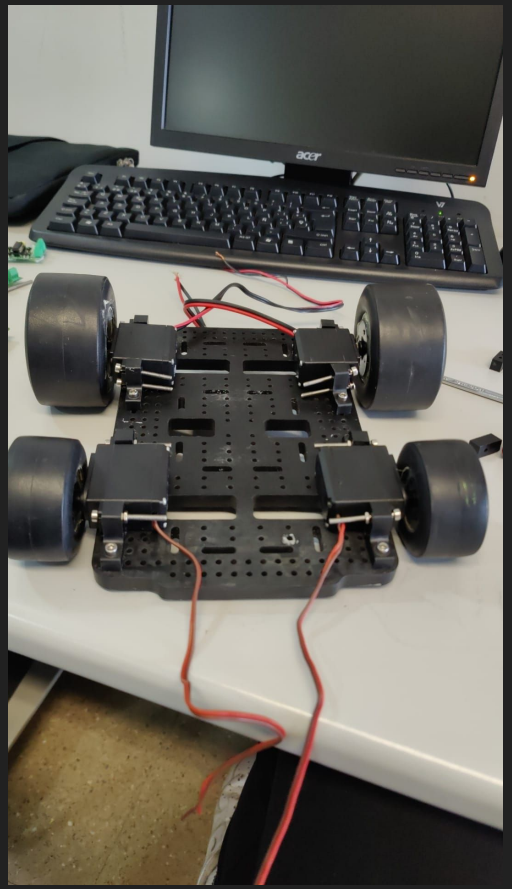
RAYO MCQUEEN

Juan Antonio Martínez Sánchez
Pablo Huertas Arroyo

HARDWARE

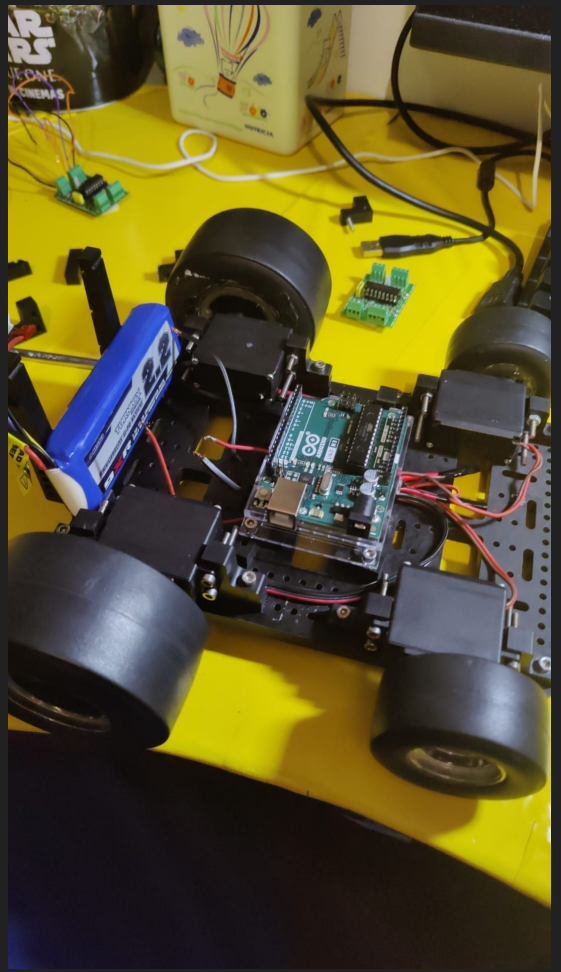
Unión del chasis con las ruedas y con los servomotores



