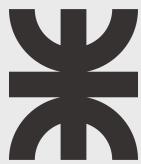


Universidad Tecnológica Nacional FRRo



Tecnologías para la automatización

Comisión 402

Trabajo Práctico N.º 02

Auto sigue línea

Profesores:

Bigatti, Cristian

Garnica, Hernan

Maurino, Gabriel

Integrantes:

47447 - Botali, Santiago

48959 - Mongelos, Manuel

47066 - Gorosito, Adriel

49449 - Fani, Nicolás

50194 - Zallocco Emilio

Fecha realización: 13/07/2024

Índice

Introducción.....	3
Análisis y desarrollo.....	3
Introducción.....	3
Análisis y planificación.....	4
Requerimientos.....	4
Funcionales.....	4
No funcionales.....	4
Análisis del problema.....	5
Componentes.....	5
Diseño del circuito.....	9
Desarrollo.....	10
1. Construcción del chasis del automóvil.....	10
2. Ensamblado de componentes en el chasis.....	11
3. Cableado.....	12
4. Programación.....	12
Controlador PID.....	13
Características del control PID.....	13
Lógica de control.....	13
Lectura de sensores.....	13
Algoritmo de seguimiento.....	13
Funcionamiento del PID en el seguimiento de la línea.....	14
Cálculo del error.....	15
Sintonización.....	17
5. Resultado final.....	17
Pruebas.....	18
Implementación.....	18
Conclusiones.....	18
Bibliografía.....	19
Anexo.....	20
Código.....	20
Tinkercad.....	20
Video y código .ino.....	20

Introducción

El objetivo del trabajo práctico es diseñar y desarrollar un auto seguidor de línea negra por su propia cuenta. Este proyecto de coche de seguimiento automático aprovecha las capacidades de Arduino como unidad de control central. Utilizando un driver de motor, mediante el cual Arduino regula con precisión la velocidad y dirección de los motores, se facilita un movimiento fluido.

Para detectar la línea utilizaremos sensores de reflectancia. Estos sensores emiten luz con un diodo (normalmente infrarrojo) y leen la luz rebotada con un sensor de luz (típicamente un fotodiodo o un fototransistor). Cuando la luz rebota sobre una superficie negra da una lectura diferente que cuando rebota sobre una superficie blanca.

A través de la codificación, se deberá lograr que el automóvil avance en sentido opuesto a los costados donde cesó la línea (ya que esto estaría determinando que se ha desviado hacia el costado).

El proyecto se desarrolla siguiendo un enfoque de ciclo de vida de proyectos, abarcando desde la planificación y diseño del circuito hasta la implementación y pruebas del sistema. El objetivo de implementar este enfoque es integrar y afianzar los temas vistos en la materia de “Administración de sistemas de información”.

Análisis y desarrollo

Introducción

Para comenzar a documentar el paso a paso el desarrollo del auto que sigue líneas, primero comenzaremos con la planificación del mismo. Para planificar de forma eficiente el proyecto y garantizar su éxito, utilizaremos las fases de ciclo de vida de proyectos de sistemas vistas en la materia integradora de cuarto año, “Administración de Sistemas de Información”. Dichas fases son:

1. Análisis y planificación.
2. Diseño del circuito.
3. Desarrollo.
4. Pruebas.
5. Implementación.
6. Control y seguimiento.

Análisis y planificación

Nos basaremos en la documentación provista por la cátedra para entender el funcionamiento del auto y para realizar la compra de los componentes del mismo. El objetivo es conseguir un coche que se maneje por sí mismo, siguiendo una línea (en este caso, de color negro) similar al siguiente:

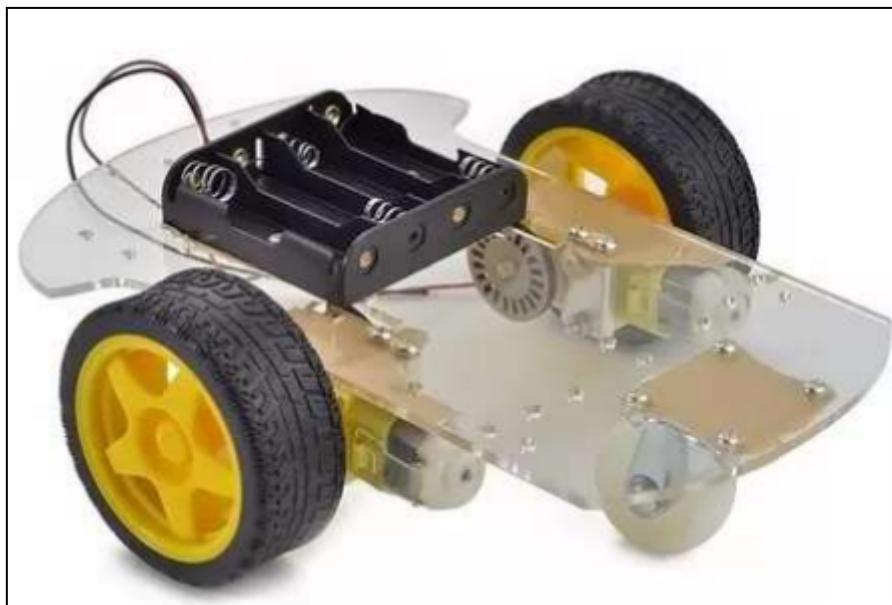


Figura 1: Imagen representativa del chasis del robot sigue líneas.

