

FUTUROS INGENIEROS **RED MANACHINE**

PROYECTO
“Julián”



Liceo
Los Robles

Integrantes:

Cano Barros, Juan Diego
Galban Franco, Samuel José
Rodríguez Guerra, Ángel Saúl



Delegación
Venezolana



Octubre de 2023

CONTENIDO

Introducción

1. Formación del equipo de trabajo

2. Proceso del diseño

2.1 Sistema Mecánico

2.1.1 Dirección

2.1.2 Conducción

2.1.3 Diseño de chasis

2.2 Electrónico

2.2.1 Sensores

2.2.2 Control de velocidad / dirección

2.3 Programación

2.3.1 Diagrama de flujo

3. Estrategias

4. Discusión de códigos

5. Entradas del diario

5.1 Cronología - desafíos y soluciones.

6. Oportunidades de mejoras.

Conclusión

Apéndices

Apéndice A - Vistas del carro robot

Apéndice B - Videos de rendimiento

Apéndice C - Especificación de los sensores

Apéndice D - Especificación de los actuadores

Apéndice E - Código del Arduino.

INTRODUCCIÓN

La robótica es una ciencia que aglutina varias disciplinas con el objetivo de diseñar máquinas programadas para realizar tareas de forma automática, la misma no sólo implica el desarrollo de robots, sino también su diseño, programación, producción y aplicación.

Además permite desarrollar destrezas y habilidades cognitivas y sociales como la resolución de problemas, el trabajo en equipo, el aprendizaje independiente y la comunicación.

World Robotic Olympiad (WRO) Es una competencia celebrada por primera vez en **2004** cuya misión sería ayudar a los jóvenes a desarrollar su creatividad y sus habilidades para resolver problemas de una manera divertida y atractiva. Para ello, se organizaron competiciones de robótica en cuatro categorías diferentes para estudiantes entre **8 y 19** años.

La categoría en la que se hace referencia en este documento es futuros ingenieros, la cual cuenta con dos retos a cumplir. El reto número uno consiste en que el robot debe completar **tres (3)** vueltas en la pista con ubicaciones aleatorias de las paredes en la pista.

El reto número dos consiste en que el robot debe completar **tres (3)** vueltas en la pista con señales de tráfico verdes y rojas colocadas al azar dentro de las paredes interiores de la pista. Para el cumplimiento del reto se desarrolla un robot, que sea capaz de afrontar las condiciones planteadas, a nivel de diseño y funcionamiento definidas en las bases de la competencia.

Siguiendo la estructura de los criterios de evaluación de la categoría, este documento describe en detalle la solución diseñada para el robot **"Julián"**, el cual representa al equipo **"Red Machine"**, definiendo los criterios aplicados en la parte mecánica específicamente la dirección, conducción y diseño del chasis, así como también, se explica todo el sistema electrónico que lo compone como son los sensores y el controlador de velocidad y dirección, ya finalmente se expone la programación del mismo, mediante el sistema de los módulos de los códigos utilizado, para lograr el cumplimiento de los retos número uno y dos.

1. FORMACIÓN DEL EQUIPO DE TRABAJO

El equipo **Red Machine** surge inicialmente de un grupo de amigos a los que se les presentó la oportunidad de iniciar en el mundo de la robótica, si bien algunos de estos tenían conocimientos previos, comenzaron desde lo más básico hasta luego hacer prototipos mucho más funcionales y avanzados.

El equipo para la competencia está conformado por:

Samuel José

Galbán Franco



Juan Diego

Cano Barros



Ángel Saúl

Rodríguez Guerra



En el inicio de esta travesía, el equipo de trabajo se decidió por el nombre **"Robstar"** el cual fue creado mediante una unión de "Robótica" y "Tareas". Luego, se presentó un nuevo reto, en este caso era una competencia de robótica, denominada copa ka'í, la cual al ser un evento colegial se decidió que el nombre de **"Maquinaria roja"** encajaría perfectamente con el equipo en representación a la institución, debido a una tradición en todas las competencias de cualquier índole que participaba el Liceo Los Robles era conocido como tal. Finalmente, al comenzar en la **WRO** el equipo decidió que la mejor opción de nombre sería **"Red Machine"** en honor a **"Maquinaria Roja"**, para la construcción del robot **"Julián"**

Para guardar y compartir este proceso se decide abrir medios de redes sociales

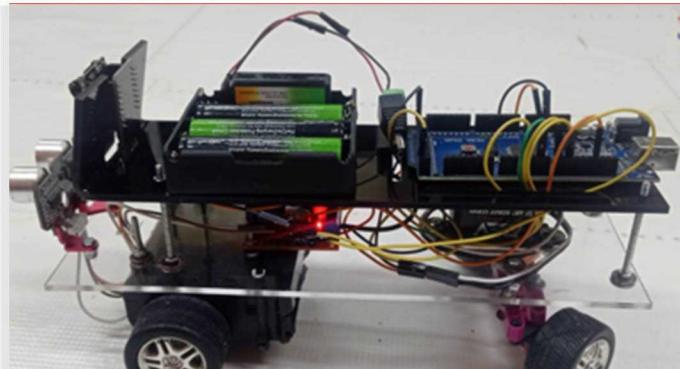
 @redmachine_vzla

 @REDMACHINE.WRO.

2. PROCESO DEL DISEÑO

A continuación se presenta el proceso de diseño de cada una de las fases del proyecto, describiendo detalladamente cada uno de los sistemas mecánico, electrónico y programación.

2.1. SISTEMA MECÁNICO



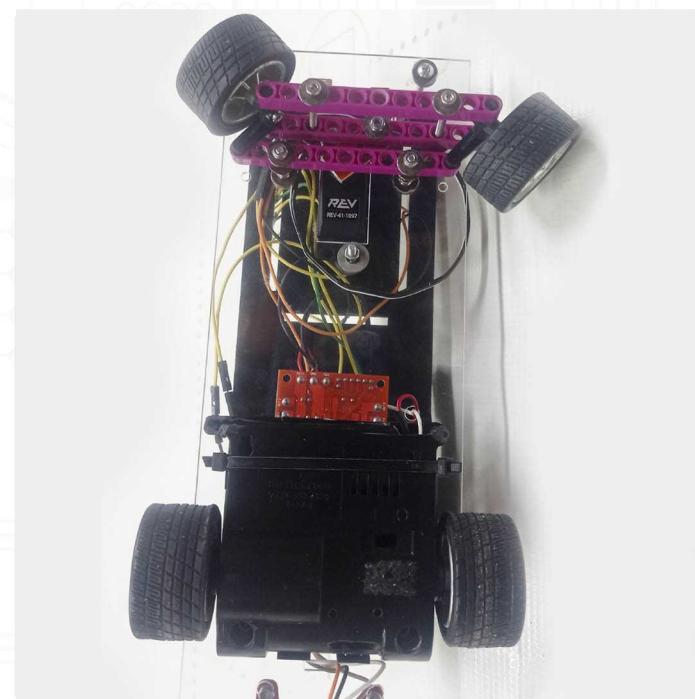
El sistema mecánico fue tomado de un carro a control remoto modelo **Ford Mustang**, del cual se extrajeron los elementos de conducción, y los mismos fueron acoplados a los motores presentes en el sistema electrónico.



2.1.1. DIRECCIÓN

El subsistema de dirección del robot se encuentra en la parte trasera de éste, para así proporcionar un mayor radio de giro, fue creado con piezas de lego de un kit de robótica denominado **Kit Lego Spike Prime**, uniendo estas piezas con las ruedas de un carro control remoto (**RC**) modelo **Ford Mustang**, al cual se le adaptó un servomotor de la marca **Rev Robotics**.

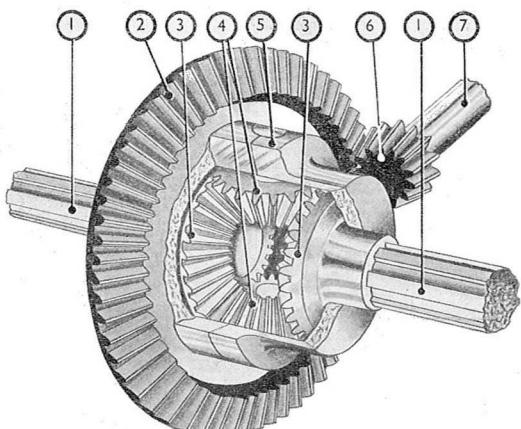
Posteriormente fue programado a través de un arduino mega **2560**, conectado el servomotor a los pines **GND, 5V** y **4** del arduino como puede visualizarse en la siguiente imagen.



2. PROCESO DEL DISEÑO

2.1.2. CONDUCCIÓN

El subsistema de conducción del robot fue tomado de un carro a control remoto modelo **Ford Mustang**. Se utilizó este sistema debido a que portaba un motor y una caja de cambios, y proporcionaba el torque y la velocidad necesaria para el robot.

