro.
  + **Transición de Salida**: Una vez que el giro ha posicionado al robot sobre la nueva trayectoria (un sensor en negro y otro en blanco), transita al estado de giro normal (Doblar Izquierda o Doblar Derecha) para terminar de alinearse.

### **Introducir Memoria (Último Estado Conocido)**

Se puede hacer la maniobra de búsqueda en el estado Perdido más inteligente si el robot "recuerda" la última dirección en la que giró.

* **Lógica Mejorada**: En lugar de girar siempre en la misma dirección al perderse, el robot puede girar hacia el lado donde es más probable que esté la línea.
  + **Implementación**: Se utiliza una variable interna en la máquina de estados (ej. ultimoGiro).
  + Si el último estado antes de Perdido fue Doblar Izquierda, en el estado Perdido la maniobra de búsqueda será girar hacia la izquierda.
  + Si el último estado fue Doblar Derecha, la búsqueda se realizará hacia la derecha.
  + Esta simple adición de memoria puede reducir drásticamente el tiempo de recuperación.