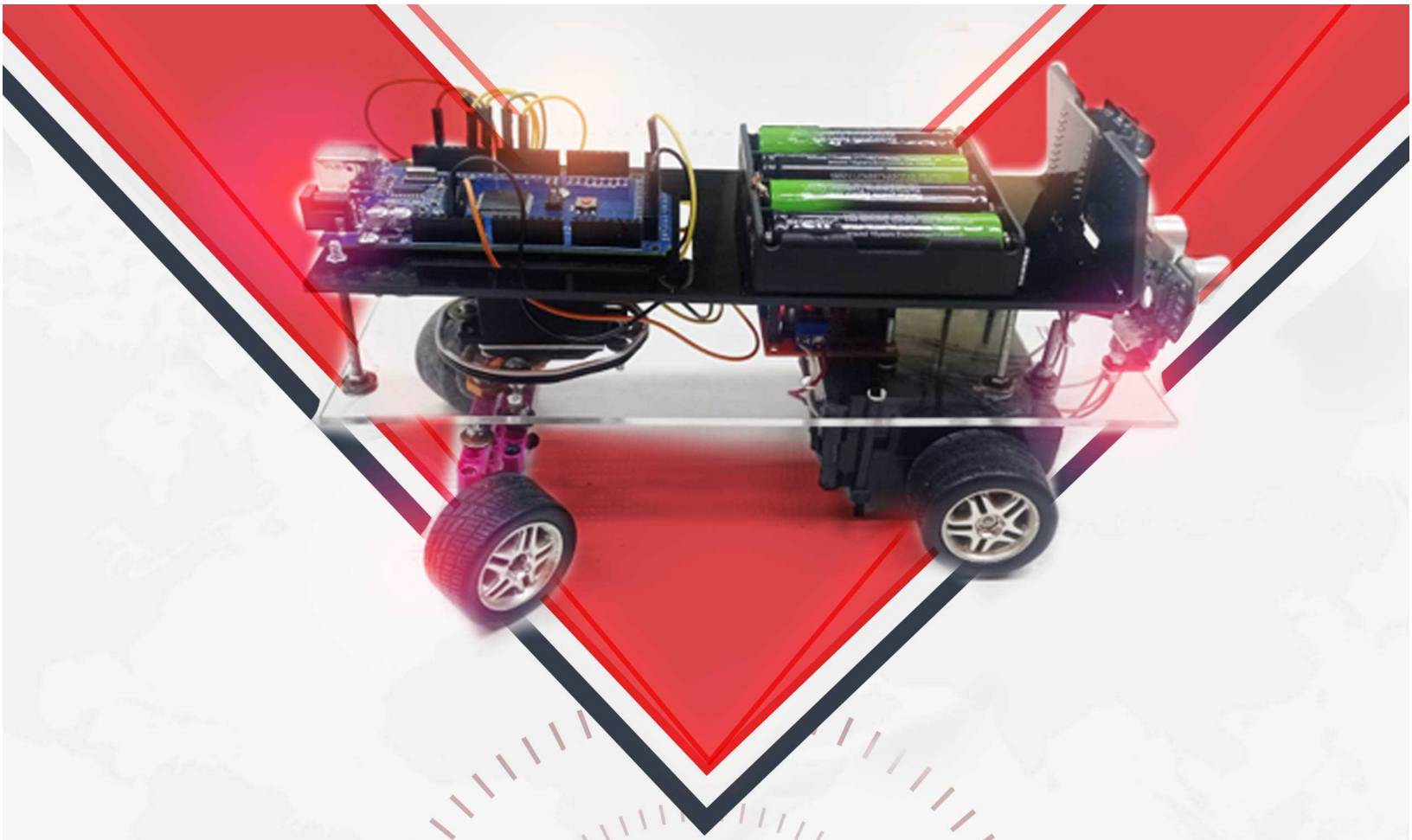


WRO®

WORLD ROBOT OLYMPIAD™

FUTURE ENGINEERS

RED MACHINE

PROJECT "Julián"

Liceo Los Robles

Team members

Cano Barros, Juan Diego

Galban Franco, Samuel José

Rodríguez Guerra, Ángel Saúl

CONNECTING THE WORLD

Venezuelan Delegation

October 2023

CONTENT

Introduction

1.Team formation

2. Design process

2.1 Mechanics
2.1.1 Steering
2.1.2 Driving
2.1.3 Chassis design

2.2 Electronic
2.2.1 Sensors
2.2.2 Speed/direction control

2.3 Programming
2.3.1 Flowchart

3. Strategies

4. Code discussion

5. Diary entries

5.1. Chronology - challenges and solutions.

6. Opportunities for improvements.

Conclusion

Appendices
Appendix A - Views of the robot car
Appendix B - Performance Videos
Appendix C - Sensor Specification
Appendix D - Actuator Specification
Appendix E - Arduino code.

INTRODUCTION

Robotics is a science that brings together several disciplines with the aim of designing machines programmed to perform tasks automatically. It involves not only the development of robots, but also their design, programming, production and application.

It also allows you to develop cognitive and social skills and abilities such as problem solving, teamwork, independent learning and communication.

World Robotic Olympiad (WRO) is a competition first held in **2004** and its mission is to help young people develop their creativity and problem-solving skills in a fun and engaging way. To this end, robotics competitions were organized in four different categories for students between **8** and **19** years old.

The category referred in this document is future engineers, which has two challenges to accomplish. Challenge number one is that the robot must complete three laps on the track with random locations within the track walls. Challenge number two consists of the robot must complete three laps on the track with green and red signs placed randomly within the walls of the track. To complete the challenge, a robot is developed to be able of facing the conditions set, at the design level and at the operating level defined in the competition rules.

Following the structure of the category evaluation criteria, this document describes in detail the solution designed for the **"Julián"** robot, which represents the **"Red Machine"** team, defining the criteria applied in the mechanical part, specifically steering, driving, and design of the chassis, as well as, explaining the entire electronic system that composes it, such as the sensors, speed and direction controller, and finally its programming which is explained through the system of code modules used to achieve the fulfillment of challenges number one and two.

1. TEAM FORMATION

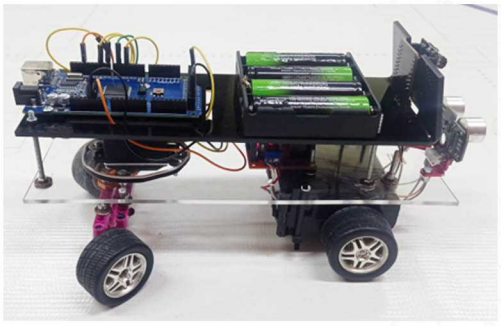
Red Machine team initially emerged from a group of friends who were presented with the opportunity to start in the world of robotics. Although some of them had prior knowledge, they started from the most basic and then made much more functional and advanced prototypes.

The team for the competition is made up of:

Samuel José
Galbán Franco

Juan Diego
Cano Barros

Ángel Saúl
Rodríguez Guerra



At the beginning of this journey, the team came up with the name **"Robstar"**, which was created through a union of "Robotics" and "Tasks". Then, a new challenge was presented, in this case it was a robotics competition, called the copa Ka´i, which, was an interschools event, it was decided that the name **"Maquinaria roja"** would fit perfectly with the team representing the institution. , due to a tradition in all competitions of any kind that the Los Robles high school participated in, it was known as such. Finally, when starting in the **WRO** the team decided that the best name option would be **"Red Machine"** in honor of **"Maquinaria Roja",** for the construction of **"Julian"** robot

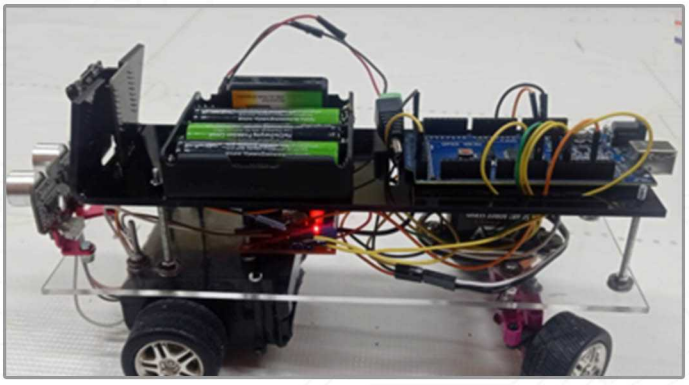
To save and share this process it was decided to open social media on

@redmachine_vzla
@REDMACHINE.WRO.

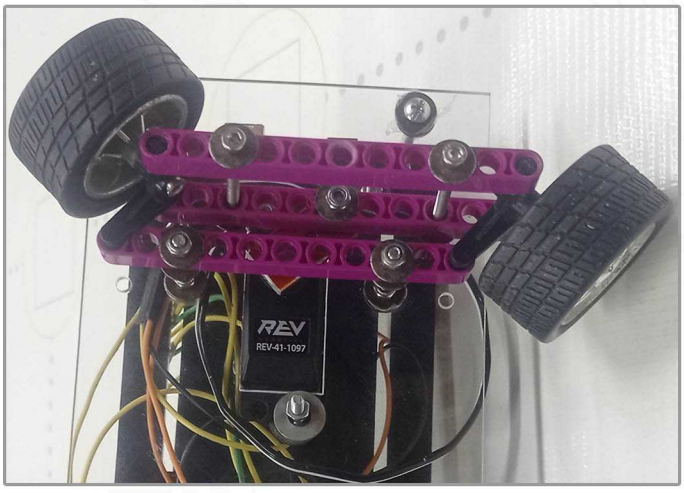
2. DESING PROCESS

The designing process of each of the phases of the project is presented below, describing in detail each of the mechanical, electronic and programming systems.