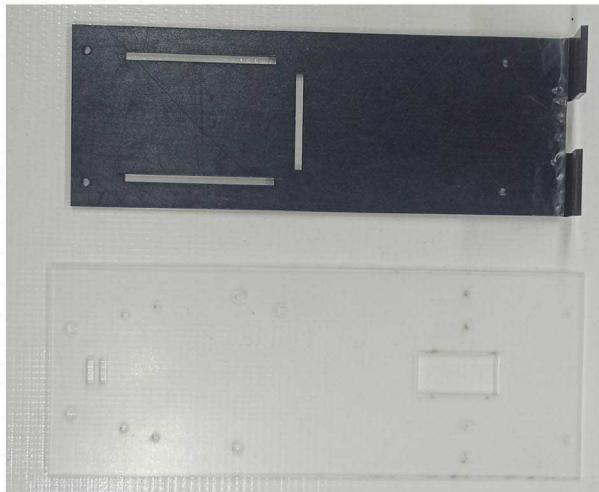


Este funciona con un juego de engranajes, el cual le permite al robot girar a diferentes velocidades sin perder agarre, dicho mecanismo fue ubicado en la parte delantera del robot. El sistema es alimentado a través de un puente H doble modelo **L298n** por medio de un pack de tres baterías de **3,7V**.

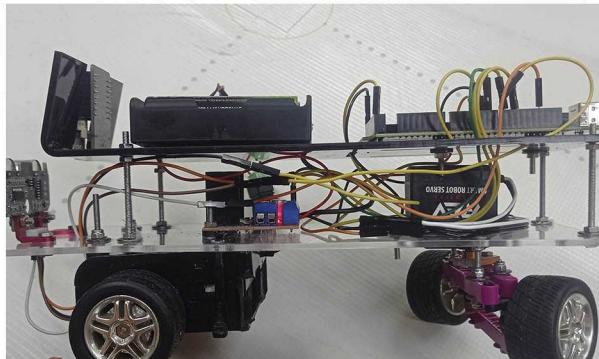
2. PROCESO DEL DISEÑO

2.1.3. DISEÑO DE CHASIS

La base del chasis fue construida con acrílico, el cual fue cortado y perforado en base a las necesidades del cableado y el ensamblaje de las piezas utilizadas como puede visualizarse en la siguiente imagen.



El robot cuenta con dos pisos ambos construidos con acrílico, tal como se muestra en la imagen a continuación



En el “**primer piso**” del robot fueron ensamblados los sistemas de tracción y dirección (servomotor), el sensor de ultrasonido, el sensor de color **RGB 34725**, dos interruptores uno para el encendido y el otro para iniciar y puente H doble modelo **L298n**.

Mientras que en el “**segundo piso**” se encuentra el pack de baterías, la batería de 9V, la cámara **ESP 32-cam**, arduino mega **2560**, y el conector para las baterías.

2. PROCESO DEL DISEÑO

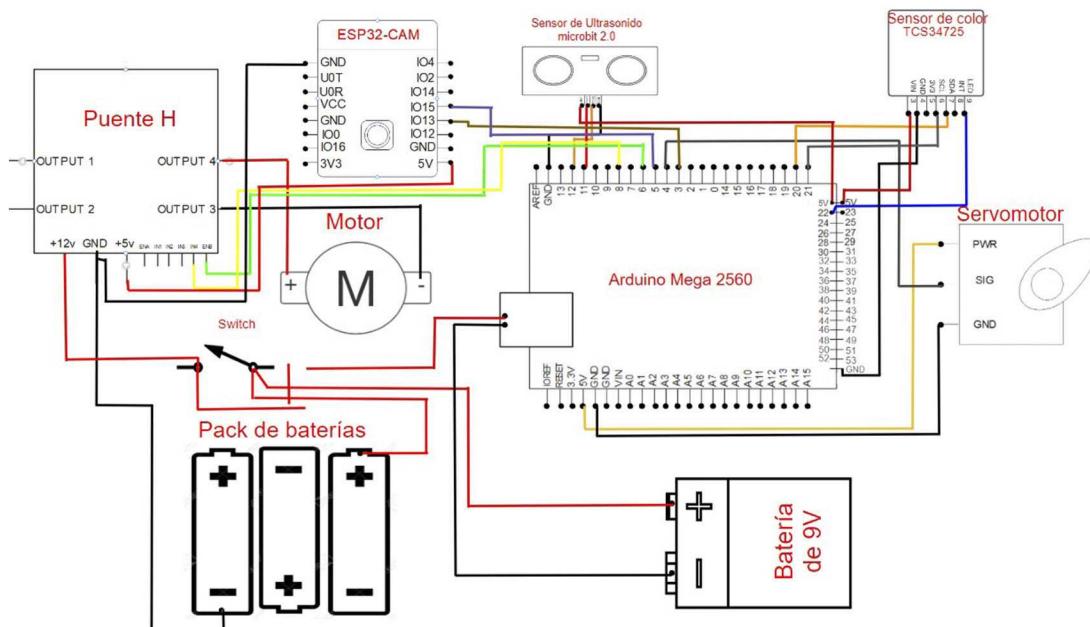
2.2. ELECTRÓNICO

El sistema electrónico del robot está basado en un micro controlador **Arduino Mega 2560** con el que se controlan, según el diseño original, tres sensores y dos actuadores de la siguiente manera:

- Un sensor de color modelo: **RGB TCS-34725**.
- Un motor DC tipo: Motor eléctrico **RS390, 12V**.
- Un sensor de proximidad ultrasónico modelo: **Hc-sr04**.
- Un servomotor modelo: **REV Robotics Smart Robot Servo (SRS)**.
- Una cámara web modelo: **ESP32-cam**.



A continuación se presenta el diagrama de conexión de los componentes que se utilizaron en el presente diseño



Cada uno de los sensores están especificados en el Apéndice C - Especificación de los Sensores y el Apéndice D - Especificación de los actuadores.

2. PROCESO DEL DISEÑO

2.2.1. SENSORES

Microbit 2.0

El robot cuenta originalmente con un sensor de ultrasonido.



Este es utilizado para medir la distancia existente entre el robot y las paredes de la pista de juego. Este sensor se encuentra ubicado en la parte frontal del robot, y es utilizado para medir la distancia existente entre el robot y la pared, para así determinar cuándo éste debe cruzar.

El sensor Microbit 2.0 detecta distancias desde **2 cm** hasta **450 cm** con una excelente precisión.

Este sensor cuenta con cuatro pines:

VCC (voltaje de corriente directa): Es la entrada de alimentación, este recibe **5V**.

TRIG (trigger) : Pin que envía el pulso de ultrasonido.

ECHO (eco): Pin que calcula el tiempo que se tarda el pulso en regresar al ultrasonido, para así determinar la distancia.

GND (ground): Pin que funciona como conector a tierra.

El funcionamiento del sensor es el siguiente: el emisor piezoelectrico emite ocho pulsos de ultrasonido(**40KHz**) luego de recibir la orden en el **Pin TRIG**.



Las ondas de sonido viajan en el aire y rebotan al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoelectrico, luego el **Pin ECHO** cambia a Alto (**5V**) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada.

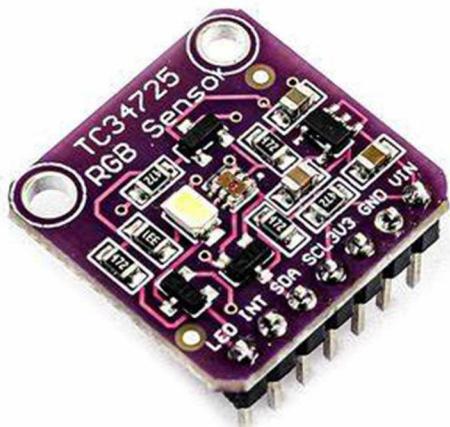
El tiempo del pulso eco es medido por el micro - controlador y así se puede calcular la distancia al objeto.



2. PROCESO DEL DISEÑO

RGB TCS34725

El robot cuenta con un sensor de color.



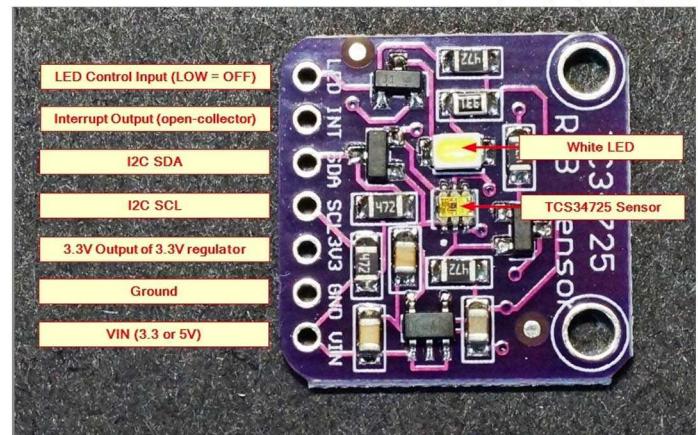
Este es utilizado para identificar los colores de las líneas naranja y azul que se encuentran en las esquinas de la pista, para así determinar en qué sentido debe cruzar el robot.

El sensor **RGB TCS34725** cuenta con los siguientes pines:

VIN (voltaje de entrada): es una de las fuentes de alimentación del módulo que recibe **5V**.

3V3: Fuente de alimentación del módulo que recibe **3,3V**.

GND (ground): pin de tierra del dispositivo.



TCS34725

Es un convertidor de luz a digital de color basado en **bus I2C** con un **filtro IR** que proporciona un retorno digital de valores de detección de luz roja, verde, azul (**RGB**) y clara.

SCL: empleado como señal de reloj **I2C**.

SDA: utilizado para el intercambio de datos **I2C**.