Requerimientos

Funcionales

- El auto debe identificar una línea negra usando sensores de reflectancia.
- El auto debe seguir la línea con exactitud, ajustando su dirección a medida que va detectando desviaciones.
- El auto debe gestionar la velocidad de los motores para mantenerse sobre la línea.
- El chasis del auto debe estar ordenado y ser agradable a la vista.
- El cableado y los componentes electrónicos no deben implicar un peligro para quien utilice el auto.
- El uso del auto debe ser simple: debe haber un botón de encender o apagar.

No funcionales

- El código fuente y el hardware deben estar documentados y organizados de manera que permitan realizar mantenimiento y actualizaciones de manera eficiente.
- Tanto el diseño del chasis como el código deben ser escalables: se debe permitir una fácil integración de mejoras en el futuro sin requerir una reestructuración completa.

Análisis del problema

Para que el auto pueda moverse por sí mismo, es necesario implementar sensores de reflectancia. Necesitaremos tres: uno (el del medio) debería detectar siempre la línea negra, lo que indica que la está siguiendo correctamente, mientras que los otros dos (uno a la derecha y otro a la izquierda) son los que van a detectar cuando el auto se salga de la posta. Luego de pensarla y analizarla, consideramos que se trata de un problema servo. La justificación es que la entrada inicial es la del sensor, el cual detectará la pista y luego, a medida que el auto avanza, si se desvía, entonces los sensores modificarán la entrada (por medio de la retroalimentación).

Se podría decir que el diagrama de bloques es el siguiente:

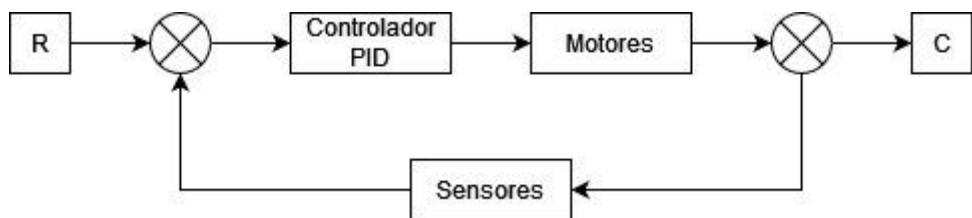


Figura 2: Diagrama de bloques del proyecto.

Nuestro controlador PID será el Arduino UNO. Se encargará de ejecutar y hacer funcionar el código que le proporcionaremos y también de calcular el error teniendo en cuenta los parámetros del PID y la lectura de los sensores. De esta forma, el controlador será capaz de determinar si el auto se salió de la pista.

Por otro lado, la planta serán los motores. En caso de que el controlador PID detecte que el vehículo se desvió, entonces luego de calcular el error, le enviará la señal al motor que debe disminuir la velocidad para ajustar el auto a la línea negra.

La retroalimentación, como se mencionó anteriormente, son los sensores. Por lo tanto, se concluye que no será unitaria.

Componentes

- Microcontrolador Arduino Uno R3.
- Chasis del robot con 2 motores, rueda loca y portapilas.
- 3 sensores de reflectancia.
- Driver de motor L298N.
- Placa de pruebas (protoboard).
- 40 cables para protoboard macho-macho.
- 20 cables para protoboard macho-hembra.
- 4 pilas de 1,5V cada una.
- Cinta aisladora.

A continuación, adjuntamos fotos de los componentes:



Figura 3: Microcontrolador Arduino UNO.

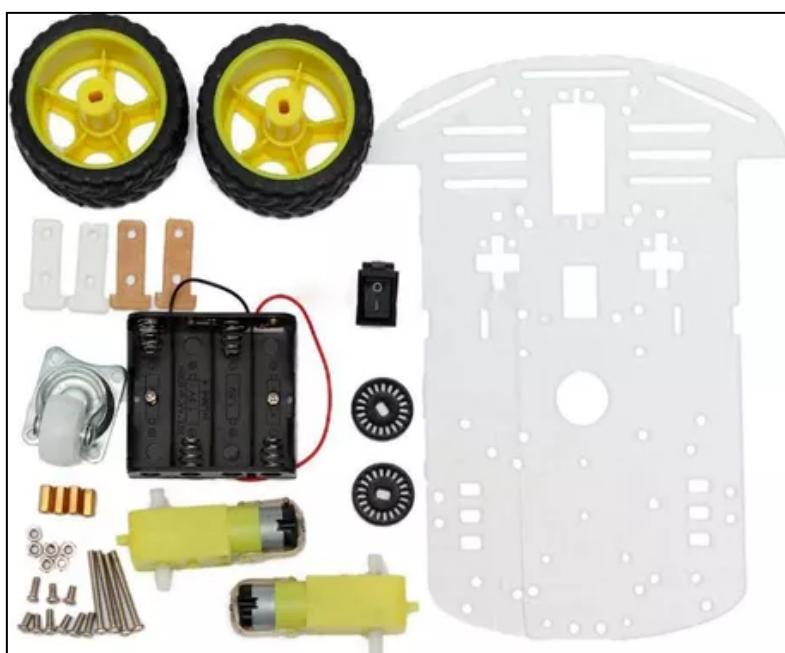


Figura 4: Chasis del vehículo.