2. 1.MECHANICAL SYSTEM



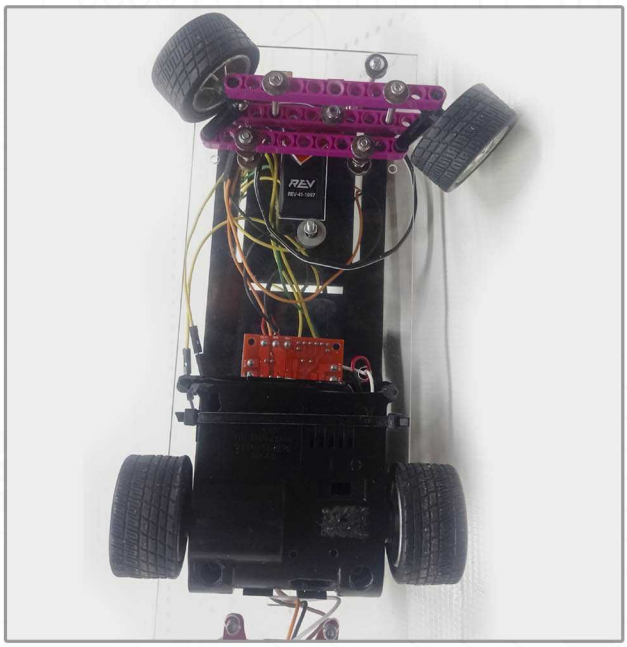
The mechanical system was taken from a remote control **Ford Mustang** model car, from which the driving elements were extracted by coupling to the motors presented in the electronic system.



2. 1. 1. STEERING

The robot's steering subsystem is located in the rear part of it, in order to provide a greater turning radius, it was created with Lego pieces from a robotics kit called **Lego Spike Prime Kit**, joining these pieces with the wheels of a car remote control **(RC) Ford Mustang** model, to which a **Rev Robotics** brand servomotor was adapted.

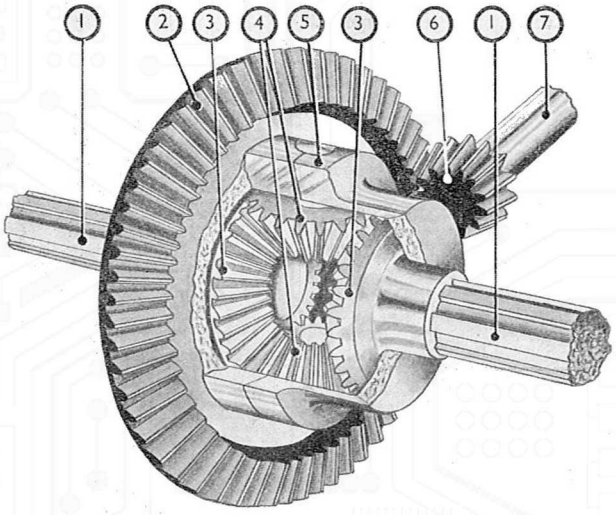
It was later programmed through an Arduino mega 2560, connecting the servomotor to the **GND, 5V** and **4 pins** of the Arduino as can be seen in the following image.



2. DESING PROCESS

2. 1. 2. DRIVING

The robot's driving subsystem was taken from a **Ford Mustang** remote control car. This system was used because it had a motor and a gearbox, and provided the necessary torque and speed for the robot.

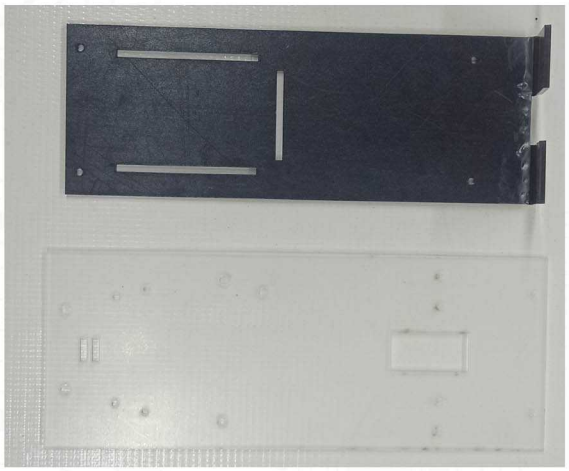


This works with a set of gears, which allows the robot to rotate at different speeds without losing grip. This mechanism was located in the front of the robot. The system is powered through a double H-bridge model **L298n** by means of a pack of three **3.7 V** batteries.

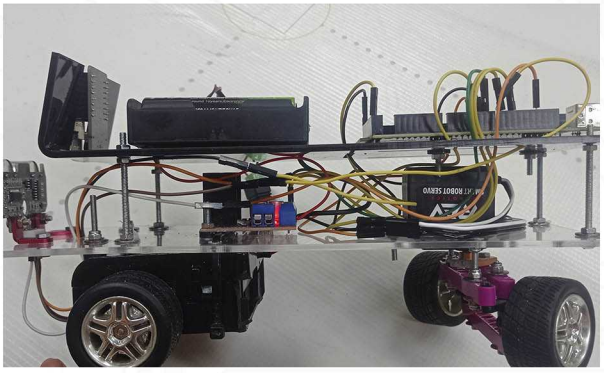
2. DESING PROCESS

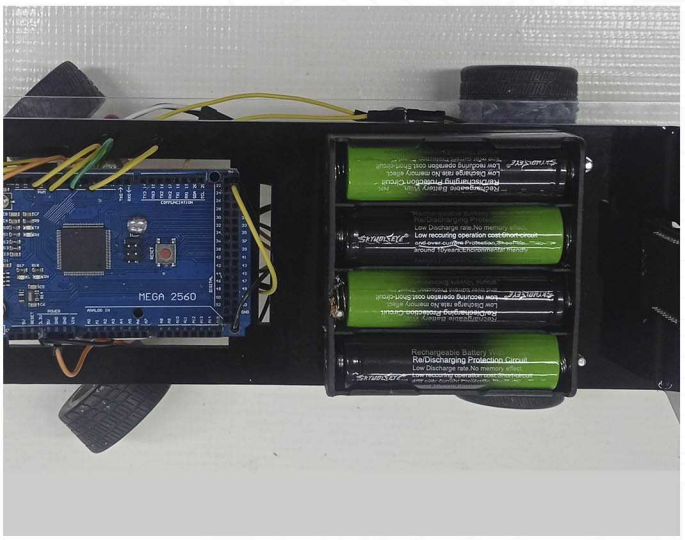
2. 1. 3. CHASSIS DESING

The base of the chassis was built with acrylic, which was cut and drilled based on the needs of the wiring and the assembly of the parts used as can be seen in the image below.



The robot has two decks, both built with acrylic, as shown in the image below.





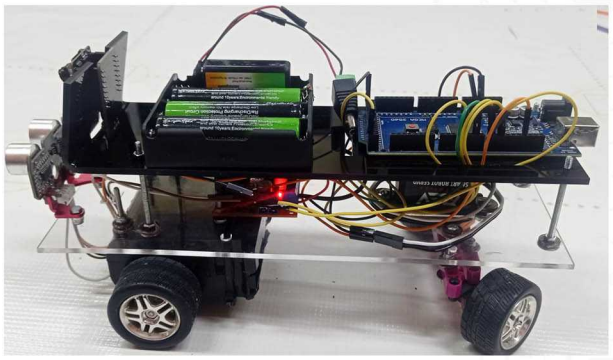
On the **"first floor"** of the robot, the traction and steering systems, the ultrasound sensor, the **TCS 34725 RGB** color sensor, two switches (one for power on and the other for start) and the double H-bridge model **L298n** were assembled.

While on the **"second floor"** there is the battery pack, the **9V battery**, the **ESP-32cam** camera, **Arduino mega 2560**, and the connector for the batteries.