HARDWARE

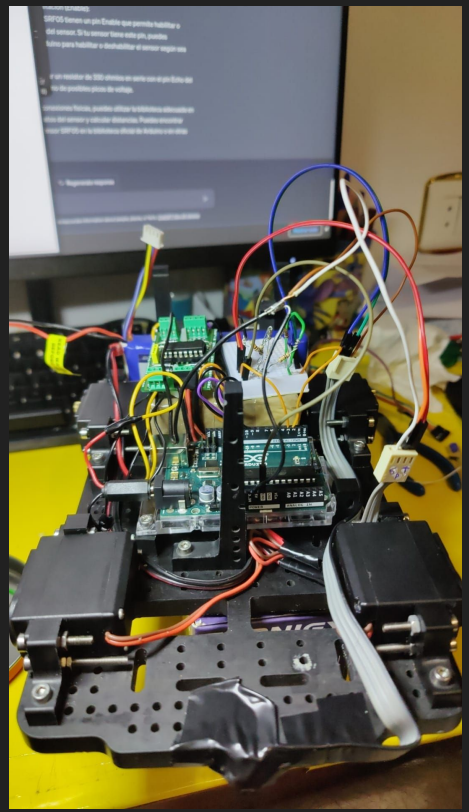
Incorporación de la batería

Incorporación del puente H y primeros giros de ruedas



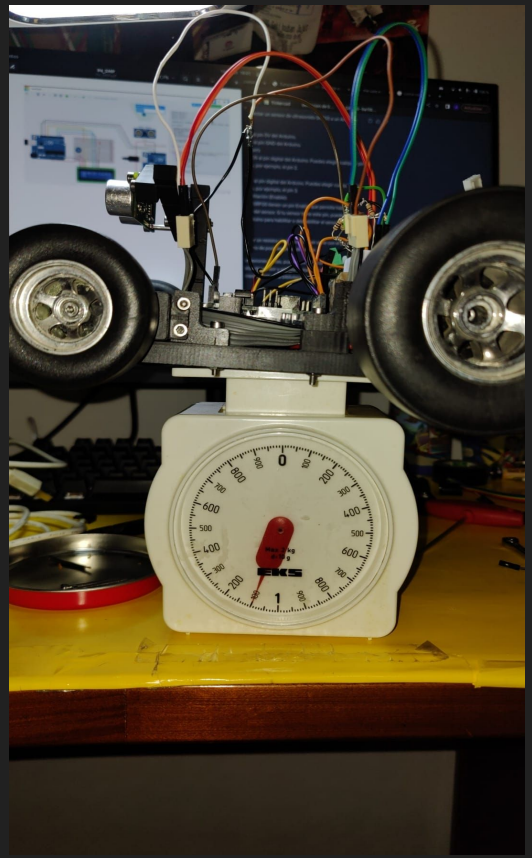
HARDWARE

Incorporación de los infrarrojos y nueva placa de pruebas pequeña



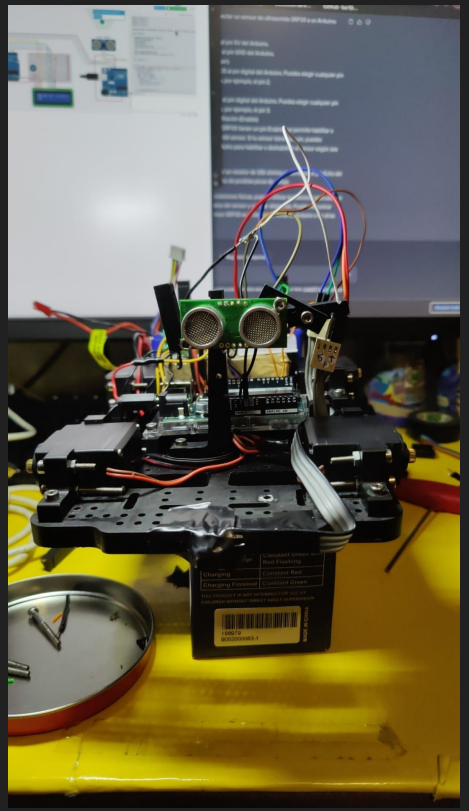
Primer pesaje (sin bulking aún)