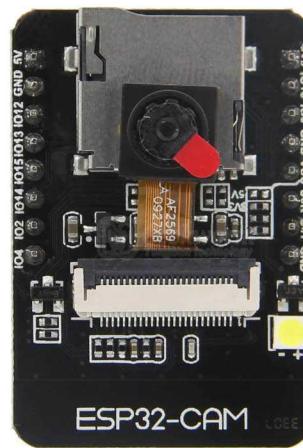
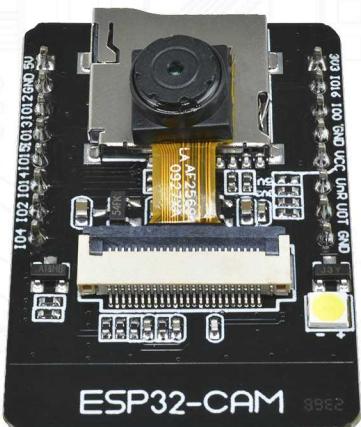
INT: Ajustar dirección **I2C**.

LED (Light-emitting diode): pin que funciona para el encendido del **LED** del módulo Active Low.

2. PROCESO DEL DISEÑO

ESP32 - CAM

Se utiliza esta cámara para poder identificar el color de los semáforos (**rojo o verde**), para así poder esquivarlos correctamente. **RGB**



Esta cámara cuenta con los siguientes pines:

5V	GPIO 14
3V	GPIO 3
GND	GPIO 2
GPIO 16	GPIO 1
GPIO 12	GPIO 4
GPIO 0	GND
GPIO 13	
GND	
GPIO15	
3.3V/5V	

La cámara cuenta con tres pines de tierra y dos pines para alimentación: ya sea **3,3V** o **5 V**.

GPIO 1 y **GPIO 3** son los pines seriales con los que cuenta esta cámara. Necesita estos pines para cargar código en su placa. Además, **GPIO 0** también juega un papel importante, ya que determina si el **ESP32** está en modo intermitente o no.

Los siguientes pines están conectados internamente al lector de tarjetas micro SD:

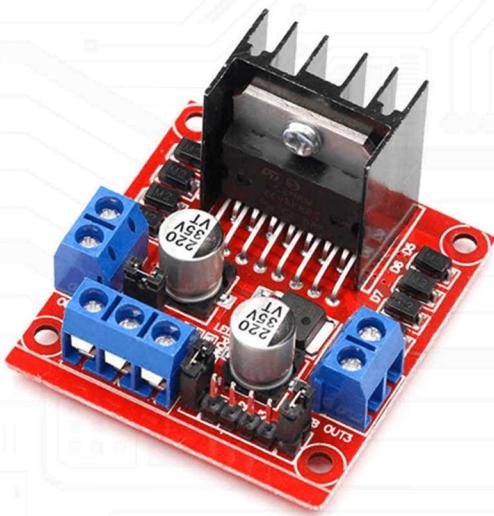
- GPIO 14:** CLK
- GPIO 15:** CMD
- GPIO 2:** Datos 0
- GPIO 4:** Datos 1
(también conectado al LED integrado)
- GPIO 12:** Datos 2
- GPIO 13:** Datos 3

2. PROCESO DEL DISEÑO

2.2.2. CONTROL DE VELOCIDAD / DIRECCIÓN

La Velocidad

Es controlada a través de un puente H doble modelo **L298N**, el cual, gracias a una señal analógica enviada por el **Arduino Mega 2560**, es capaz de controlar el voltaje que se emite a los motores, así variando y controlando la velocidad necesaria para que el robot cumpla el reto.



La Dirección

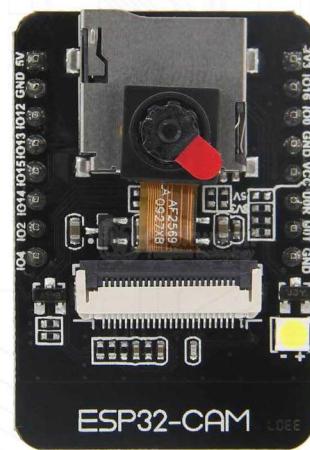
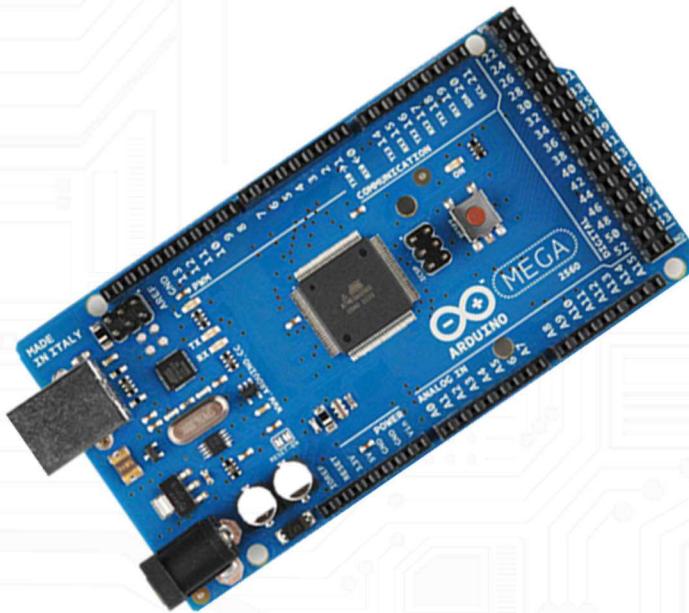
Es controlada con un **smart robot servo** creado por la empresa **Rev Robotics**, el cual a través de una señal analógica enviada por el **Arduino mega 2560**, controla su radio de giro para así lograr un cambio de dirección óptimo.



2. PROCESO DEL DISEÑO

2.3. PROGRAMACIÓN

El robot dispone de dos dispositivos programables:



2.3.1. El **Arduino Mega 2560** que se utiliza para el control general de los dispositivos del robot.

En cuanto a los algoritmos mostrado en el diagrama de flujo, sólo se está presentando el utilizado en la sección `loop()`, del arduino Mega ya que en la sección `setup()`, sólo se realizaron configuraciones y declaraciones.

2.3.2. La **cámara ESP32 Cam** cuya función es fundamentalmente la detección de colores de los semáforos.

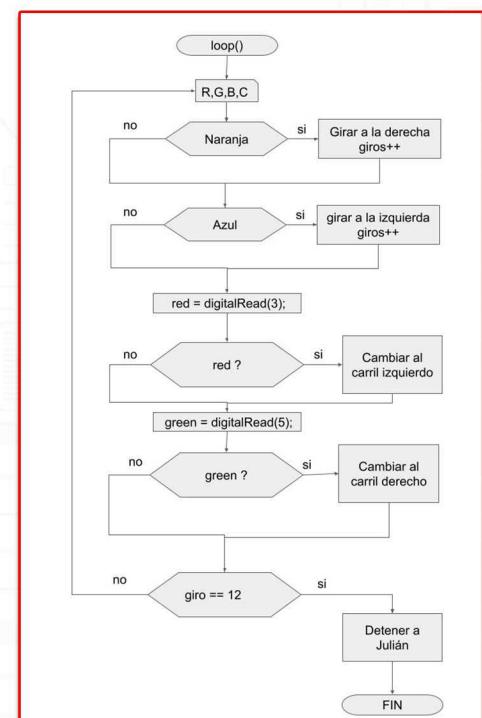
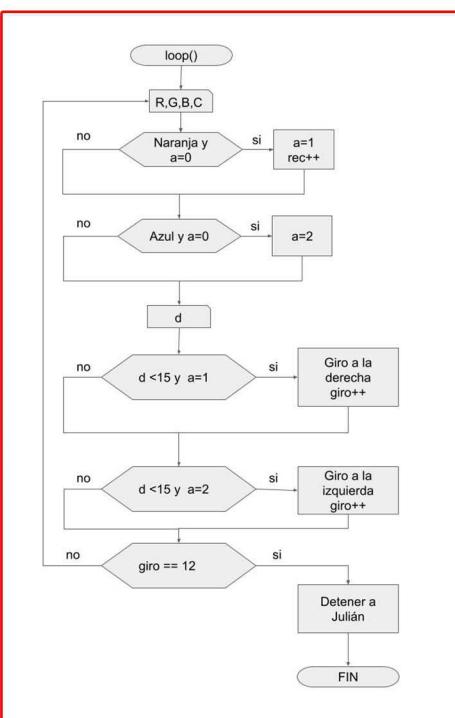
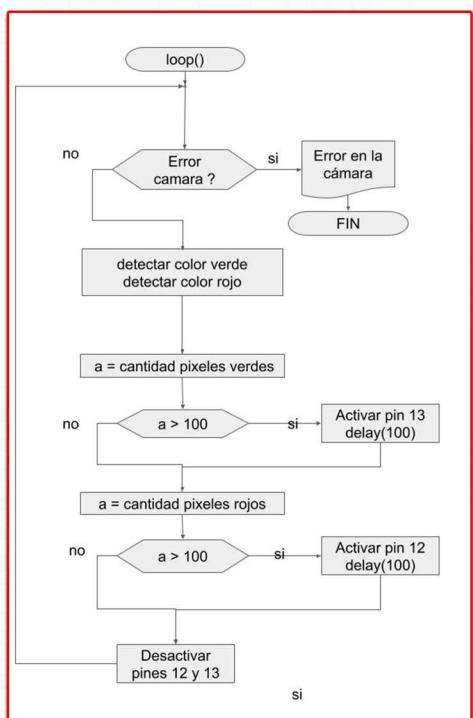
Se entiende que el algoritmo presentado en el diagrama de flujo, se repite indefinidamente como parte de la estructura de programación del Arduino, por lo cual no se representa expresamente en el diagrama.