2. DESING PROCESS

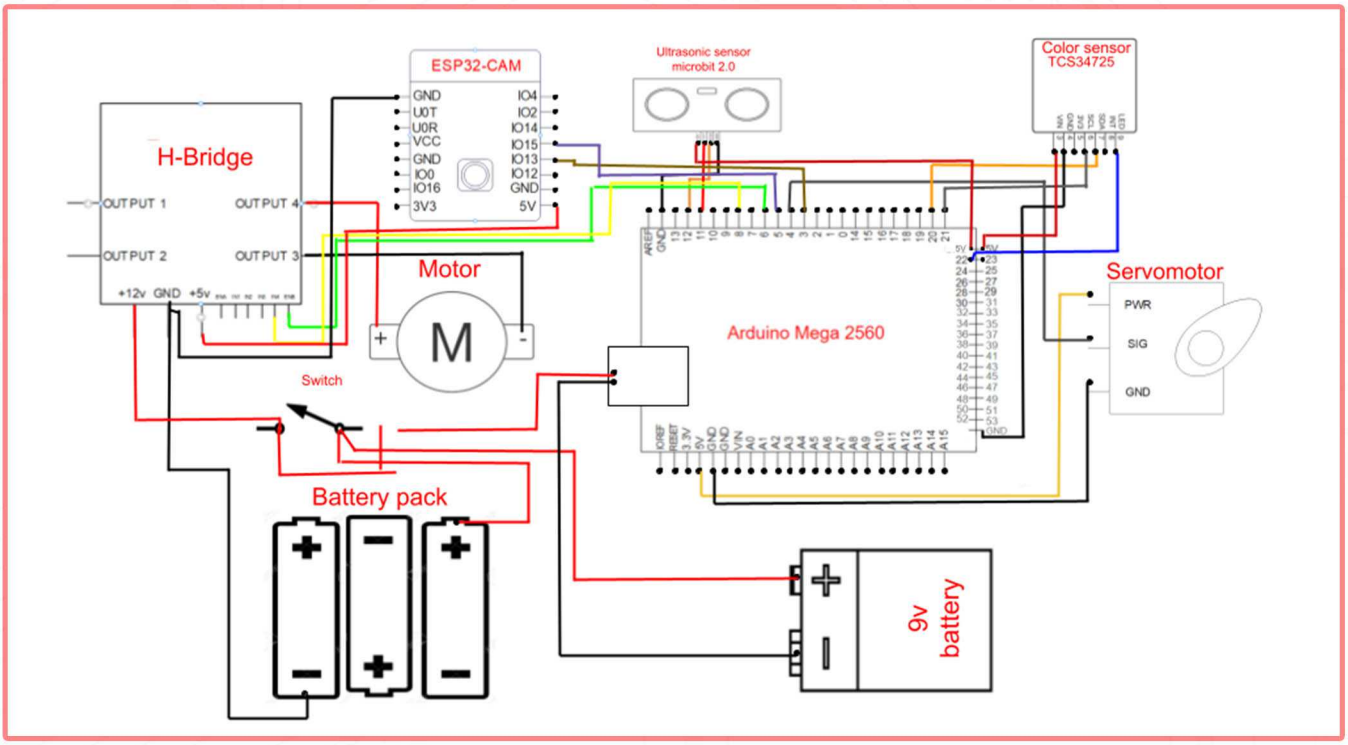
2. 2. ELECTRONIC

The robot´s electronic system is based on an **Arduino Mega 2560** microcontroller with which, according to the original design, three sensors and two actuators are controlled as follows:



- A color sensor model: **RGB TCS-34725.**
- A DC motor type: **RS390 electric motor, 12V.**
- One ultrasonic proximity sensor model: **microbit 2.0**
- A servo motor model: **REV Robotics Smart Robot Servo (SRS).**
- A webcam model: **ESP32-cam.**

Below is the connection diagram of the components that were used in this design.



Each of the sensors are specified in Appendix C - Specification of Sensors and Appendix D - Specification of Actuators.

2. DESING PROCESS

2. 2. 1. SENSORS

Microbit 2.0

The robot originally has an ultrasound sensor.

This sensor is located on the front of the robot, and is used to measure the distance between the robot and the wall, in order to determine when it should cross.

The microbit 2.0 sensor detects distances from **2 cm** to **500 cm** with excellent precision.

This sensor has four pins:

VCC (direct current voltage): is the power input, it receives 5V.

TRIG (trigger): pin that sends the ultrasound pulse.

ECHO (echo): pin that calculates the time it takes for the pulse to return to the ultrasound, in order to determine the distance.

GND (ground): pin that functions as a ground connector.

The operation of the sensor is as follows: the piezoelectric emitter emits eight ultrasound pulses **(40KHz)** after receiving the order on the **TRIG pin**.

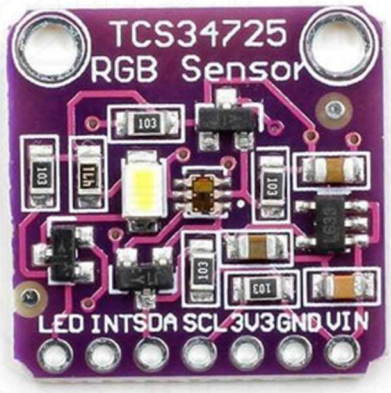
Sound waves travel in the air and bounce when they encounter an object, the bouncing sound is detected by the piezoelectric receiver,

then the **ECHO pin** changes to High **(5V)** for a time equal to the time the wave took from when it was emitted to that was detected. The time of the echo pulse is measured by the microcontroller and thus the distance to the object can be calculated.

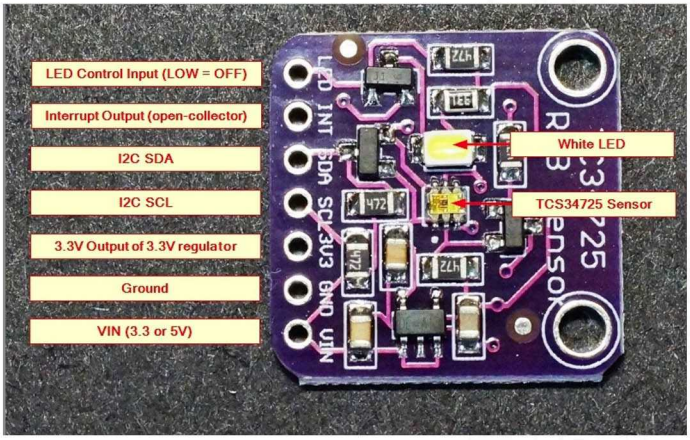
2. DESING PROCESS

RGB TCS34725

The robot has a color sensor.



This is used to identify the colors of the orange and blue lines that are at the corners of the track, in order to determine which direction the robot should cross.



TCS34725

Is an **I2C bus-based** color light-to-digital converter with an **IR filter** that provides digital return of red, green, blue **(RGB)**, and clear light detection values.

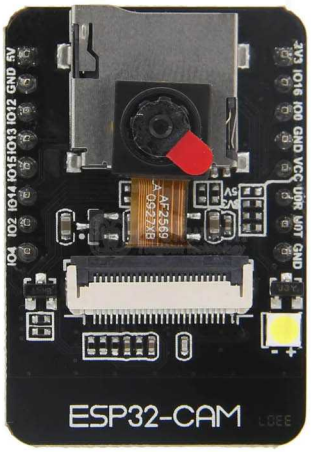
The TCS34725 RGB sensor has the following pins:

- **VIN (input voltage):** it is one of the power supplies of the module that receives **5V.**
- **3V3:** Power supply of the module that receives **3.3V.**
- **GND (ground):** device ground pin.
- **SCL:** used as an **I2C** clock signal.
- **SDA:** used for **I2C** data exchange.
- **INT:** Set I2C address.
- **LED: (Light-emitting diode):** pin that works to turn on the **LED** of the Active Low module.

2. DESING PROCESS

ESP32 - CAM

This camera is used to identify the color of traffic lights **(red or green)**, in order to avoid them correctly.



This camera has the following pins:

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

The camera has three ground pins and two pins for power: either **3.3V** or **5V.**

GPIO 1 and **GPIO 3** are the serial pins that this camera has. You need these pins to load code onto your board. Additionally, **GPIO 0** also plays an important role as it determines whether the **ESP32** is in intermittent mode or not.

The following pins are connected internally to the microSD card reader:

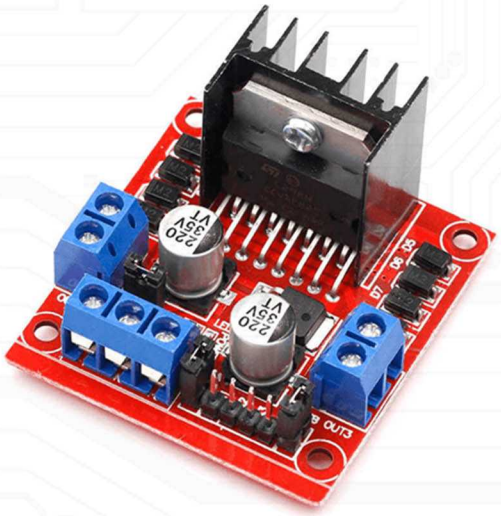
GPIO 14: CLK
GPIO 15: CMD
GPIO 2: Data 0
GPIO 4: Data 1 (also connected to the integrated LED)
GPIO 12: Data 2
GPIO 13: Data 3

2. DESING PROCESS

2. 2. 2. SPEED / DIRECTION CONTROL

The Speed

Is controlled through a double H-bridge model **L298N**, which, thanks to an analog signal sent by the **Arduino Mega 2560**, is capable of controlling the voltage that is emitted to the motors, thus varying and controlling the speed necessary to that the robot meets the challenge.



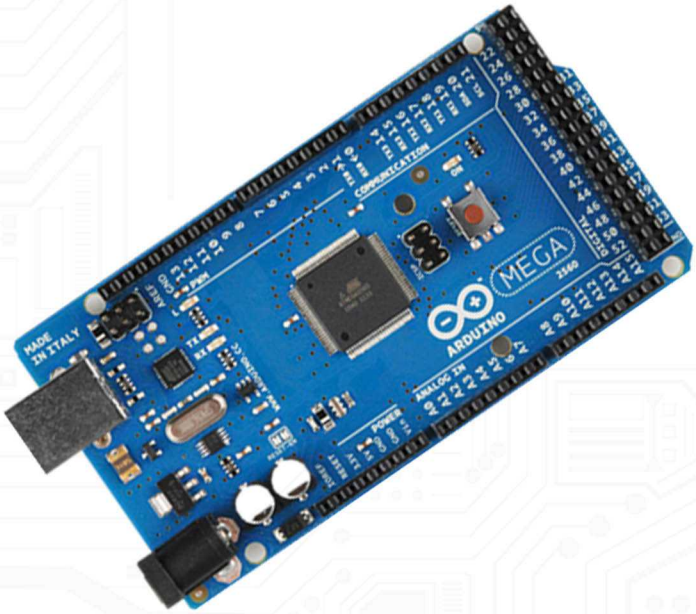
The Direction

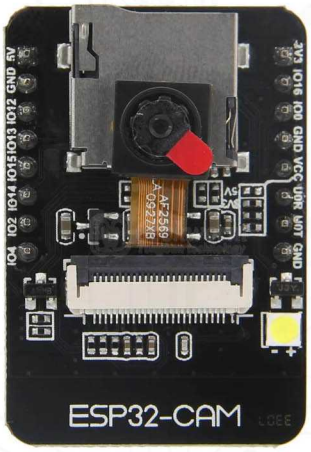
Is controlled with a **smart servo robot** created by the company **Rev Robotics,** which, through an analog signal sent by the **Arduino mega 2560**, controls its turning radius to achieve an optimal change of direction.