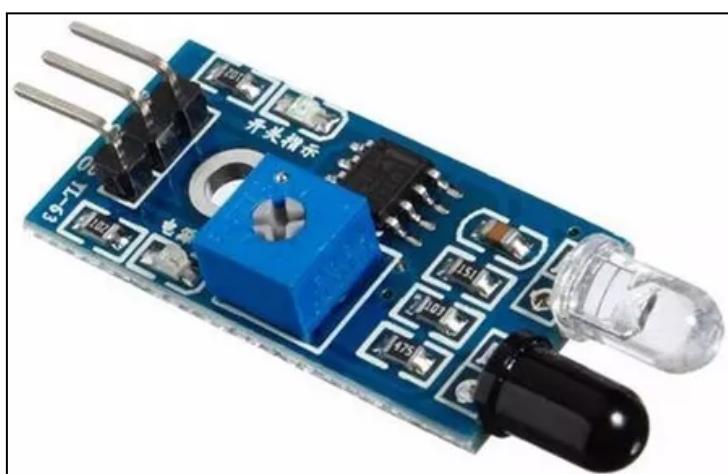


Figura 5: Sensor de reflectancia.

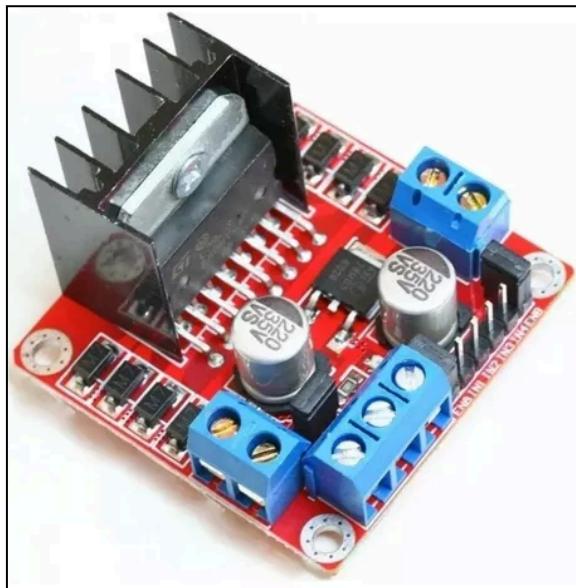


Figura 5: Driver de motor L298N.

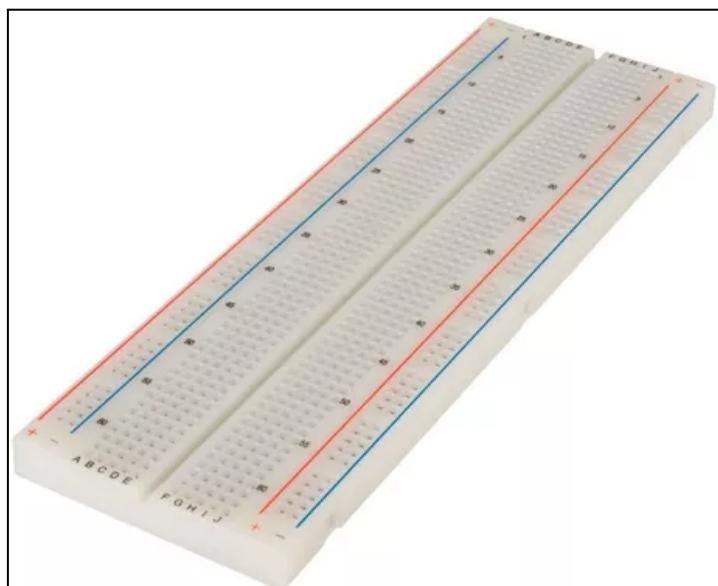


Figura 5: Protoboard.



Figuras 6 y 7: Cables macho (izquierda) y hembra (derecha).



Figura 8: Pila de 1,5 V.



Figura 9: Cinta aisladora.

	Entregado Llegó el 28 de mayo Cables Para Protoboard Macho Macho 20cm 40 Unidades 1 unidad	TODOMICRO Enviar mensaje
	Entregado Llegó el 28 de mayo Placa Compatible Con Arduino Uno R3 Atmel Atmega328 + Ch340g 1 unidad	TODOMICRO Enviar mensaje
	Entregado Llegó el 28 de mayo Kit Chasis Robot Auto Smart Car 2wd 2 Motores + Rueda Loca 1 unidad	TODOMICRO Enviar mensaje
	Entregado Llegó el 28 de mayo Sensor Infrarrojo Seguidor De Linea Tracker Ir Tcr5000 3 unidades	TODOMICRO Enviar mensaje
	Entregado Llegó el 28 de mayo Driver De Motor L298n Dc Desarrollo Arm Avr L298 Impresoras 1 unidad	TODOMICRO Enviar mensaje

Figura 10: Lista de algunos de los componentes comprados por MercadoLibre.

No tuvimos que comprar herramientas (como destornilladores, pinzas pelacables, taladro agujereador, etc) porque ya teníamos, así que fue un gasto menos.

Diseño del circuito

En esta etapa del proyecto realizamos el diseño del circuito, utilizando el software “Tinkercad”. Lo que buscamos fue imitar el funcionamiento del robot sigue líneas, mediante el uso de componentes similares a los utilizados en el proyecto para probar el mismo antes de comenzar a desarrollar el proyecto.