2. PROCESO DEL DISEÑO

2.3.1. DIAGRAMA DE FLUJO

En el siguiente gráfico se muestra el diagrama de flujo del primer y segundo reto, en la cual el robot, en el reto numero uno debe completar **tres (3)** vueltas en la pista con ubicaciones aleatorias de las paredes interiores de la pista.

y el reto número dos el robot debe completar **tres (3)** vueltas en la pista con señales de tráfico verdes y rojas colocadas al azar dentro de las paredes interiores de la pista.



3. ESTRATEGIAS

La estrategia es una de las partes fundamentales de todo el reto, por lo tanto, en el proceso de creación fue tarea de todo el equipo encontrar y definir las mejores métodos en función a los componentes.

Como primera estrategia, el equipo trabajo decidió que lo mejor sería enfocarse en el reto número 1. En este se consideró que:

El mayor reto, sería definir la dirección. Para ello, el equipo utilizó las líneas naranjas y azules que se encuentran en la cancha para determinar si se debía hacer un giro en sentido horario o anti horario.

Bajo esto, como ya el equipo de trabajo conocía que luego del primer giro, los demás serían en el mismo sentido, se decide que después de la detección del sensor de color, el robot gire con el sensor de ultrasonido.

Como segunda estrategia para el reto número 2, se utiliza la cámara **ESP32-cam** y el sensor de ultrasonido, para definir:

El color de los semáforos y como debería esquivar estos, para evitar chocar y desviarse del camino.

Asimismo, con la información obtenida del ultrasonido, se puede calcular la posición en la cual se encuentra el robot, y repositionarse en caso de ser necesario.

4. DISCUSIÓN DE CÓDIGOS

El lenguaje de programación utilizado para la elaboración del código es **C++** desarrollado para el funcionamiento autónomo del robot y posterior discusión.

Los códigos desarrollados se pueden observar en el apéndice E. El código esencialmente está dividido en seis partes, las cuales serían las siguientes:

Las librerías

Las librerías utilizadas fueron “**Servo.h**” encargado de facilitar la tarea de movilizar el servo y en general la dirección y “**Adafruit_TCS34725.h**” encargado de facilitar la detección de color por el sensor “**RGB TCS - 34725**”.

Declaración de pines

Se define la función de cada uno de los pines del **Arduino Mega 2560**, de manera que la mayoría se encuentran “**OUTPUT**” (salida) y algunos en “**INPUT**” (entrada) dependiendo de las necesidades.

Inicialización del sensor de color

Se enciende el sensor de color y luego se revisa mediante el monitor serial, confirmando si existe cualquier tipo de inconveniente al iniciarse.

Detección de color

Una vez realizado el proceso de “**Setup**” se empieza a guardar los valores de RGB definidos por el sensor de color, en variables denominadas “**R”, “G” y “B**” que mediante un comando de forma “**IF**” se pueden utilizar para hacer ciertos movimientos de dirección.

Detección de distancia

Al momento de definir distancias, para hacerlo de forma más eficiente lo establecimos en la función ping (); para que de esta manera sea mucho más fácil realizar acciones en función de eso.

Movimientos a realizar

Siempre que se detecte el color naranja o azul en el sensor de color, se moverá el robot en sentido horario o anti horario respectivamente, en un ángulo de 90 grados, donde este movimiento resulta eficiente para la realización de las tres vueltas, las cuales son todas completamente importantes al momento de la programación del robot. Al ser programado en el **Arduino IDE**, todo se encuentra en lenguaje de **C++**.

5. ENTRADA DEL DIARIO

5. 1. CRONOLOGÍA DESAFÍOS Y SOLUCIONES

1era semana de julio



El primer día de preparación para la competencia regional el equipo comenzó a estudiar y analizar las reglas de la competencia.

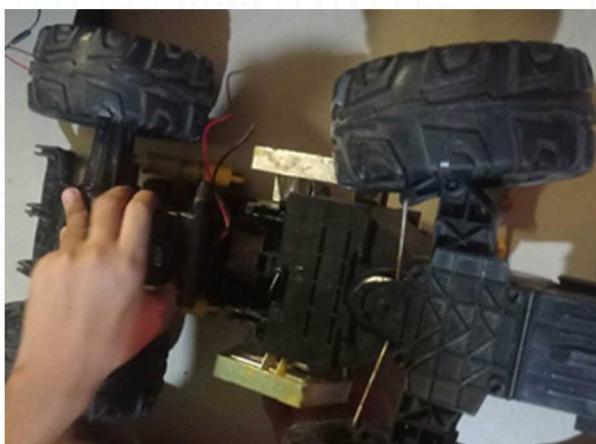


Los siguientes días se comenzó a estudiar lo que podría ser el primer modelo del chasis y se investigaron diversas maneras de solucionar lo que serían las primeras problemáticas que se presentaron, las cuales fueron cómo se diseñaría el sistema de dirección y qué motor se utilizaría para conseguir la velocidad y el torque necesario.

5. ENTRADA DEL DIARIO

A continuación el equipo empezó a **buscar motores** que pudieran utilizarse, desarmando juguetes, impresoras, entre otros dispositivos.

Finalmente se consiguió el motor necesario al desarmar un carro de control remoto modelo: **Dodge T-Rex Ram** de la marca **Nikko**, el cual proporcionó las piezas mecánicas necesarias para poder diseñar el sistema de dirección.



2 da Semana de julio

Una vez resuelto el problema del sistema de tracción, se abordó la problemática del **sistema de dirección**, y luego de investigar diferentes soluciones, se consiguió la manera de adaptar un servomotor a las piezas mecánicas obtenidas anteriormente.

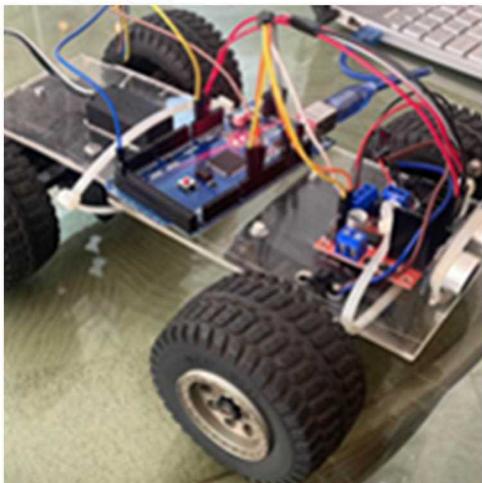


5. ENTRADA DEL DIARIO

Posteriormente el equipo de trabajo procedió al ensamblaje de ambos sistemas, y de los diferentes dispositivos que necesitaría el robot en bases de acrílico.

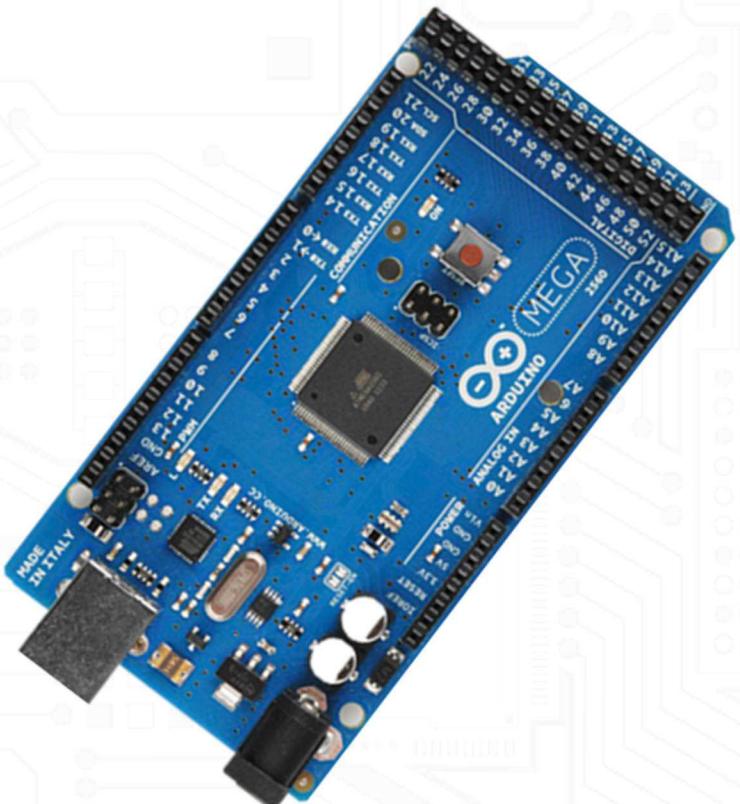


Finalizamos el **primer prototipo** de lo que sería el chasis, y poder proseguir con el área de programación.



3 ra Semana de julio

Para la **programación** se utilizó un **Arduino mega 2560** como programador, un puente **H doble** como regulador de potencia y velocidad y un sensor de ultrasonido para medir distancia.



5. ENTRADA DEL DIARIO

Consecutivamente el equipo de trabajo reanudó la búsqueda de diferentes soluciones para **detectar los colores** de los semáforos, decidiendo utilizar una cámara **ESP32-cam** con un lente **OV2640**, siendo la siguiente problemática cómo programar ésta con Arduino.



Se investigó qué fuente de poder se iba a utilizar para energizar el robot, debido a que luego de utilizar baterías de **9V**, el equipo se dio cuenta de que éstas no eran ideales para el robot debido a que se desgastan en muy poco tiempo.

Por consiguiente se terminó uniendo dos packs de baterías, los cuales tenían ocho baterías recargables de **1.2 V** en serie, las cuales finalmente sumaban un total de **9.6 V**.