2. DESING PROCESS

2. 3. PROGRAMMING

The robot has two programmable devices:





2.3.1. The **Arduino Mega 2560** which is used for general control of the robot devices.

Regarding the algorithms shown in the flowchart, only the one used in the loop() section of the Arduino Mega is being presented since in the setup() section, only configurations and declarations were made.

2.3.2. The **ESP32 Cam** camera whose function is fundamentally to detect the colors of traffic lights.

In both cases, the Arduino IDE software (https://www.arduino.cc/en/software) was used as a development environment taking advantage of the USB connections available on both devices.

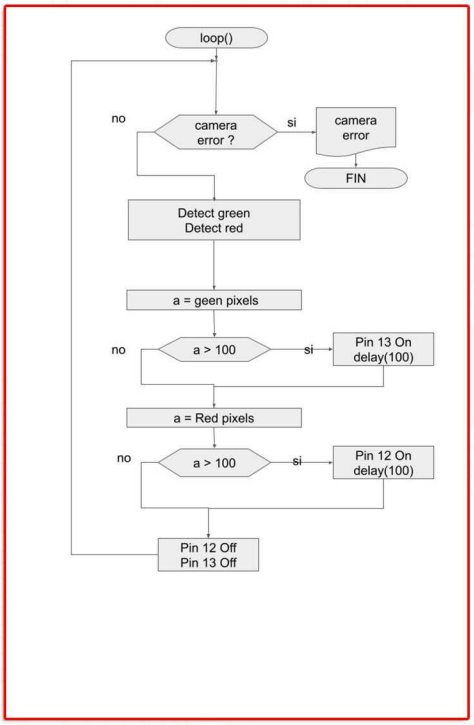
2. DESING PROCESS

2. 3. 1. FLOW DIAGRAM

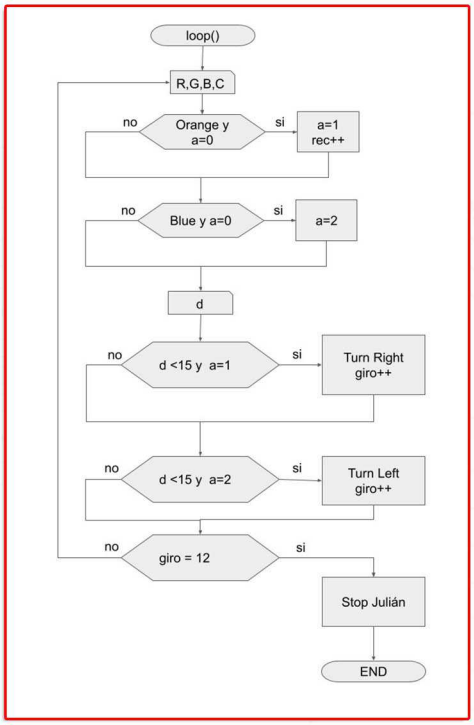
The following graph shows the flow chart of the first and second challenge, in which the robot in Challenge number one must complete three laps on the track with random locations within the track walls.

In the challenge number two, the robot must complete three laps on the track with green and red signs placed randomly within the walls of the track.

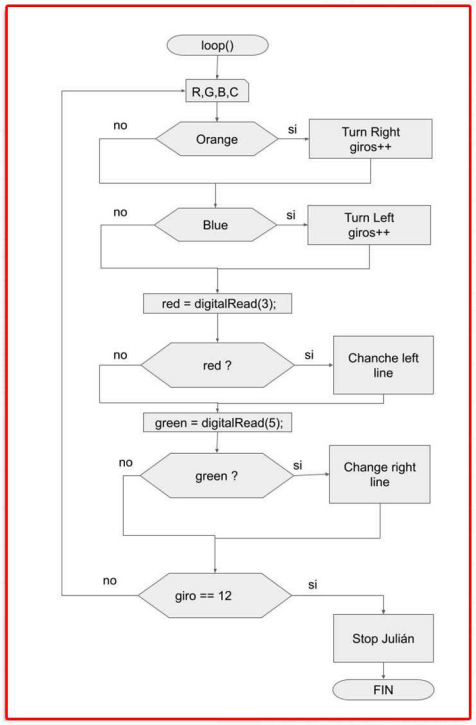
Camara Code



First Challenge



Second Challenge



3. STRATEGIES

The strategy is one of the fundamental parts of the entire challenge, therefore, in the creation process it was the task for the entire team to find and define the best methods based on the components.

As a first strategy, the work team decided that it would be best to focus on challenge number one. In this it was considered that:

The biggest challenge would be to define the direction. To do this, the work team used the orange and blue lines found on the court to determine whether to make a clockwise or counterclockwise turn.

Under this, as the team already knew that after the first turn, the others would be in the same direction, it was decided that after the detection of the color sensor, the robot would rotate with the ultrasound sensor.

As a second strategy for challenge number two, the **ESP32-cam** camera and the ultrasound sensor are used to define:

The color of the traffic lights and how the robot should avoid them, to avoid crashing and deviating from the road.

With the information obtained from the ultrasound, the position of the robot can be calculated and repositioned if necessary.

4. CODE DISCUSSION

The programming language used to prepare the code is **C++** developed for the autonomous operation of the robot and subsequent discussion. The developed codes can be seen in Appendix E.

The code is essentially divided into six parts, which would be the following:

The libraries.

The libraries used were **"Servo.h"** in charge of facilitating the task of mobilizing the servo and in general the direction and **"Adafruit_TCS34725.h"** in charge of facilitating color detection by the "RGB **TCS-34725"** sensor.

Pin declaration.

The function of each of the **Arduino Mega 2560** pins is defined, so that most are found in **"OUTPUT"** (output) and some in **"INPUT"** (input) depending on needs.

Color Sensor Initialization

The color sensor is turned on and then checked using the serial monitor, confirming if there is any type of problem when starting up.

Color detection

Once the **"Setup"** process has been completed, the RGB values defined by the color sensor begin to be saved in variables called **"R", "G", and "B"** that can be used using an **"IF"** command. to make certain steering movements.

Distance detection

When defining distances, to do it more efficiently we established it in the ping() function; so that in this way it is much easier to carry out actions based on that.

Movements to perform

Whenever the color orange or blue is detected in the color sensor, the robot will move clockwise or counterclockwise respectively, at an angle of 90 degrees, where this movement is efficient for carrying out the three turns, which are all completely important when programming the robot. Being programmed in the **Arduino IDE**, everything is in the **C++** language.

5. DIARY ENTRIES

5. 1. CHRONOLOGY
CHALLENGES AND SOLUTIONS

1st week of July

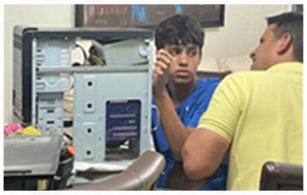
On the first day of preparation for the regional competition the team began to study and analyze the rules of the competition.

The following days the team began to study what could be the first model of the chassis and investigated various ways to solve what would be the first problems that arose, which were how the steering system would be designed and what engine would be used to achieve the speed and torque needed.

5. DIARY ENTRIES

Later the team began to **look for motors** that could be used, disassembling toys, printers, among other devices.

Finally, the necessary engine was obtained by disassembling a Nikko brand **Dodge T-Rex Ram** remote control car, which provided the necessary mechanical parts to design the steering system.

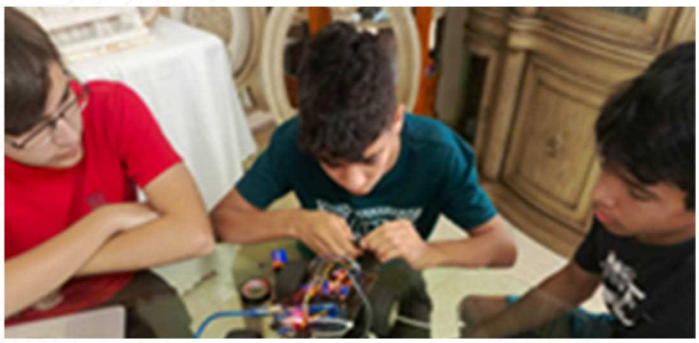


2nd week of July

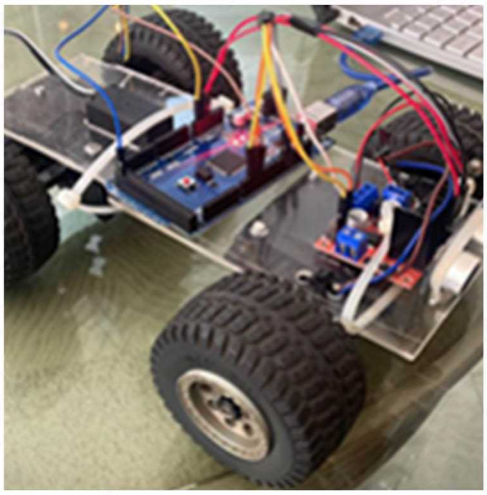
Once the problem of the traction system was solved, the problem of the steering system was approached, and after investigating different solutions, a way was found to fit a servo motor to the mechanical parts previously obtained.