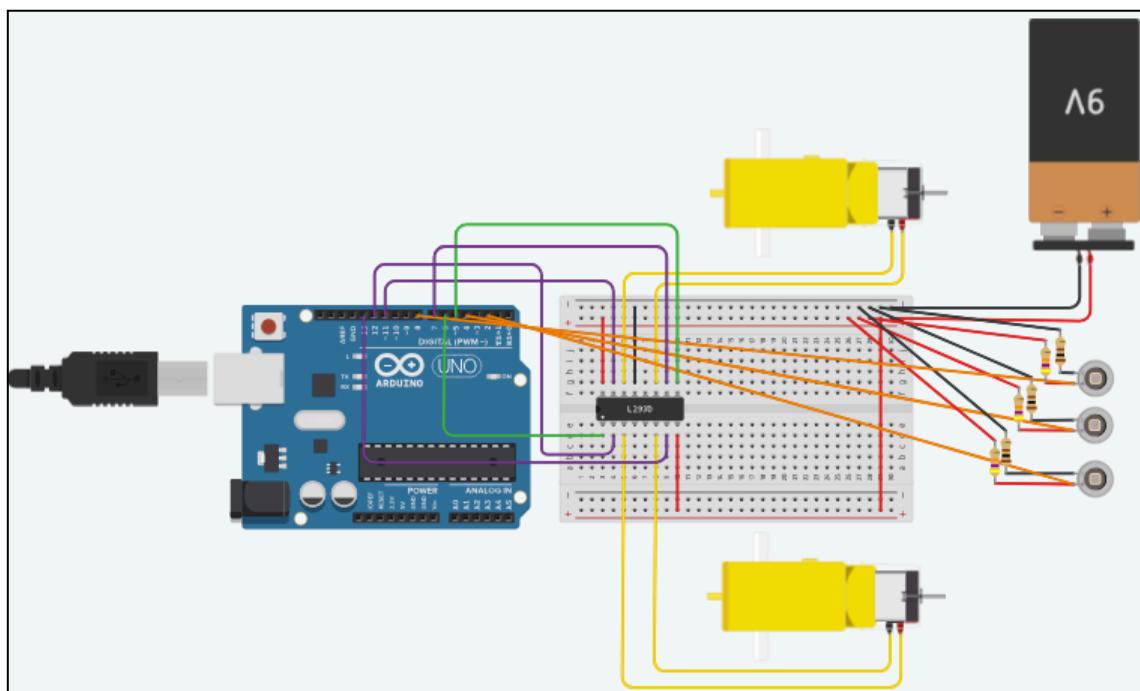


Figura 11: Diseño realizado en Tinkercad.

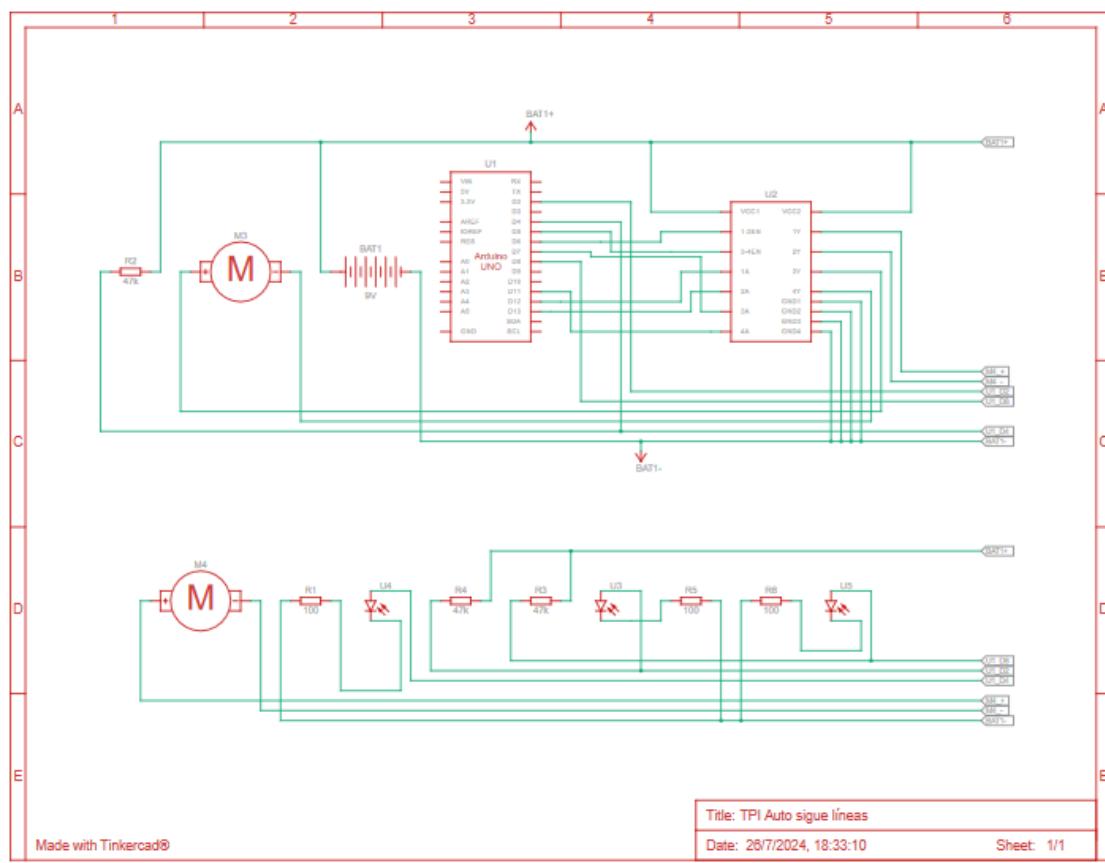


Figura 12: Plano del diseño en Tinkercad.

Cabe aclarar que este diseño es solo para tener una noción para guiarnos a la hora de ensamblar los componentes. Esto es así ya que Tinkercad no provee los mismos elementos que utilizamos de forma física. Por ejemplo, no tiene el Driver de motor L298N. Por lo tanto, es muy probable que el diseño final sea un poco distinto al de la figura 12.

Desarrollo

El paso a paso que seguiremos es:

1. Construir el chasis del automóvil.
2. Ensamblar los componentes en el chasis.
3. Cablear los sensores y los motores.
4. Programar el código para la detección de la línea.