HARDWARE

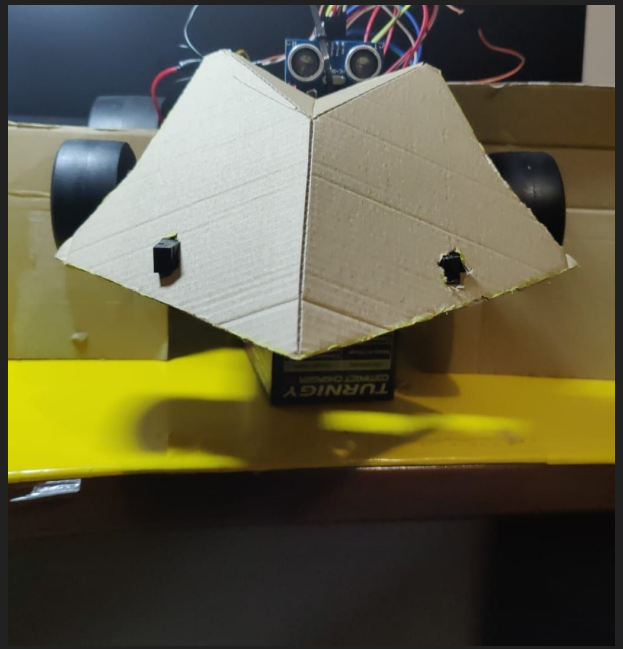


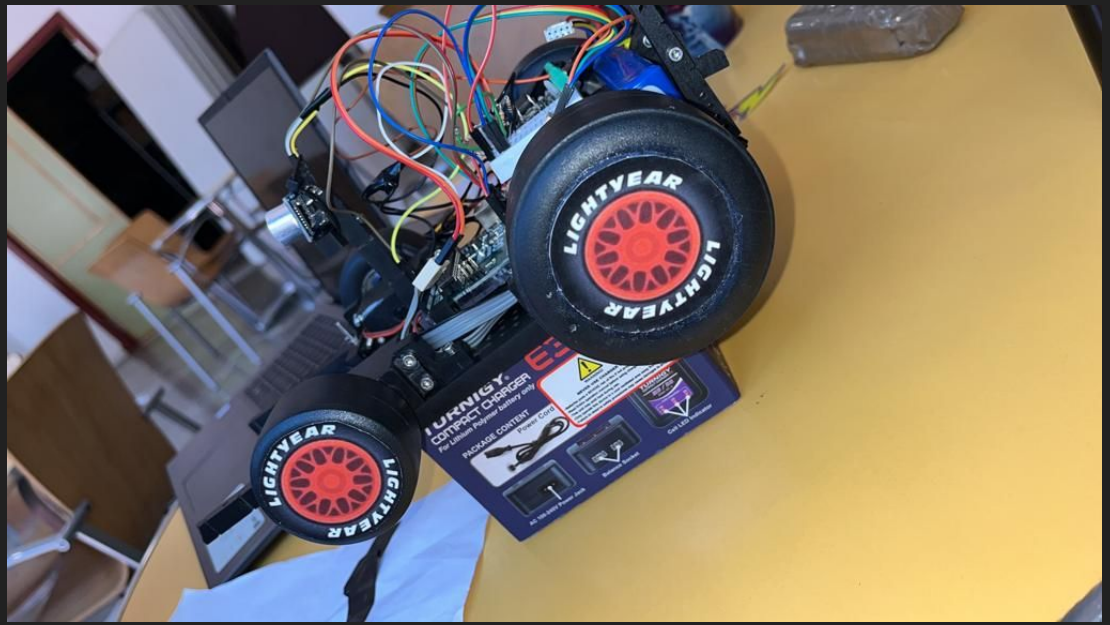
Peso sin chasis exterior

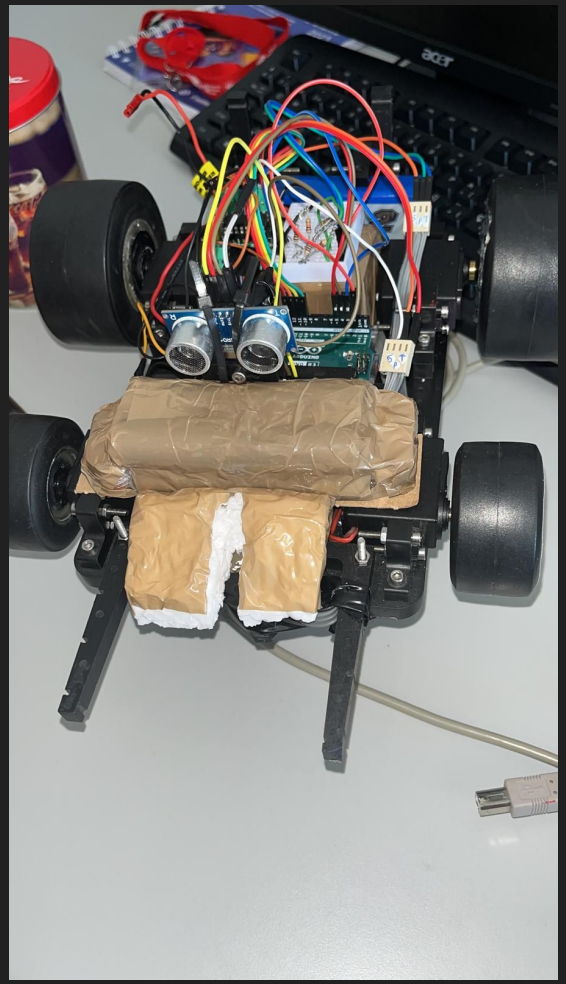
Incorporación del ultrasonidos



HARDWARE




LIGHTYEAR
LIGHTYEAR
LIGHTYEAR
LIGHTYEAR
COMPACT CHARGER
PACKAGE CONTENT




Medicated
Rust-eze
Biker Ointment
Weed Formula

SOFTWARE

Declaración de variables del programa

```
// Infrarrojos
#define TIMEOUT_ECO 1000000 // <anchura máxima del pulso generado por el módulo en µs>
const int irPinDel = 2; // Pin del sensor de infrarrojos delantero
const int irPinTra = 3; // Pin del sensor de infrarrojos trasero
int irValueDel; // Valor del sensor de infrarrojos delantero
int irValueTra; // Valor del sensor de infrarrojos trasero


// Ultrasonidos
long duration; // Duración del pulso de eco
int distance;  // Distancia medida en centímetros
const int trigPin = 4; // Pin de activación del sensor ultrasónico (TRIGGER)
const int echoPin = 5; // Pin de recepción de eco del sensor ultrasónico (ECHO)

// Puente h
const int pwmPin1 = 13; // E1 del puente H conectado al pin 13 del Arduino
const int dirPin1 = 12; // E2 del puente H conectado al pin 12 del Arduino
const int pwmPin2 = 11; // E3 del puente H conectado al pin 11 del Arduino
const int dirPin2 = 10; // E4 del puente H conectado al pin 10 del Arduino

// Variable para el temporizador de 3 segundos
const unsigned long initialDelay = 3000;
```

SOFTWARE

Inicialización del programa

```
6
7   void setup() {
8       // Iniciar comunicación serial
9       Serial.begin(9600);
0
1       // INFRARROJOS
2       pinMode(irPinDel, INPUT);
3       pinMode(irPinTra, INPUT);
4
5       // ULTRASONIDOS
6       //  Inicializar los pines de TRIGGER y ECHO
7       pinMode(trigPin, OUTPUT);
8       pinMode(echoPin, INPUT);
9
0       digitalWrite(trigPin, LOW);
1
2       // Esperar 3 segundos antes de iniciar las acciones
3       delay(initialDelay);
4
5       pinMode(pwmPin1, OUTPUT);
6       pinMode(pwmPin2, OUTPUT);
7       pinMode(dirPin1, OUTPUT);
8       pinMode(dirPin2, OUTPUT);
9
0       parar();
1
2   }
```

SOFTWARE

Función para obtener la distancia a través del ultrasonidos

```