5. DIARY ENTRIES

Subsequently, the work team proceeded to assemble both systems, and the different devices that the robot would need on acrylic bases.

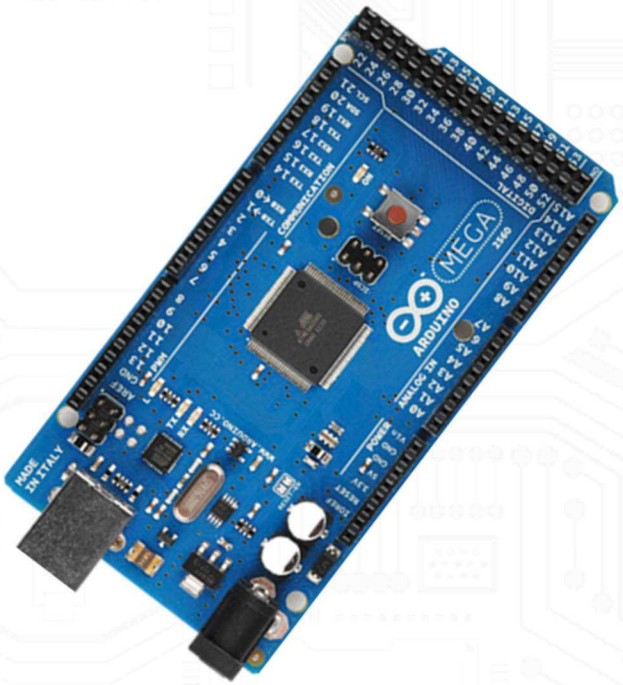


The team finished the **first prototype** of what would be the chassis, and were able to continue with the programming area.



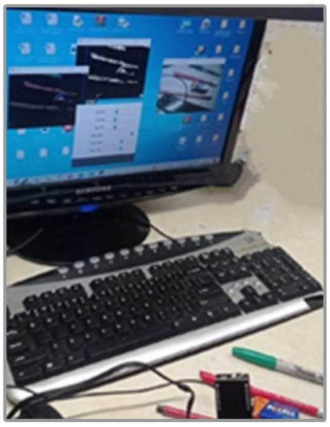
3rd week of July

For programming, an **Arduino Mega 2560** was used as a programmer, a double **H-bridge** as a power and speed regulator and an ultrasound sensor to measure distance.



5. DIARY ENTRIES

Consecutively, the work team resumed the search for different solutions to **detect the colors** of traffic lights, deciding to use an **ESP32-cam** camera with an **OV2640** lens, the next problem being how to program this with Arduino.



It was investigated what power source was going to be used to power the robot, because after using **9V** batteries, the team realized that these were not ideal for the robot because they wear out in a very short time.

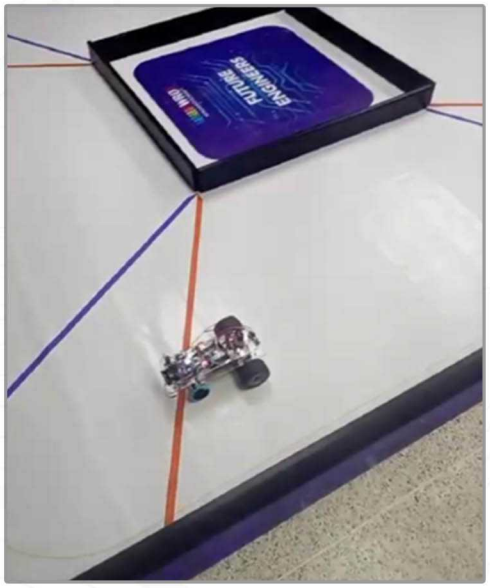
Consequently, two battery packs ended up being joined, which had eight **1.2 V** rechargeable batteries in series, which finally added up to a total of **9.6 V.**

Due to the space needed, a **second prototype** was designed in which it was decided to add a second floor to the robot, locating the electronics area in it, and the batteries, the traction system and the steering system on the first floor.

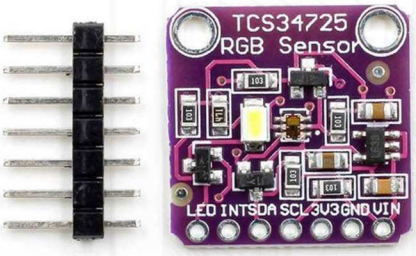
5. DIARY ENTRIES

4th week of July

Without yet being able to solve the programming, it was decided to use a **34725 RGB** sensor so that the robot could detect which direction it should cross.



Before the regional competition, the work team found rechargeable batteries with a higher voltage **(3.7 V)**, so it was decided to eliminate one of the two battery packs, and modify the remaining one so that it works with three batteries.



A strategy is proposed, it is decided to cross with the color detection of the lines of the track, also using two more ultrasounds, one on each side of the robot, so that, once it detects a wall, it crosses to avoid a crash.

However, these two ultrasounds were not a help, but rather a problem, since, when they detected, the robot lost its trajectory, which is why it was finally decided not to use these two ultrasound sensors.

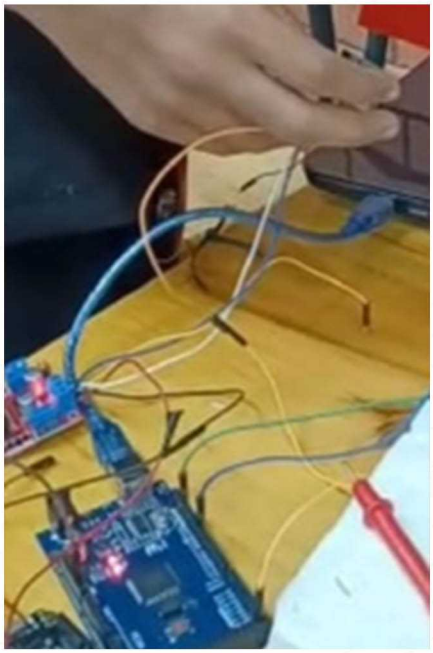
5. DIARY ENTRIES

1st weet of August

After participating in the First Regional Competition, the research for solutions to the problems presented began. It was decided to change the **steering system, creating** a new one with parts obtained from a **Spike Prime** robotics kit number **45678**, since with this new steering system it would be possible to have a greater turning radius, as well as a more precise turn.

2nd week of August

New strategies were proposed, it was decided that the ideal way to cross would be with the help of ultrasound and that the **TCS34725** sensor would only detect the first line to determine if the robot should cross clockwise or counterclockwise.



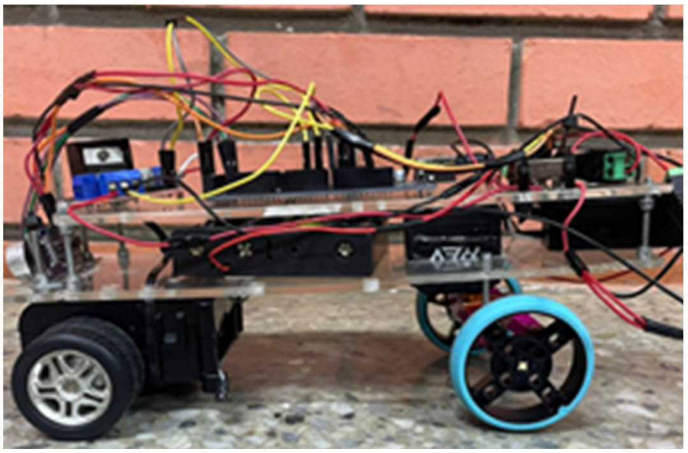
3rd week of August

Following with the second part of the challenge, the team began to program the camera looking for how to transfer the information from the camera to the Arduino without having to use **Wi-Fi**. After searching several sources, the solution was found by transmitting the information through serial ports.

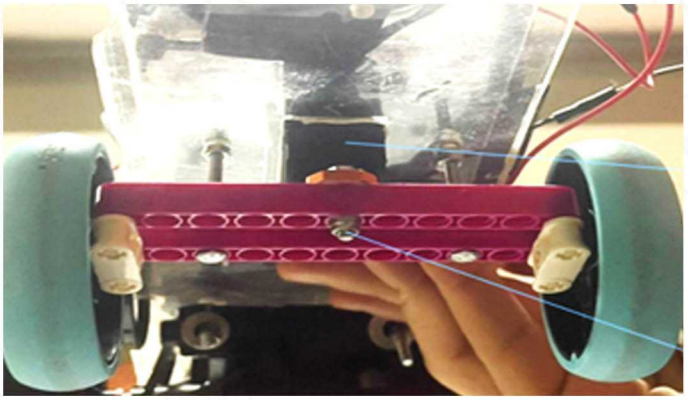
5. DIARY ENTRIES

4th week of August

Then the engine used for all this time began to fail very frequently, preventing the team from progressing in the second challenge, so with just a few days left before the competition, the work team removed the engine and therefore the gearbox from another remote control car to attach it to the robot.