1. Construcción del chasis del automóvil

Para la construcción del chasis primero realizamos la fijación de los motores, ruedas y el soporte de pilas al mismo.

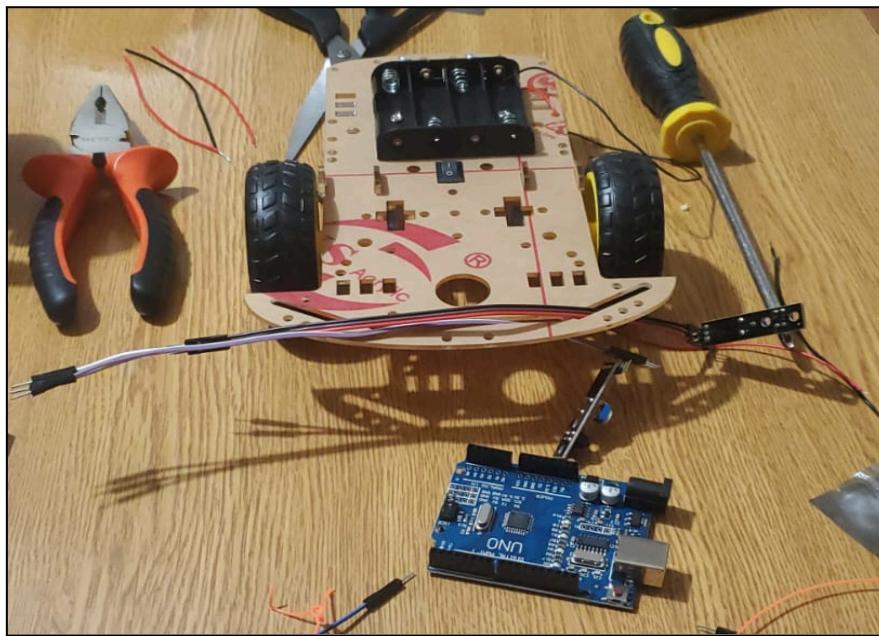


Figura 13: Construcción del chasis.

2. Ensamblado de componentes en el chasis

En el ensamblado lo primero que hicimos fue fijar el driver de motor L298N al chasis y realizar las conexiones necesarias del driver, los motores y las pilas mediante la placa Arduino UNO para corroborar su funcionamiento.

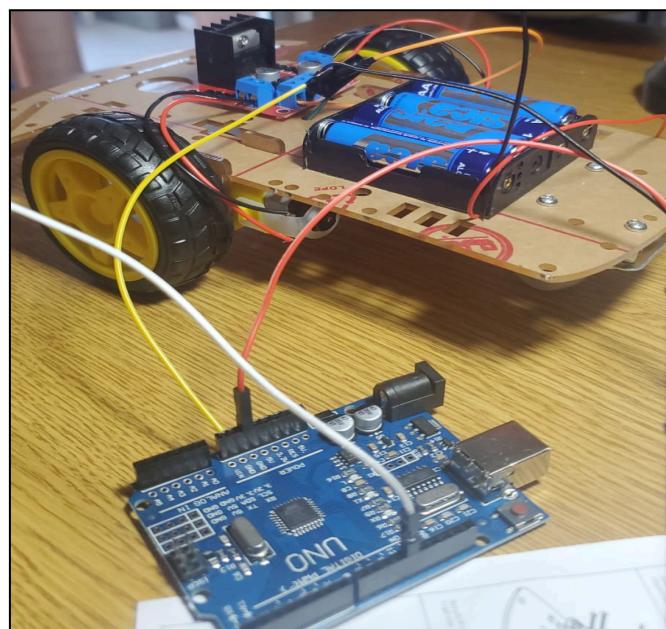


Figura 14: Ensamblado de componentes en el chasis.

3. Cableado

Para el cableado, mediante la incorporación de una protoboard, conectamos los sensores a los pines del Arduino. Los pines de salida del driver se conectan a los motores y los de entrada a los pines de PWM del Arduino. El soporte de pilas se conectan al driver, al interruptor y al Arduino adecuando la alimentación necesaria para el sistema.

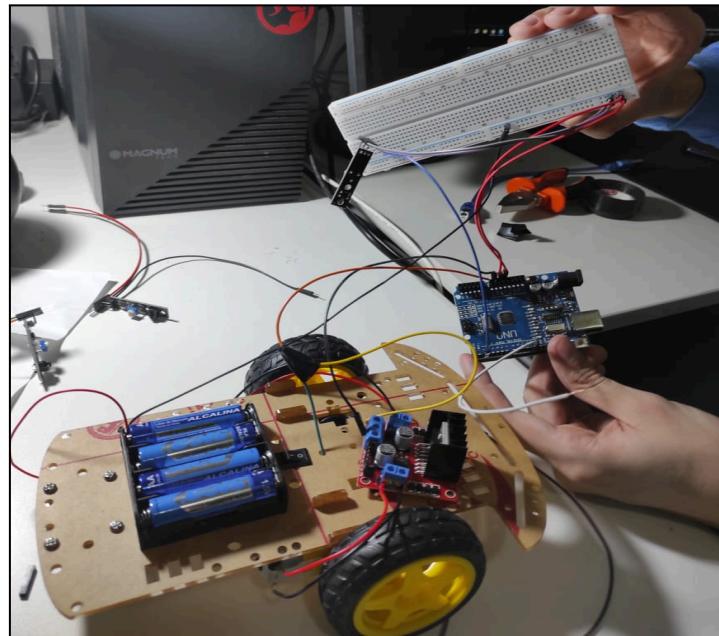


Figura 15: Incorporación de la protoboard.