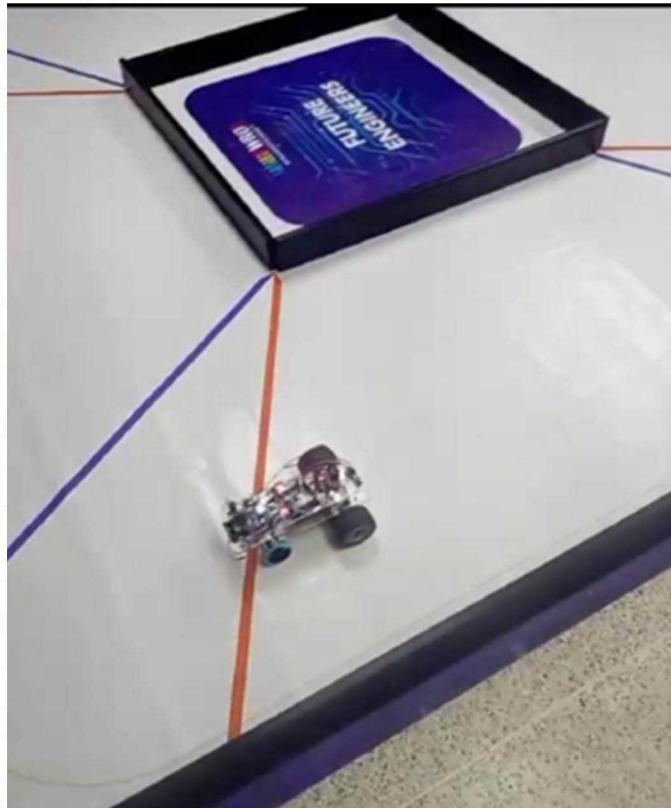
Debido al espacio necesario, se diseño un **segundo prototipo** en el cual se decidió agregar un segundo piso en el robot, ubicando el área de electrónica en éste, y las baterías, el sistema de tracción y el sistema de dirección en el primer piso.



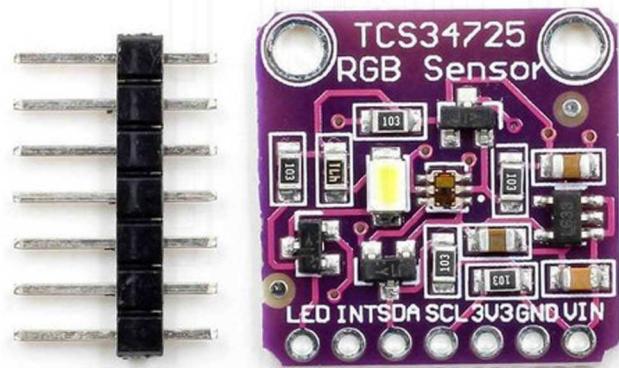
5. ENTRADA DEL DIARIO

4ta Semana de julio

Sin aún poder solucionar la programación, se decidió utilizar un sensor **RGB 34725** para que el robot detectara en cuál dirección debe cruzar.



Antes de la competencia regional el equipo de trabajo encontró baterías recargables con un voltaje mayor (**3.7 V**), por lo cual se decidió eliminar uno de los dos pack de baterías, y modificar el restante para que trabaje con tres baterías.



Se plantea estrategia, se decide cruzar con la detección de color de las líneas de la pista, utilizando asimismo dos ultrasonidos más, uno en cada lado del robot, para que así, una vez detecte una pared, este cruce para evitar un choque.

Sin embargo, estos dos ultrasonidos no significaban una ayuda, sino más bien un problema, ya que, cuando éstos detectaban, el robot perdía la trayectoria, por lo cual finalmente se decidió no utilizar estos dos sensores de ultrasonido.



5. ENTRADA DEL DIARIO

1ra Semana de agosto



Después de participar en la primera competencia regional, comienza la búsqueda de soluciones a los problemas presentados. Se decide cambiar el **sistema de dirección**, creando uno nuevo con piezas obtenidas en un kit de robótica **Spike Prime** número **45678**, ya que con este nuevo sistema de dirección se conseguiría tener un mayor radio de giro, así como se podría tener un giro más preciso.

2da Semana de agosto



Se plantea nuevas estrategias, se decidió que la manera idónea de cruzar sería con ayuda del ultrasonido y que el sensor **TCS34725** sólo detectase la primera línea para determinar si el robot debe cruzar en sentido horario o anti horario.



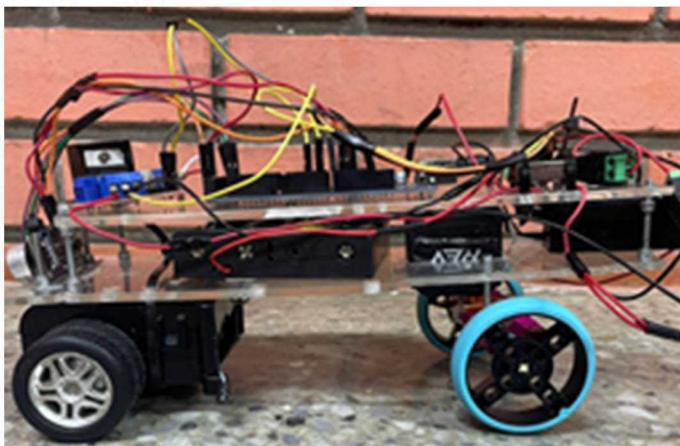
3ra Semana de agosto

Siguiendo con la segunda parte del reto, el equipo comenzó a programar la cámara buscando cómo transferir la información de la cámara al Arduino sin necesidad de utilizar **WIFI**. Luego de buscar en varias fuentes se encontró la solución trasmitiendo la información a través de los puertos seriales.

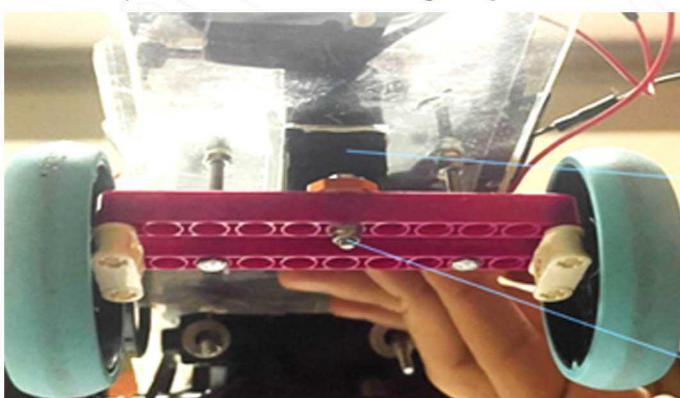
5. ENTRADA DEL DIARIO

4ta Semana de agosto

Luego el motor utilizado por todo este tiempo comenzó a fallar muy seguido, impidiendo el avance del equipo de trabajo en el segundo reto, por lo que faltando pocos días para la competencia, el equipo sacó el motor y por lo tanto la caja de cambio de otro carro a control remoto para acoplarlo al robot.

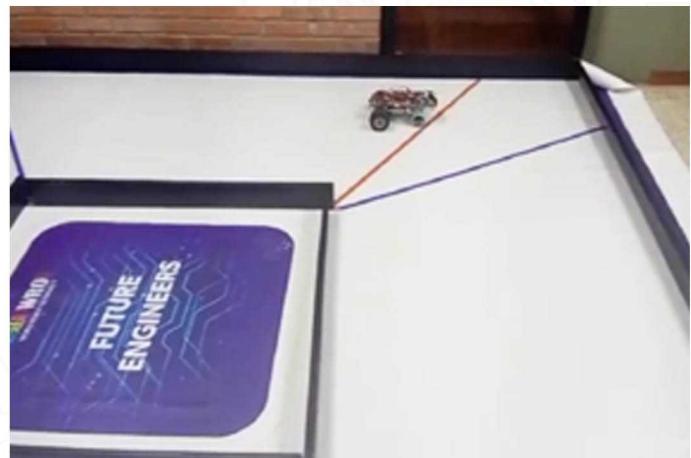


Para proporcionar un mayor radio de giro se modificó el sistema de dirección, usando piezas de un **Kit Lego Spike Prime**.



1ra Semana de septiembre

Se realizaron prácticas en la pista, para el rendimiento del robot en el reto número 1 y 2



2da Semana de septiembre

Elaboración y actualización del informe.