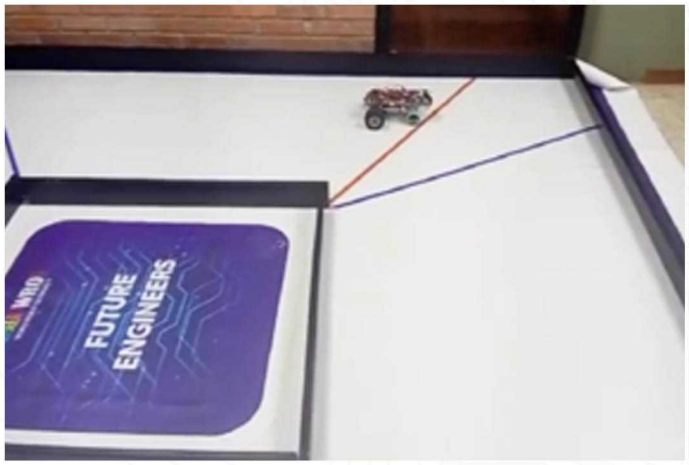


To provide a greater turning radius, the steering system was modified, using pieces from a **Lego Spike Prime Kit.**



1st week of September

Practices were carried out on the track, for the performance of the robot in challenge number one and two.



2nd week of September

Preparation and updating of the report.

5. DIARY ENTRIES

3dr week of September

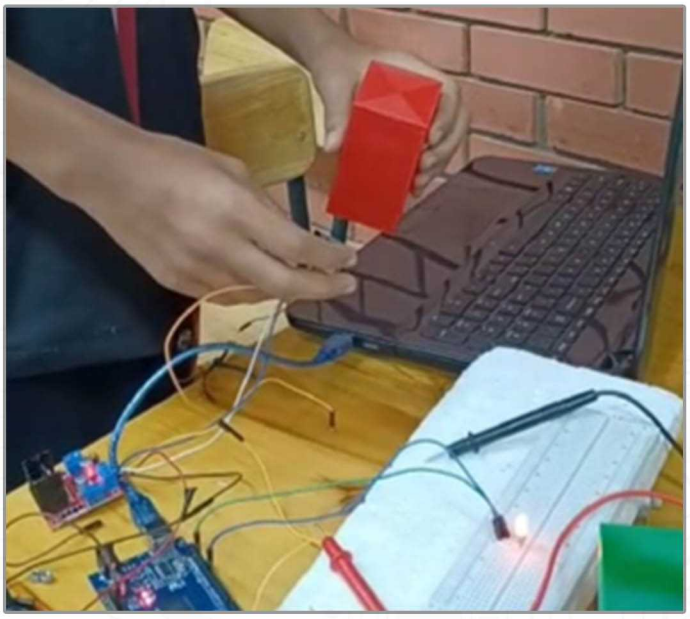
National Competition.

National Competition.

5. DIARY ENTRIES

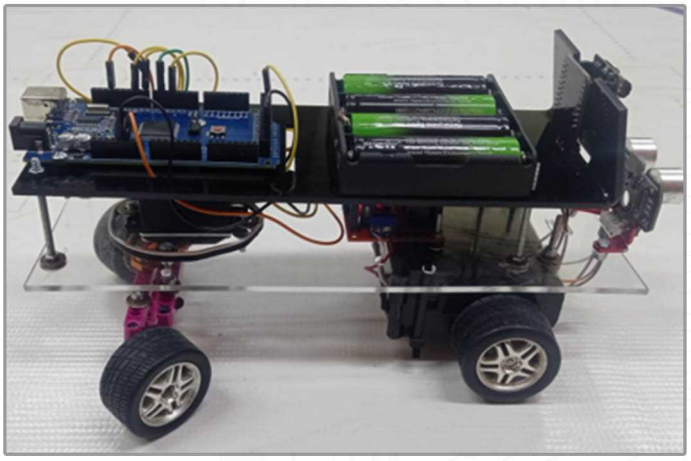
4th week of September

The camera programming for challenge number two was continued.



Replacement of steering system wheels to increase safety and aesthetics.

1st week of October



Construction of the third prototype of the robot, which is based on the restoration of the acrylics, organization of the cables through connectors to improve the aesthetics.

5. DIARY ENTRIES

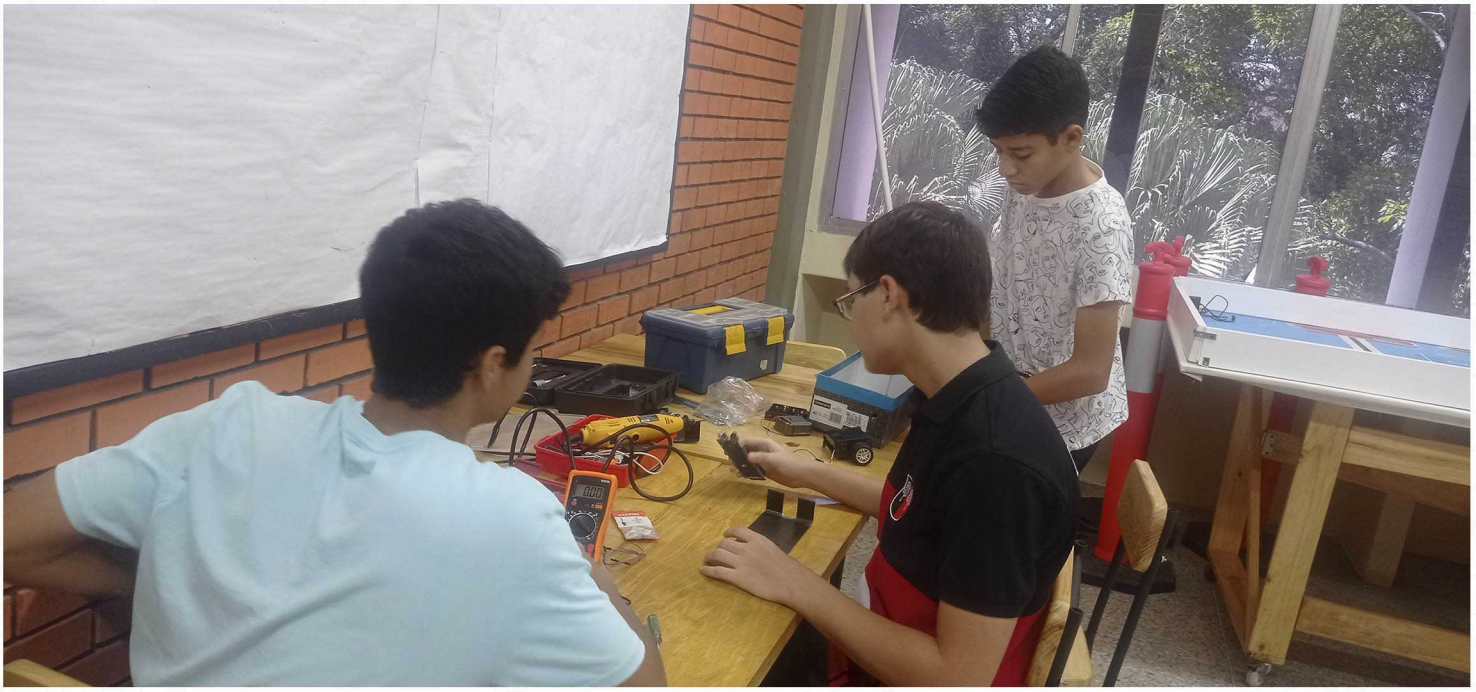
2nd week of October

Programming the camera and practicing with it.

Looking for camera options with better image clarity and width of vision. Testing is done with **Raspberry Pi card** and wide-angle webcam.

Raspberry PI board

wide angle camera



Team working

6. IMPROVEMENT OPPORTUNITIES

The functions of the robot presented are sufficient to overcome all the challenges posed by the category. However, the following opportunities for improvement were identified, and will be carried out on future competitions.

Add two ultrasonic sensors for distance detection:

The aim is to avoid lateral collisions, therefore a sensor should be placed on each side of the robot.

Add connectors:

Adding a greater number of connectors makes it easier to change parts and improve the organization of dupont cables.

Build a bodywork:

It is an important element which will provide better aesthetics of the robot.

External luminosity control:

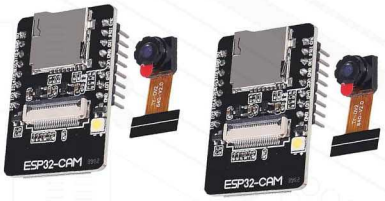
Place a high-brightness LED, controlled by the controller card, and black side dimmers on the camera lenses to control external lighting.

Add a Raspberry Pi 3 with a webcam:

Mainly we want to add the Raspberry since it allows us to use a webcam, in order to have a greater angle for detecting traffic lights.

Improve the viewing angle of the Robot:

It consists of adding another ESP32 cam that, when properly placed, not only improves the viewing angle, but also favors the identification of traffic lights.

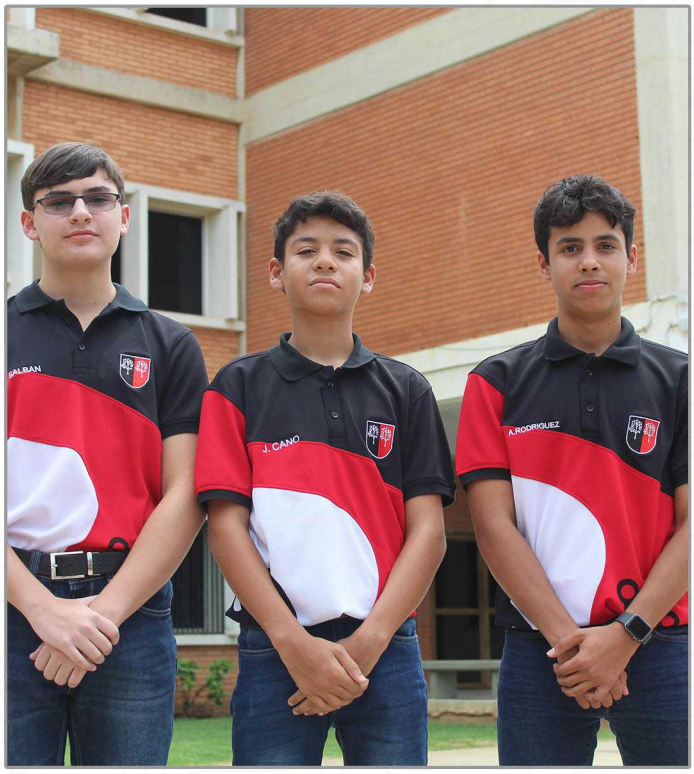


CONCLUSION

The work team managed to develop an autonomous robot **"Julian"** capable of carrying out the two challenges required for the Future Engineers category. To achieve this, a robot equipped with different electronic and mechanical components was used, in turn meeting the specifications established in the competition premises.