4. Programación

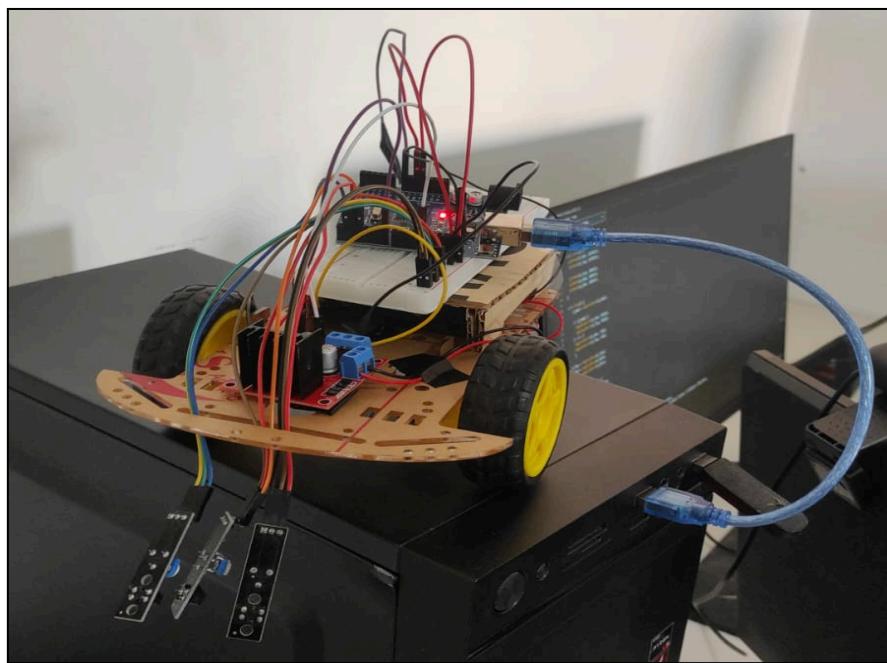


Figura 16: desarrollo e implementación del código en Arduino.

Controlador PID

El controlador PID (Proporcional-Integral-Derivativo) es un mecanismo de control utilizado para mantener una variable en un valor deseado ajustando la salida del sistema. En este proyecto el control PID ajusta la dirección del automóvil para seguir la línea de manera eficiente.

- **Componente Proporcional (P):** esta componente es directamente proporcional al error actual. Si el automóvil se desvía de la línea, la componente P corrige el error en proporción a su magnitud.

$$P = K_p \cdot Error$$

- **Componente Integral (I):** esta componente tiene en cuenta la acumulación del error a lo largo del tiempo.

$$I = I + Error$$

- **Componente Derivativa (D):** esta componente es proporcional a la tasa de cambio del error. Ayuda a prever la dirección del error y ajusta la corrección en consecuencia.

$$D = Error - ultimoError$$

El control PID combina estos tres componentes para calcular una corrección precisa.

$$velocidad = P \cdot K_p + I \cdot K_I + D \cdot K_D$$

Características del control PID

La **componente proporcional** permite que el automóvil reaccione rápidamente a cualquier desviación de la línea ajustando su dirección inmediatamente. La **componente integral** ayuda a eliminar el error en estado estacionario, asegurando que el automóvil mantenga su curso sobre la línea a lo largo del tiempo. La **componente derivativa** aumenta el amortiguamiento del sistema, lo que reduce las oscilaciones y mejora la respuesta transitoria. La acción combinada del PID proporciona un control estable del sistema.

Lógica de control

Lectura de sensores

El Arduino realiza lecturas continuas de los sensores de reflectancia para determinar la ubicación de la línea en relación con el automóvil. Estos sensores detectan cambios en la reflectancia del suelo, permitiendo identificar si el automóvil está sobre la línea o se ha desviado hacia un lado.

Algoritmo de seguimiento

Con base en las lecturas de los sensores, el Arduino ajusta la velocidad y dirección de los motores para seguir la línea. Este ajuste se realiza utilizando un algoritmo de control Proporcional Integral Derivativo (PID), el cual fue explicado previamente.

Funcionamiento del PID en el seguimiento de la línea

El algoritmo PID toma la desviación de la línea detectada por los sensores y ajusta los motores para minimizar este error. De esta manera, si el automóvil se desvía a la derecha, el controlador PID disminuirá la velocidad del motor derecho y aumentará la del motor izquierdo para corregir la trayectoria, y viceversa para desvíos a la izquierda.

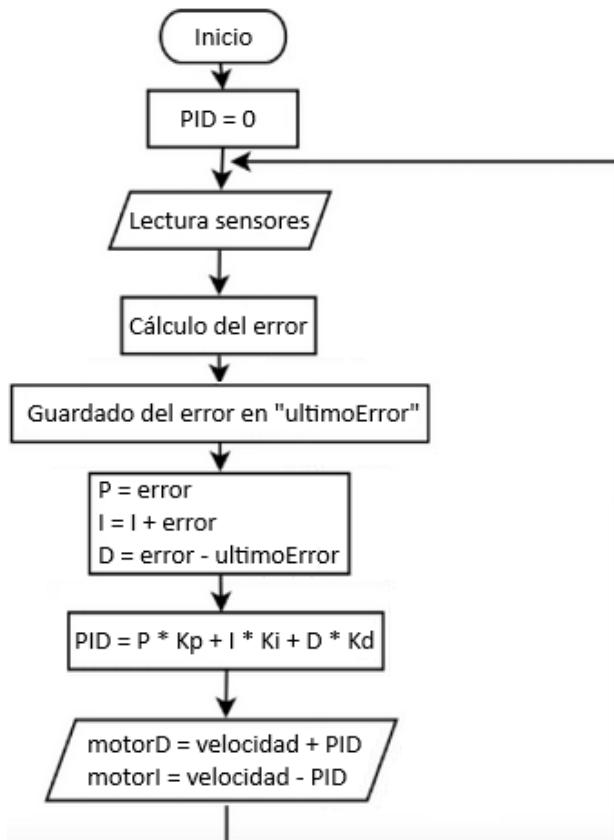


Figura 17: diagrama del funcionamiento del PID.