5. ENTRADA DEL DIARIO

3ra Semana de septiembre

Competencia Nacional



Competencia Nacional

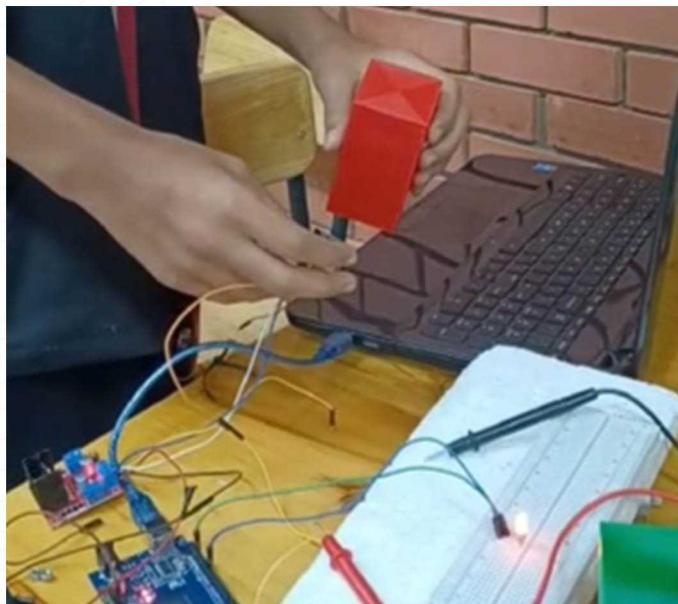


Octubre del 2023

5. ENTRADA DEL DIARIO

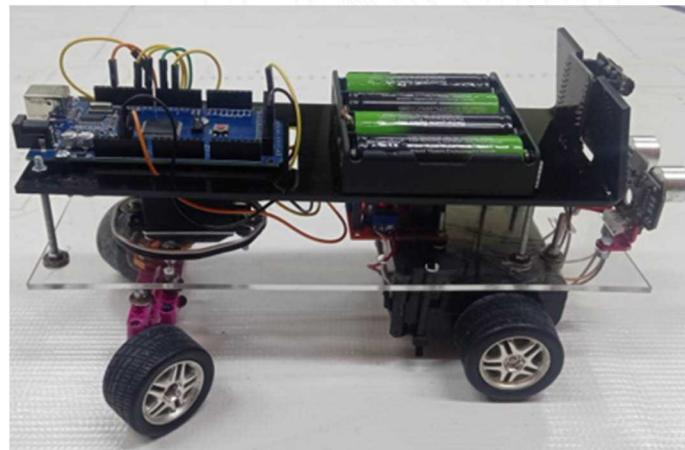
4ta Semana de septiembre

Se continuo la programación de la cámara para el reto número dos.



Cambio de ruedas del sistema de dirección para aumentar la seguridad y la estética.

1ra Semana de octubre



Construcción del tercer prototipo del robot, que se basa en la restauración de los acrílico, organización de los cables por medio de conectores para mejorar la parte estética.

5. ENTRADA DEL DIARIO

2da Semana de octubre

Programación de las cámaras y prácticas con ellas.

Se comenzó a buscar opciones de cámaras con mejor imagen en nitidez y amplitud de visión. Se hace prueba con una tarjeta **Raspberry Pi** y cámara web gran angular.

Y se comienza a decidir qué sistema usar para cámara.



Tarjeta Raspberry Pi



Cámara gran angular



Equipo trabajando

6. OPORTUNIDADES DE MEJORAS

Las funciones del robot presentado son suficientes para superar todos los retos planteados por la categoría, sin embargo se identifican las siguientes oportunidades de mejora, en las que se estará trabajando para próximas competiciones.

Agregar dos sensores ultrasónico para la detección de distancia:

Se tratará de evitar choques laterales por lo tanto se debería colocar un sensor en cada lado del robot.



Agregar conectores:

Al agregar mayor cantidad de conectores permite facilitar el cambio de piezas y mejorar la organización de los cables dupont.



Construir una carrocería:

Es un elemento importante el cual mostrara una mejor estética del robot.

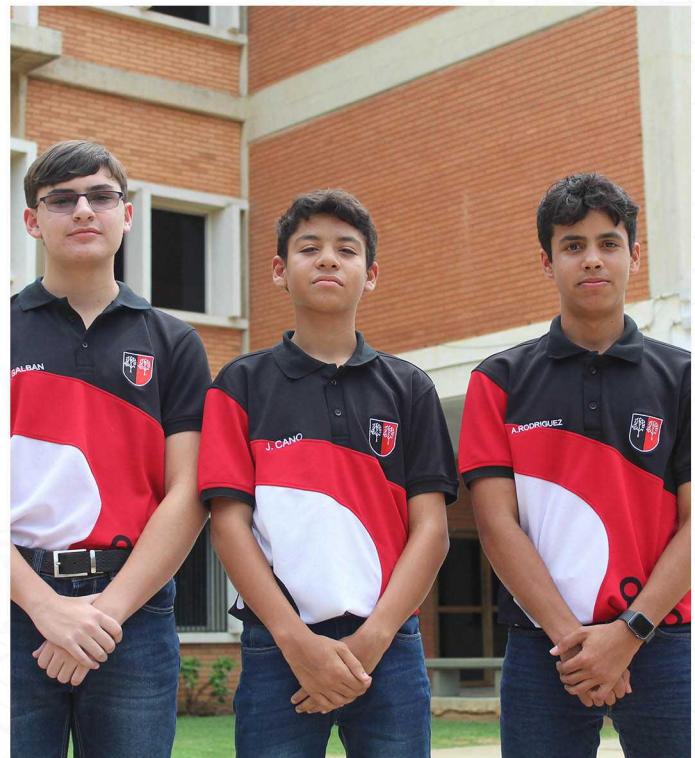


CONCLUSIÓN

El equipo de trabajo logró desarrollar un robot autónomo **"Julian"** capaz de realizar los dos retos exigidos para la categoría de Futuros Ingenieros. Para lograr esto, se utilizó un robot equipado con diferentes componentes electrónicos y mecánicos, a su vez cumpliendo con las especificaciones establecidas en las premisas de la competencia.

Durante el proceso, se enfrentaron varios desafíos, como la detección y evitación de obstáculos, el cumplimiento de las tres vueltas y la detección de las líneas sobre la pista.

Los resultados obtenidos demuestran la eficacia de esta aproximación en entornos controlados y la capacidad del robot para adaptarse a situaciones cambiantes.

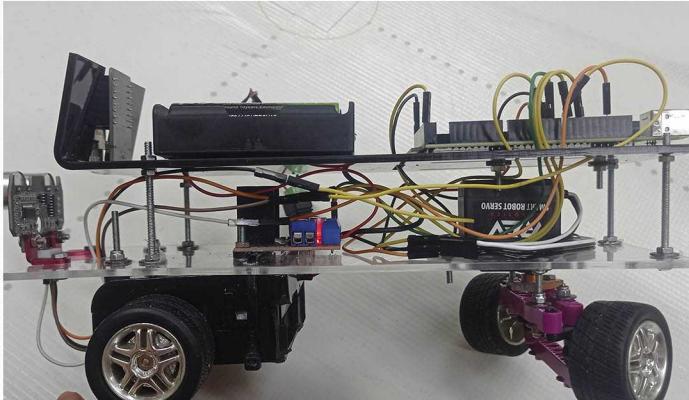


Durante estos meses de arduo trabajo, el equipo ha logrado cumplir sus objetivos en cuanto a la competencia, así como cada uno de los participantes profundizaron conocimientos en las diversas áreas como son la programación, la electrónica, la mecánica, entre otras.

APÉNDICES

Apéndice A:

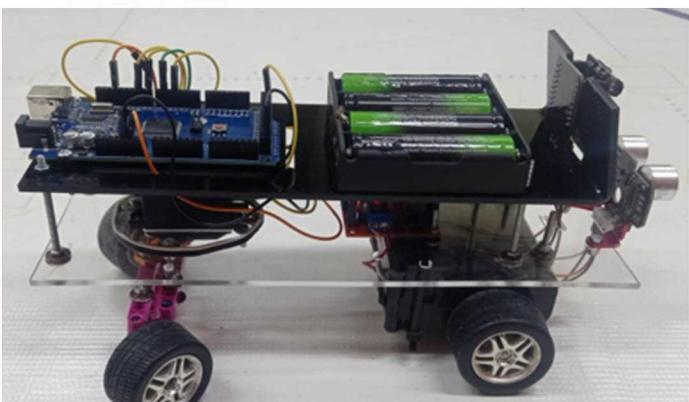
Vistas del Carro Robot (fotos)



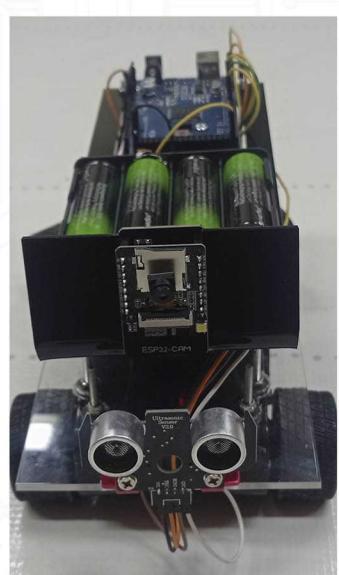
Vista Lateral Izquierda



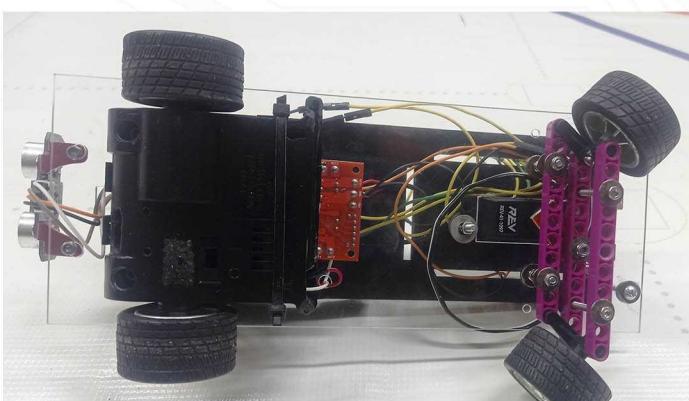
Vista Trasera



Vista Lateral Derecha



Vista Frontal



Vista Inferior



Vista Superior

APÉNDICES

Apéndice B:



Desafío 1

<https://youtu.be/8zjeBPan2Oo?si=nGEqHtsf5115c2X7>



Prueba de desafío 2

<https://youtube.com/shorts/BUVsap2CQz4?si=4hXoWpP2MHMu4uv>

APÉNDICES

Apéndice C: Especificaciones de los sensores



Módulo de rango ultrasónico Microbit 2.0

Fuente:

<http://www.datasheet.es/PDF/779948/HC-SR04-pdf.html>

Características del producto:

El módulo de rango ultrasónico **Microbit 2.0** proporciona una función de medición sin contacto de **2 cm** a **400 cm**, la precisión de rango puede alcanzar los **3 mm**. Los módulos incluyen transmisores ultrasónicos, receptores y circuito de control.