During the process, they will face several challenges, such as detecting and avoiding obstacles, completing the three laps and detecting the lines on the track.

The results obtained demonstrate the effectiveness of this approach in controlled environments and the robot's ability to adapt to changing situations.

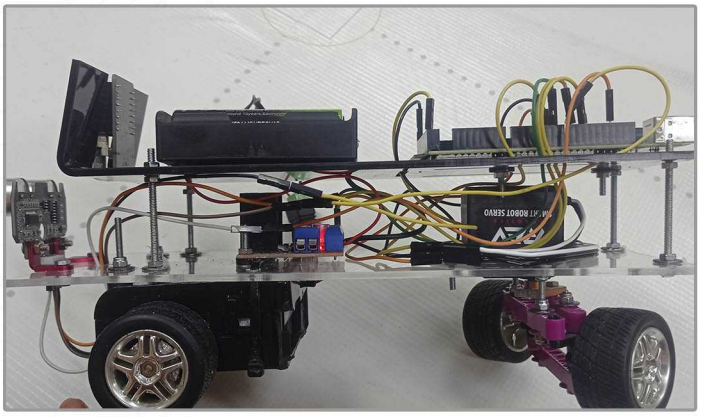


During these months of hard work, the team has managed to meet its objectives in terms of the competition, as well as each of the participants deepened their knowledge in the various areas in which the WRO was deepened, such as programming, electronics, the mechanics, among others.

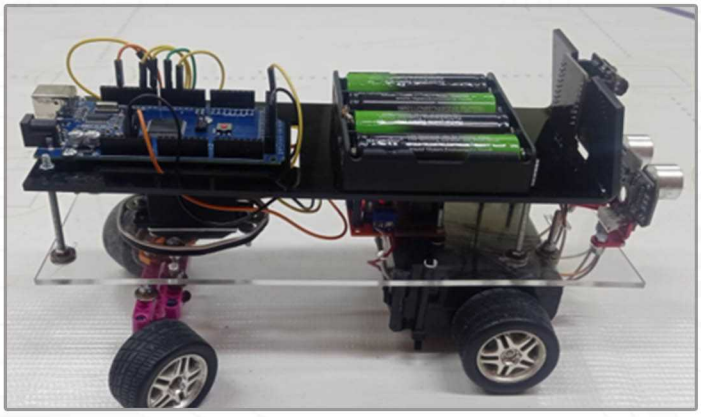
APPENDICES

Appendix A:

Views of the robot car (photos)

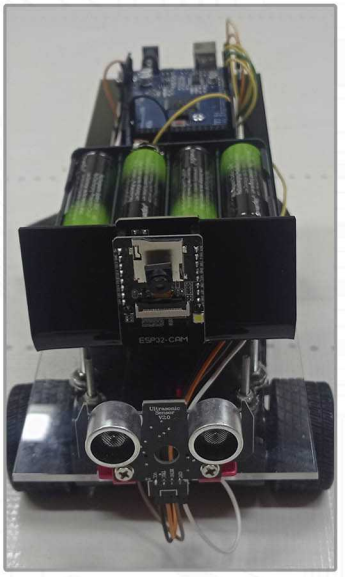


Left Side View

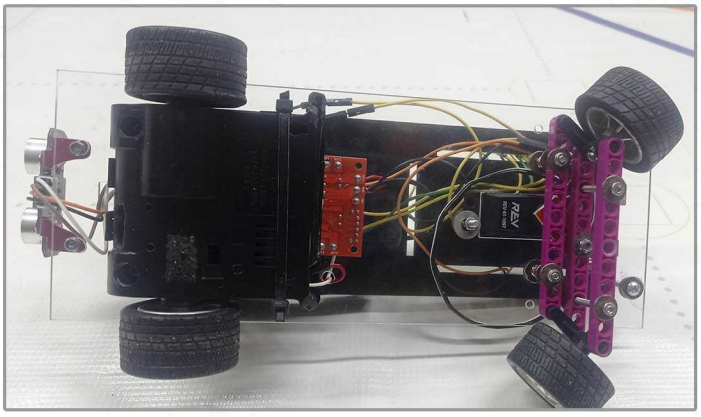


Right Side View

Back View



Frontal View



Inferior View

Superior View

APPENDICES

Appendix B:

Performance Videos

Challenge 1

https://youtu.be/8zjeBPan2Oo?si=eekGrBMWVPs4Mw3y

Challenge Test 2

https://youtu.be/8zjeBPan2Oo?si=UVurdqN9m1lv_jjU

APPENDICES

Appendix C: Sensor Specification

Microbit 2.0 Ultrasonic Ranging Module

Source: https://www.okystar.com/product-item/smart-car-robot-wireless-ultrasonic-sensor-distance-ranging-obstacle-for-microbit-oky6010/

Product characteristics:

The **microbit 2.0** ultrasonic ranging module provides non-contact measurement function from **2 cm** to **500 cm**, the ranging accuracy can reach **3 mm**. The modules include ultrasonic transmitters, receiver and control circuit.

The basic working principle:

(1) Using the IO trigger for at least 10us of high level signal.

(2) The module automatically sends eight **40 kHz** and detects whether there is a return pulse signal.

(3) IF the signal returns, via high level, the high **output IO** duration time is the time from ultrasonic send to return.
Test distance = (high level time × sound speed **(340M/S) / 2.**

Electrical parameters

Working voltage	DC 5V
Working current	15 mA
Work frequency	40 Hz
Maximum range	4 m
Minimum range	2 cm
Angle measurement	15 degree
Trigger input signal	10 US TTL PULSE
Echo output signal	TTL toggle signal and range in proportion
Dimension	45mm x 20 mm x 15 mm

APPENDICES

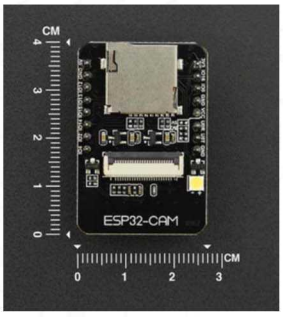
Appendix C: Sensor Specification

ESP 32 - cam Camera

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

CHARACTERISTICS

- Clock speed up to 160 MHz, summary computing power up to 600 DMIPS
- Built-in 520KB SRAM, external 4MPSRAM
- Support UART/SPI/I2C/PWM/ADC/DAC-
- Support OV2640 and OV7670 cameras, built-in flash lamp.
- Support WiFI image upload-Support TF card
- Supports multiple sleep modes.
- Integrated Lwip and FreeRTOS
- Support STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for remote and local serial port (FOTA) firmware updates



APPENDICES

Appendix C: Specifications

- **SPI Flash:** default 32Mbit
- **RAM:** built-in 520KB+4MPSRAM external
- **Dimension:** 27*40.5*4.5 (±0.2)mm/1.06*1.59*0.18”
- **Bluetooth:** Bluetooth 4.2 BR/EDR and BLE standards
- **WiFi:** 802.11b/g/n/e/i
- **Support interface:** UART, SPI, I2C, PWM
- **TF card support:** maximum support 4G
- **I/O port:** 9
- **Serial port baud rate:** Default 115200 bps
- **Image output format:** JPEG (only compatible with Ov2640), BMP, grayscale
- **Spectrum range:** 2412 ~ 2484 MHz
- **Antenna:** Integrated PCB antenna, 2dBi gain
- **Transmission power:**
  - **802.11b:** 17±2 dBm (@11Mbps);
  - **802.11g:** 14±2 dBm (@54Mbps);
  - **802.11n:** 13±2 dBm (@MCS7)
- **Reception sensitivity:**
  - **CCK, 1Mbps:** -90dBm;
  - **CCK, 11Mbps:** -85dBm;
  - **6Mbps (1/2 BPSK):** -88dBm;
  - **54Mbps (3/4 64-QAM):** -70dBm;}
  - **MCS7 (65Mbps, 72.2Mbps):** -67dBm
- **Power consumption:**
  - **Turn off flash:** 180mA@5V
  - **Turn on the flash and set the brightness to maximum:** 310mA@5V
  - **Deep sleep:** lowest power consumption can reach 6mA @ 5V
  - **Modern-sleep:** up to 20mA@5V
  - **Light sleep:** up to 6.7 mA at 5 V
- **Security:** WPA/WPA2/WPA2-Enterprise/WPS
- **Power supply range:** 5V
- **Operating temperature:** -20°C ~ 85°C
- **Storage environment:** -40°C ~ 90°C, < 90% RH
- **Weight:** 10g