Explicación de cada bloque:

1. Inicio: inicializa el valor del PID a 0, preparando el sistema para empezar el control.
2. Lectura sensores: recoge los datos de los sensores que detectan la posición del auto.
3. Cálculo del error: determina la desviación del auto respecto a la línea, calculando el error.
4. Guardado del error: guarda el error calculado en la variable "ultimoError" para usarlo en el siguiente ciclo de control.
5. Cálculo de P, I y D: a P le asigna el valor del error actual, a I le acumula el error (sumando el error actual al acumulado previamente) y a D le asigna la diferencia entre el error actual y el error anterior (ultimoError).
6. Cálculo de PID: calcula el valor total del PID, según la fórmula descrita anteriormente.
7. Ajuste de motores: ajusta las velocidades del motor derecho (motorD) e izquierdo (motorI) utilizando el valor del PID para corregir la trayectoria del automóvil.

Cálculo del error

Los sensores registran la posición del automóvil en relación con la línea, pero debemos decidir cómo usar esos datos para crear una señal de error. Para ello, necesitamos definir una dirección positiva y una negativa. En nuestro caso, consideramos la posición positiva más grande hacia la derecha, la intermedia en el medio y la izquierda con un valor de 0. Con esto lo que logramos es que al detectar una línea derecha, el error será negativo y si se detecta una izquierda, el error será positivo y en caso de solo detectar la línea con el sensor del medio, el error será 0.



Figura 18: Detección por parte del sensor del medio.

El sensor del medio detecta la línea negra, asignando 1000 a la posición, logrando un error = 0 y ocasionando que el auto siga derecho.



Figura 19: Detección por parte del sensor izquierdo.

El sensor izquierdo detecta la línea negra, asignando 0 a la posición, logrando un error = 1000 y generando una corrección hacia la derecha.



Figura 20: Detección por parte del sensor derecho.

El sensor derecho detecta la línea negra, asignando 2000 a la posición, logrando un error = -1000, generando una corrección hacia la izquierda.

Además, puede darse casos en que hayan lecturas por parte de dos sensores:

- Si el sensor de la izquierda y el del medio detectan la línea negra, se recibe una detección igual a 500 y luego, el error también será de 500.
- Si el sensor de la derecha y el del medio detectan la línea negra, se recibe una detección igual a 1500 y luego, el error será de -500.

En la siguiente figura se explica el comportamiento de la velocidad:

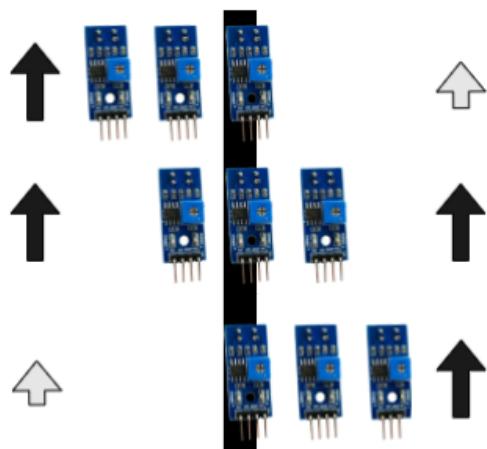


Figura 21: Detección por parte del sensor derecho.

Sintonización

Una vez que corroboramos que el robot era capaz de seguir una línea recta durante un tiempo considerable, nos dimos cuenta que era hora de ajustar los parámetros del controlador PID para conseguir una mejor corrección y un movimiento más fluido.

A esto se le llama sintonización y consiste en ir variando los parámetros hasta conseguir la solución más óptima. Sin embargo, hay que tener en cuenta que para realizar la sintonización no hay ningún método, sino que es prueba y error. Por lo tanto, estuvimos probando durante horas cuáles eran los valores que mejor se ajustaban a nuestro auto.

Finalmente, los parámetros elegidos son:

- $K_p = 10$
- $K_I = 0.01$
- $K_D = 10$

(Nótese el valor pequeño de K_I , lo que casi hace que se trate de un Controlador PD).

5. Resultado final

Finalmente, agregamos una cobertura desplegable de cartón a las pilas, fijando la protoboard y la placa de Arduino encima de ella. Además, agregamos una base frontal para los sensores, logrando que se sitúen más cerca del suelo y obteniendo una mejor respuesta del sistema.