El principio básico de trabajo:

(1) Usando el disparador **IO** para al menos **10us** de señal de alto nivel.

(2) El módulo envía automáticamente ocho **40 kHz** y detecta si hay una señal de pulso de regreso.

(3) Si la señal regresa, a través de alto nivel, el tiempo de duración de **IO** de salida alta es el tiempo desde el envío ultrasónico hasta el regreso. Distancia de prueba = (tiempo de alto nivel × velocidad del sonido **(340M/S)**) / 2.

Parámetros eléctricos

• Voltaje de trabajo	DC 5V
• Corriente de trabajo	15 mA
• Frecuencia de trabajo	40 Hz
• Rango máximo	4 m
• Rango mínimo	2 cm
• Medición de ángulo	15 degree
• Señal de entrada del disparador	10 US TTL PULSE
• Señal de salida de eco	Señal de palanca TTL y el rango en proporción
• Dimensión	45mm x 20 mm x 15 mm



APÉNDICES

Apéndice C: Especificaciones de los sensores



Camara ESP 32 - cam

Fuente:

https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf

Características

- Velocidad de reloj de hasta **160 MHz**, potencia informática resumida de hasta **600 DMIPS**
- **SRAM** de **520 KB** incorporada, **4MPS RAM** externa
- Admite **UART/SPI/I2C/PWM/ADC/DAC**
- Admite cámaras **OV2640** y **OV7670**, lámpara de flash incorporada.
- Admite carga de imagen **WIFI** - Admite tarjeta **TF**
- Admite múltiples modos de suspensión.
- **Lwip** integrado y **FreeRTOS**
- Admite el modo de operación **STA/AP/STA+AP**
- Admite la tecnología **Smart Config / AirKiss**
- Compatibilidad con actualizaciones de firmware remotas y locales de puerto serie (**FOTA**)



APÉNDICES

Apéndice C: Especificaciones

- **SPI Flash:** Predeterminado 32Mbit
- **RAM:** Incorporada 520 KB+4MPSRAM externa
- **Dimensión:** 27*40,5*4,5 ($\pm 0,2$) mm/1,06*1,59*0,18"
- **Bluetooth:** Bluetooth 4.2 BR/EDR y estándares BLE
- **WiFi:** 802.11b/g/n/e/i
- **Interfaz de soporte:** UART, SPI, I2C, PWM
- **Soporte de tarjeta TF:** Soporte máximo 4G
- **Puerto E/S:** 9
- **Tasa de baudios del puerto serie:** Predeterminado 115200 bps
- **Formato de salida de imagen:** JPEG (sólo compatible con OV2640), BMP, escala de grises
- **Rango de espectro:** 2412 ~ 2484 MHz
- **Antena:** antena PCB integrada, ganancia 2dBi
- **Potencia de transmisión:**
 - 802.11b: 17±2 dBm (@11Mbps);
 - 802.11g: 14±2 dBm (@54Mbps);
 - 802.11n: 13±2 dBm (@MCS7)
- **Sensibilidad de recepción:**
 - CCK, 1 Mbps: -90dBm;
 - CCK, 11 Mbps: -85dBm;
 - 6 Mbps (1/2 BPSK): -88dBm;
 - 54 Mbps (3/4 64-QAM): -70dBm;}
 - MCS7 (65 Mbps, 72.2 Mbps): -67dBm
- **Consumo de energía:**
 - Apagar el flash:** 180mA@5V
 - Enciende el flash:** ajuste del brillo al máximo: 310mA@5V
 - Sueño profundo:** el consumo de energía más bajo puede alcanzar 6mA @ 5V
 - Modern-sleep:** hasta 20mA@5V
 - Sueño ligero:** hasta 6,7 mA a 5V
- **Seguridad:** WPA/WPA2/WPA2-Enterprise/WPS
- **Rango de fuente de alimentación:** 5V
- **Temperatura de funcionamiento:** -20 °C ~ 85 °C
- **Entorno de almacenamiento:** -40 °C ~ 90 °C, <90% HR
- **Peso:** 10g

APÉNDICES

Apéndice C: Especificaciones de los sensores



TCS34725 RGB SENSOR

Fuente: <https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf>

Características

- **Luz roja, verde, azul (RGB) y clara**

Detección con filtro de bloqueo de infrarrojos

- Ganancia analógica programable y Tiempo de integración

- **3,800,000:1** Rango Dinámico

- Sensibilidad muy alta: ideal para Operación detrás de un cristal oscuro

- **Interrupción enmascarable**

- Superior e inferior programables Umbrales con filtro de persistencia

- **Administración de energía**

- Baja potencia: estado de suspensión de **2,5 A**

- **65-A** Estado de espera con espera programable

Tiempo de estado de **2,4 ms a > 7 segundos**

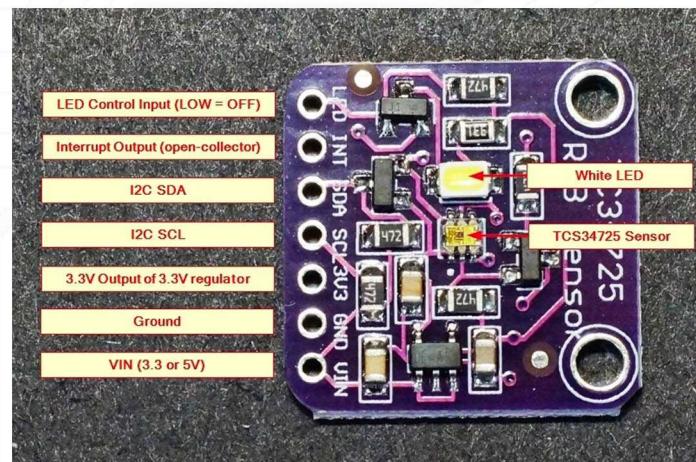
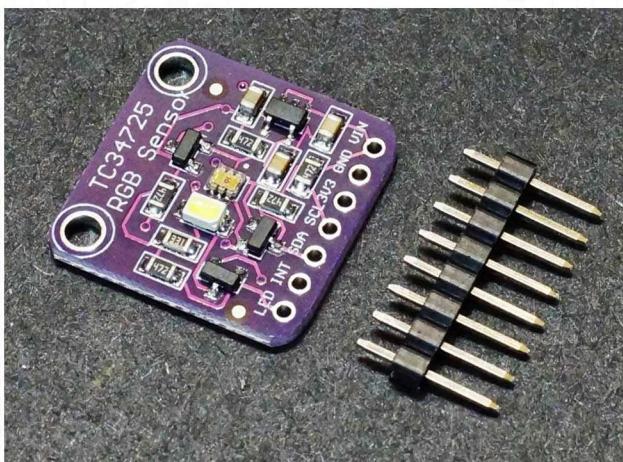
- **Interfaz compatible con modo rápido 2C**

- Velocidades de datos de hasta **400 kbit/s**

- Niveles de voltaje de entrada compatibles con **VDD o bus de 1,8 V**

- **Set de Registro y Pin Compatible** con el Serie **TCS3x71**

- **Pequeño** 2 mm 2,4 mm **Plano doble** Sin plomo (**FN**) Paquete



APÉNDICES

Apéndice D: Especificaciones de los actuadores



Motor Eléctrico Rs 390

Fuente: <https://es.aliexpress.com/item/32891103572.html?gatewayAdapt=Msite2Pc>



Características

- **Modelo:** Rs390
- **Voltaje nominal:** 6V
- **Potencia:** 25-35W
- **Velocidad sin carga:** 14000/min
- **Tamaño:** aprox. 70*28mm
- **Eje:** 2mm



Smart Robot Servo

Fuente: <https://www.revrobotics.com/rev-41-1097/>

Características

- **Tamaño:** 40,2 mm x 20,0 mm x 38,0 mm
- **Peso:** 2,05 onzas.
- **Velocidad:** 0,14 s/60° (a 6 V): Par de parada: 13,5 kg-cm / 187,8 oz-in (a 6 V)
- **Corriente de parada:** 2A (a 6V)
- **Voltaje nominal:** 4,8 V - 7,4 V, 6 V nominal
- **Rango de pulso de entrada:** 500 µs - 2500 µs
- **Rango angular predeterminado:** 270°
- **Rango máximo programable en modo angular:** - 280°
- **Material del engranaje:** Metal
- **Tipo estriado:** - 25T
- **Spline Rosca Interna Tamaño:** M3
- **Spline profundidad de rosca interna:** 6 mm