APPENDICES

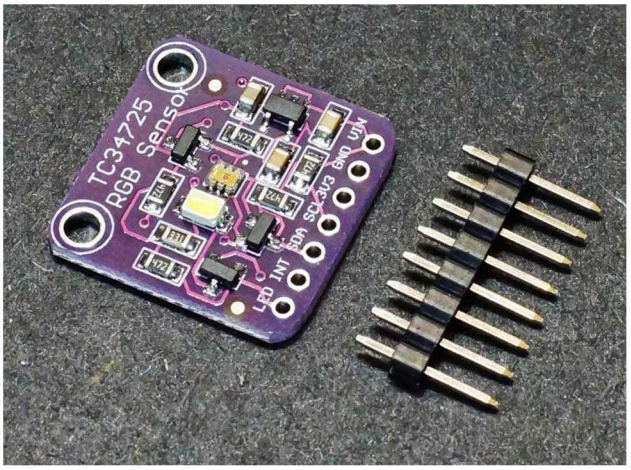
Appendix C: Sensor Specifications

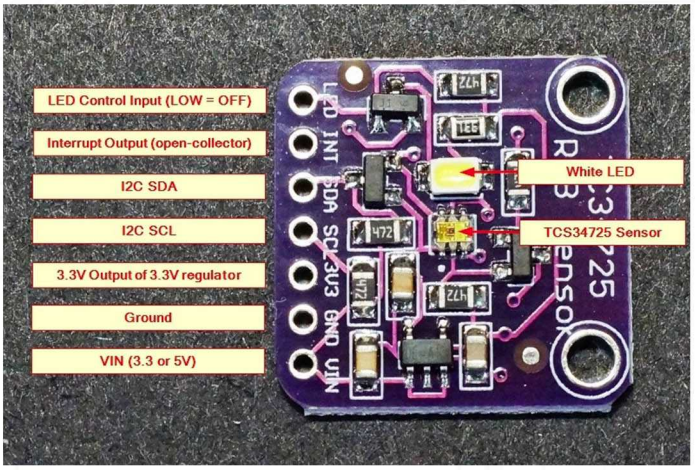
TCS34725 SENSOR RGB

Source:
https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf

CHARACTERISTICS

- **Red, green, blue (RGB) and clear light**
  Detectionwith infrared blocking filter
  - Programmable analog gain and integration time
  - **3,800,000**:1 Dynamic Range
  - Veryhigh sensitivity: ideal for operation behind dark glass
- **Maskable interrupt**
  - Programmable upper and lower thres hold swith persisten cefilter
- **Powermanagement**
  - Low power: **2.5 A** sleep state
  - **65-A** Stand by state with programmable stand by Status
  - Time from**2.4 ms to > 7 seconds**
- **Interface compatible with 2C**
  - Fast mode – Data rates up to **400 kbit/s**
  - Input voltage levels supported by **VDD** or**1.8 V bus**
- **Registrationand Pin Set Compatible** with the **TCS3x71** Series
- **Small** 2mm 2.4mm **DoubleFlat** Lead Free **(FN)** Package





APPENDICES

Appendix D: Actuator Specifications

Electric Motor Rs 390

Source:
https://es.aliexpress.com/i/32891103572.htmlgatewayAdapt=Msite2Pc

Characteristics

- **Model:** Rs390
- **Nominal voltage:** 6V
- **Power:** 25-35W
- **No-load speed:** 14000/ min
- **Size:** approx. 70*28mm
- **Axis:** 2mm

Smart Robot Servo

Source:
https://www.revrobotics.com/rev-41-1097/

Characteristics

- **Size:** 40.2mm x 20.0mm x 38.0mm
- **Weight:** 2.05 ounces.
- **Speed:** 0.14 s/60° (at 6 V): **Stalling torque:** 13.5 kg-cm / 187.8 oz-in (at 6 V)
- **Standby current:** 2A (at 6V)
- **Nominal voltage:** 4.8V - 7.4V, 6V nominal
- **Input pulse range:** 500 µs - 2500 µs
- **Default angular range:** 270°
- **Maximum programmable range in angular mode - 280°**
- **Gear Material:** Metal
- **Knurled type** - 25T
- **Spline Internal Thread Size:** M3
- **Spline internal thread depth:** 6mm

APPENDICES

Appendix E: Arduino Codes

Challenge code 1

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

Servo pro;
int d = 16; // Distancia ultrasonido izquierdo
int a;
int rec= 105; // Varor del servo para avarzar dececho
int da=16; // Distancia ultrasonido derecho
int t; // ultrasonido tiempo de rebote izquierdo
int te; // 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(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);// Trigger
  pinMode(11, INPUT);  //cho
  pinMode(12, OUTPUT); //Trigger
  pinMode(13, INPUT);  //Echo
  pinMode(22, OUTPUT);
  pro.attach(4);  // Pin de conwtrol del servo
  digitalWrite(22,HIGH);
  Serial.begin(9600);
  if(tcs.begin()){
    Serial.println("Inicializando");
    delay(2000);
  }
  else{
    Serial.println("Error");
    Serial.println("Debe de 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)){
    digitalWrite(8,LOW);

    pro.write(vel);

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

  analogWrite(9,vel);
  pro.write(rec);
  }
 //AZULLLL
  if((R<90)&&(G<110)&&(B<130)){
    pro.write(rec);
    analogWrite(9,0);
    delay(750);
    digitalWrite(8,HIGH);
    analogWrite(9,110);
    pro.write(114);
    delay(2100);
    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 utrasonido izquierdo y cálculo de distancia
int ping() {
  digitalWrite(10, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(10, LOW);

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

// Control del sensor de utrasonido derecho y cálculo de distancia
int ping1() {
  digitalWrite(12, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(12, LOW);

  te = pulseIn(13, HIGH); //obtenemos el ancho del pulso
  da = te/59;
}
```

Challenge code 2

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


Servo pro;
int b;
int d = 0; // Distancia ultrasonido izquierdo
int a = 0;
int giro = 0 ;
int rec= 73; // Varor del servo para avarzar dececho
int da=16; // Distancia ultrasonido derecho
int t; // ultrasonido tiempo de rebote izquierdo
int te; // ultrasonido tiempo de rebote derecho
int vel= 35;
int gi=vel;
int red = 0;
int green = 0;
void setup()
{
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, OUTPUT);
  pinMode(5, INPUT);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);// Trigger
  pinMode(11, INPUT);  //cho
  pinMode(12, INPUT); //Trigger
  pinMode(13, OUTPUT);  //Echo
  pinMode(22, OUTPUT);
  pinMode(53,INPUT);
  pinMode(52,INPUT);

  pro.attach(4);  // Pin de conwtrol del servo
  digitalWrite(22,HIGH);
  Serial.begin(9600);
   if(tcs.begin()){
    Serial.println("Inicializando");

    delay(2000);
  }
  else{
    Serial.println("Error");
    Serial.println("Debe de revisar las conexiones e iniciar nuevamente");
    while(1)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)&&(C<400)){
    pro.write(100);
    analogWrite(6,gi);
    delay(1150);
  pro.write(rec);
  delay(700);
  analogWrite(6,vel);
  a=1;

  }
 //detectar giro horario(Azul)
   if((R<90)&&(C<400)){
  pro.write(40);
     analogWrite(6,gi);
    delay(1150);
  pro.write(rec);
  delay(700);
  analogWrite(6,vel);
  a=1;
   }


   green = digitalRead(5);

   if(green == HIGH){
  pro.write(50);
  delay(600);
  pro.write(rec);
  delay(400);
  pro.write(100);
  delay(300);
  pro.write(rec);
   }
   red = digitalRead(3);

   if(red == HIGH){
  pro.write(50);
  delay(600);
  pro.write(rec);
  delay(400);
  pro.write(100);
  delay(300);
  pro.write(rec);
   }

   //detectar 3 vueltas y detenerse
   if (giro == 12){
  analogWrite(8,0);
  a=40;
  }

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

  t = pulseIn(12, HIGH); //obtenemos el ancho del pulso
  d = t/59;
}
```

Camera code

```
// 20_Color_Blob_Detector.ino

/**
 * Detect blobs of given color
 */
#define MAX_RESOLUTION_VGA
#define MAX_RECURSION_DEPTH 13

#include "esp32cam.h"
#include "esp32cam/JpegDecoder.h"
#include "esp32cam/apps/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(33, OUTPUT);
   Serial.begin(115200);
   delay(3000);
   Serial.println("Init");
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
   cam.aithinker();
   cam.highQuality();
   cam.vga();
   cam.highestSaturation();
   cam.disableAutomaticWhiteBalance();
   cam.disableAutomaticExposureControl();
   cam.disableGainControl();

   /**
    * Set detector tollerance
    * The higher, the more shade of colors it will pick
    */
   detector.tollerate(47);
    detector2.tollerate(35);
   /**
    * Skip blob localization (slow) if not enough
    * pixels match color
    */

   while (!cam.begin())
      Serial.println(cam.getErrorMessage());
}


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

   if (!decoder.decode(cam)) {
      Serial.println(decoder.getErrorMessage());
      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 verde ");
   Serial.print(detector.maskCount);
    a = detector.maskCount;

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

      Serial.print("\n rojo ");
   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);
}
```

BIBLIOGRAPHIC REFERENCES

- **NIKKO AUTO DRIVE:**
  https://www.youtube.com/watch?v=9tLaDAoxxio

- **Smart Car Robot Wireless Ultrasonic Sensor Distance Ranging Obstacle for Microbit 2.0**
  https://www.okystar.com/product-item/smart-car-robot-wireless-ultrasonic-sensor-distance-ranging-obstacle-for-microbit-oky6010/

- **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%20colo

- **OKY6010 ultrasonic sensor:**
  https://www.okystar.com/product-item/smart-car-robot-wireless-ultrasonic-sensor-distanc