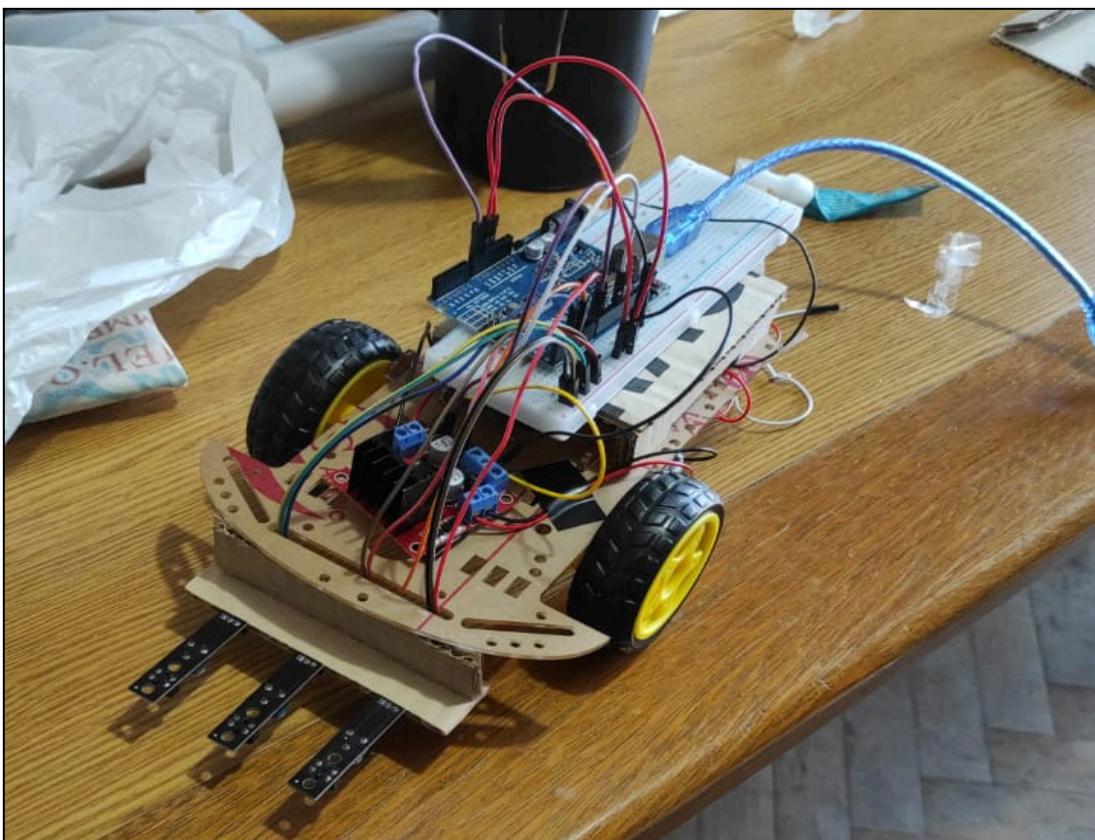


Figura 22: Resultado final del auto sigue líneas.

Pruebas

Para poder testear que el auto esté funcionando correctamente, modificamos el código y agregamos unos `Serial.println()` con el objetivo de imprimir en consola el movimiento realizado por el auto. De esta forma, corroboramos que las ruedas tengan el comportamiento esperado.

Por ejemplo, en la función de posición inicialmente probamos con el siguiente código:

```
int posicion(int i, int c, int d) {  
    if (i == LOW && c == HIGH && d == LOW) Serial.println("Va hacia adelante");  
    if (i == LOW && c == LOW && d == HIGH) Serial.println("Va hacia la derecha");  
    if (i == HIGH && c == LOW && d == LOW) Serial.println("Va hacia la izquierda");  
    return 1000
```

Implementación

Se envía como anexo el video del auto funcionando, siguiendo un circuito realizado en el piso, en donde se puede ver como realiza correcciones cuando detecta desvíos para mantenerse en la pista.

Conclusiones

El desarrollo del auto seguidor de línea permitió aplicar y consolidar los conocimientos teóricos y prácticos vistos a lo largo de la materia utilizando Arduino. A través de un enfoque sistemático y metodológico, se logró diseñar e implementar un vehículo autónomo capaz de seguir una línea negra utilizando sensores de reflectancia. Las pruebas realizadas confirmaron el correcto funcionamiento del sistema.

Por otro lado, la integración con las fases de ciclo de vida de proyectos vistos en la materia Administración de sistemas nos permitió trabajar de forma más organizada y cómoda. Si no hubiésemos aplicado este enfoque, habríamos perdido tiempo pensando qué hacer y cómo proceder en cada etapa.

Bibliografía

Bibliografía oficial de la materia “Tecnologías para la Automatización”.

Bibliografía oficial de la materia “Administración de Sistemas de Información”.

Kumar, A. (2019). How to Program a Line Following Robot. Robot Research Lab.

Disponible en: <https://robotresearchlab.com/2019/02/12/how-to-program-a-line-following-robot/>

Şen, H. (2019). PID Line Follower Tuning. Robot Research Lab. Disponible en:

<https://robotresearchlab.com/2019/02/16/pid-line-follower-tuning/>

Anexo

Código

Link al repositorio de Github:

<https://github.com/adrielgorosito/TPI-TDC-Auto-sigue-lineas/blob/main/Codigo-Auto.ino>

Tinkercad

Link al Tinkercad:

<https://www.tinkercad.com/things/d3vMJORg5tw-copy-of-tpi-auto-sigue-lineas/editel?sharecode=Q8B8KqDStns1HW3QTz7DBkTjX1kIlcKBZ-2m93dS78I>

Plano del Tinkercad:

<https://github.com/adrielgorosito/TPI-TDC-Auto-sigue-lineas/blob/main/Plano%20Tinkercad.pdf>

Video y código .ino

Video:

<https://drive.google.com/file/d/1DKDm68jxo6q7aLgM1eW8Ua9WHGoHQgXw/>

Archivo .ino:

<https://drive.google.com/file/d/1dkHKNBDvhIKLGTiAwaCfUqhFcfRhUaqd/>