APÉNDICES

Apéndice E: Códigos del Arduino

Código del reto 1

```

// Declaración de Librerías
#include <Servo.h>
#include "Adafruit_TCS34725.h"
Adafruit_TCS34725 tcs = 
  Adafruit_TCS34725(TCS34725_CS, TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_1X);

Servo pro;
int d = 16; // Distancia ultrasonido izquierdo
int d_rec = 105; // Valor del servo para avinarzar decho
int da=16; // Distancia ultrasonido derecho
int tcs_d = 105; // Distancia ultrasonido tiempo de rebote izquierdo
int tcs_da = 105; // Distancia ultrasonido tiempo de rebote derecho
int vel = 80;
void setup()
{
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT); // Trigger
  pinMode(11, INPUT); // Echo
  pinMode(12, OUTPUT); // Trigger
  pinMode(13, INPUT); // Echo
  pinMode(14, OUTPUT); // Echo
  pinMode(15, INPUT); // Echo
  pinMode(22, OUTPUT);
  pro.attach(4); // Pin de control del servo
  digitalWrite(22,HIGH);
  delay(2000);
  tcs.begin();
  Serial.println("Iniciando");
  delay(2000);
}
else{
  Serial.println("Error");
  Serial.println("Debe revisar las conexiones e iniciar nuevamente");
  while(1)delay(500);
}
pro.write(rec);
delay(800);
digitalWrite(8,HIGH);
analogWrite(9,vel);
pro.write(105);
}

void loop() {
  //CALCULAR COLOR
  uint16_t r,g,b,c;
  tcs.getRawData(&r,&g,&b,&c);
  //DIRECCIONAR Y MOVER MOTORES

  //MOVIMIENTO AL VER UNA naranja
  if((R>100&&(B<150))|| 
  (R>100&&(B>150)))
  digitalWrite(8,LOW);
  pro.write(vel);

  analogWrite(9,vel);
  digitalWrite(10,LOW);
  digitalWrite(8,HIGH);
  delay(1500);

  analogWrite(9,vel);
  pro.write(rec);

  //AZULLL
  if((R<90)&&(G>110)&&(B<130))
  pro.write(rec);
  analogWrite(9,0);
  delay(100);
  digitalWrite(8,HIGH);
  analogWrite(9,110);
  pro.write(114);
  delay(2000);
  pro.write(rec);
  analogWrite(9,70);

  ping();
  if(d<17)
  pro.write(55);
  analogWrite(9,vel);
  delay(100);
  pro.write(114);
  analogWrite(9,vel);
  delay(100);
  pro.write(rec);
  d=100;
}

// Control del sensor de ultrasonido izquierdo y cálculo de distancia
int ping10() {
  digitalWrite(12, HIGH);
  delayMicroseconds(10); //Enviamos un pulso de 10us
  digitalWrite(12, LOW);

  t = pulseIn(13, HIGH); //obtenemos el ancho del pulso
  d = t*59;
}

// Control del sensor de ultrasonido derecho y cálculo de distancia
int ping10_1() {
  digitalWrite(12, HIGH);
  delayMicroseconds(10); //Enviamos un pulso de 10us
  digitalWrite(12, LOW);

  t = pulseIn(13, HIGH); //obtenemos el ancho del pulso
  d = t*59;
}

```

Código del reto 2

```

// Declaración de Librerías
#include <Servo.h>
#include "Adafruit_TCS34725.h"
Adafruit_TCS34725 tcs = 
  Adafruit_TCS34725(TCS34725_CS, TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_1X);

Servo pro;
int d = 16; // Distancia ultrasonido izquierdo
int a = 0; // Distancia ultrasonido derecho
int giro = 75; // Valor del servo para avinarzar decho
int da=16; // Distancia ultrasonido tiempo de rebote izquierdo
int tcs_d = 105; // Distancia ultrasonido tiempo de rebote derecho
int tcs_da = 105; // Distancia ultrasonido tiempo de rebote derecho
int vel = 35;
int givel;
int da=16;
int green = 0;
void setup()
{
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, INPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, INPUT); // Trigger
  pinMode(11, INPUT); // Echo
  pinMode(12, OUTPUT); // Trigger
  pinMode(13, INPUT); // Echo
  pinMode(14, OUTPUT); // Echo
  pinMode(15, INPUT); // Echo
  pinMode(22, OUTPUT);
  pro.attach(4); // Pin de control del servo
  digitalWrite(22,HIGH);
  delay(3000);
  tcs.begin();
  Serial.begin(9600);
  if(capture)
  Serial.println("Iniciando");
  delay(500);
}

pro.write(rec);
delay(4000);
digitalWrite(8,HIGH);
analogWrite(6,vel+6);

}

void loop() {

  //CALCULAR COLOR
  uint16_t r,g,b,c;
  tcs.getRawData(&r,&g,&b,&c);
  //DIRECCIONAR Y MOVER MOTORES

  //giro antihorario (Naranja)
  if((R>90&&(G>100))|| 
  (R>100&&(G>100)))
  pro.write(100);
  analogWrite(6,g);
  delay(150);
  pro.write(100);
  delay(700);
  analogWrite(6,vel);
  a+=1;

  }

  //detectar giro horario(Azul)
  if((R<90)&&(G<400))
  pro.write(40);
  analogWrite(6,g);
  delay(150);
  pro.write(40);
  delay(700);
  analogWrite(6,vel);
  a+=1;

  }

green = digitalRead(5);

if(green == HIGH){
  pro.write(50);
  delay(50);
  pro.write(50);
  delay(50);
  pro.write(100);
  delay(50);
  pro.write(100);
  delay(50);
  pro.write(rec);
  red = digitalRead(3);

  if(red == HIGH){
  pro.write(50);
  delay(50);
  pro.write(50);
  delay(50);
  pro.write(100);
  delay(50);
  pro.write(100);
  delay(50);
  pro.write(rec);
  }

  //detectar 3 vueltas y detenerse
  if (giro == 12){
  analogWrite(8,0);
  a=0;
  }

  //Control del sensor de ultrasonido izquierdo y cálculo de distancia
  int ping1() {
  digitalWrite(10, HIGH);
  delayMicroseconds(10);
  digitalWrite(10, LOW);

  t = pulseIn(11, HIGH); //obtenemos el ancho del pulso
  d = t*59;
  }

  //Control del sensor de ultrasonido derecho y cálculo de distancia
  int ping1_1() {
  digitalWrite(12, HIGH);
  delayMicroseconds(10);
  digitalWrite(12, LOW);

  t = pulseIn(11, HIGH); //obtenemos el ancho del pulso
  d = t*59;
  }
}

```

Código de la cámara

```

// 20_Color_Blob_Detector.ino
/*
 * Detect blobs of given color
 */
#define MAX_RESOLUTION_VGA
#define MAX_RECUSION_DEPTH 13

#include "esp32cam.h"
#include "esp32cam/JpegDecoder.h"
#include "esp32cam/applications/ColorBlobDetector.h"

using namespace Eloquent::Esp32cam;

int a;
int b;

Cam cam;
JpegDecoder decoder;
Applications::ColorBlobDetector detector(7,134,64);
Applications::ColorBlobDetector detector2(169,68,58);

void setup() {
  pinMode(23, OUTPUT);
  Serial.begin(115200);
  delay(3000);
  Serial.println("Init");
  pinMode(13, OUTPUT);
  pinMode(14, OUTPUT);
  cam.alternate();
  cam.highQuality();
  cam.vga();
  cam.highestSaturation();
  cam.disableAutomaticWhiteBalance();
  cam.disableAutomaticExposureControl();
  cam.disableGainControl();

  /*
   * Set detector tolerance
   * The higher, the more shade of colors it will pick
   */
  detector.tolerate(47);
  detector2.tolerate(35);
  /*
   * Skip blob localization (slow) if not enough
   * pixels match color
   */
  while (!cam.begin())
  Serial.println(cam.getMessage());
}

void loop() {
  digitalWrite(33,LOW);
  if (cam.capture()) {
    Serial.println(cam.getMessage());
    return;
  }

  if (decoder.decode(cam)) {
    Serial.println(decoder.getMessage());
    return;
  }

  /*
   * Detect blob in frame
   */
  if (detector.detect(decoder)) {
  }
  else {
  }
  if (detector2.detect(decoder)) {
  }
  else {
  }

  // while debugging, these may turn out useful
  Serial.print("n rojo ");
  Serial.print(detector.maskCount);
  a = detector.maskCount;

  if(a<100){
  digitalWrite(13,HIGH);
  Serial.print("a");
  delay(100);
  }

  Serial.print("n verde ");
  Serial.print(detector2.maskCount);
  b = detector2.maskCount;

  if(b>100){
  digitalWrite(12,HIGH);
  Serial.print("b");
  delay(100);
  }

  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(33,LOW);
}

```

REFERENCIAS BIBLIOGRÁFICAS



- **TRACCIÓN DE AUTO NIKKO:**
<https://www.youtube.com/watch?v=9tLaDAoxxio>



- **PDF HC-SR04 Data sheet:**
<http://www.datasheet.es/PDF/779948/HC-SR04-pdf.html>



- **ESP32-CAM Development Board:**
https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf



- **ESP32-CAM Video Streaming and Face Recognition with Arduino IDE:**
<https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-i/>



- **TCS3472 COLOR LIGHT-TO-DIGITAL CONVERTER with IR FILTER:**
<https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf>



- **Interfacing TCS34725 Color Sensor with Arduino:**
<https://electropeak.com/learn/interfacing-tcs34725-color-sensor-with-arduino/>



- **TCS34725 Color Sensor:**
https://www.waveshare.com/wiki/TCS34725_Color_Sensor#:~:text=TCS34725%20is%20used%20for%20color



- **Sensor ultrasonido OKY6010:**
<https://www.okystar.com/product-item/smарт-кар-робот-беспроводной-ультразвуковой-сенсор-дистан>