void getDistancia() {
    // Generar un pulso de 10 microsegundos en el pin TRIGGER
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Medir la duración del pulso de eco
    duration = pulseIn(echoPin, HIGH);

    // Calcular la distancia en centímetros
    distance = duration * 0.034 / 2;

    // Imprimir la distancia en el monitor serial
    Serial.print("Distancia: ");
    Serial.print(dist (const char [4])" cm"
    Serial.println(" cm");
}
```

SOFTWARE

Funciones para mover las ruedas en diferentes direcciones

```
// Funcion para girar a la izquierda
void girarIzquierda() {
    digitalWrite(dirPin1, HIGH);
    digitalWrite(pwmPin1, LOW);

    digitalWrite(dirPin2, HIGH);
    digitalWrite(pwmPin2, HIGH);
}

// Funcion para girar a la derecha
void girarDerecha() {
    digitalWrite(dirPin1, HIGH);
    digitalWrite(pwmPin1, HIGH);

    digitalWrite(dirPin2, HIGH);
    digitalWrite(pwmPin2, LOW);
}

// Función para girar a la derecha marcha atrás
void girarDerechaMarchaAtras() {
    digitalWrite(dirPin1, LOW);
    digitalWrite(pwmPin1, HIGH);

    digitalWrite(dirPin2, HIGH);
    digitalWrite(pwmPin2, LOW);
}

// Función para girar a la izquierda marcha atrás
void girarIzquierdaMarchaAtras() {
    digitalWrite(dirPin1, HIGH);
    digitalWrite(pwmPin1, LOW);

    digitalWrite(dirPin2, LOW);
    digitalWrite(pwmPin2, HIGH);
}
```

```
// Función para mover hacia atrás
void haciaAtras() {
    digitalWrite(dirPin1, LOW);
    digitalWrite(pwmPin1, HIGH);

    digitalWrite(dirPin2, LOW);
    digitalWrite(pwmPin2, HIGH);
}

// Función para detener el movimiento
void parar() {
    digitalWrite(dirPin1, LOW);
    digitalWrite(pwmPin1, LOW);

    digitalWrite(dirPin2, LOW);
    digitalWrite(pwmPin2, LOW);
}

// Funcion para avanzar hacia delante
void haciaDelante() {
    digitalWrite(dirPin1, HIGH);
    digitalWrite(pwmPin1, LOW);

    digitalWrite(dirPin2, HIGH);
    digitalWrite(pwmPin2, LOW);
}
```

```
void loop()
{
    // Actualizar el valor de infrarrojos
    irValueDel = digitalRead(irPinDel);
    irValueTra = digitalRead(irPinTra);

    vecesNegro = 0;

    // Si el robot detecta la línea negra por el sensor delantero, realizar maniobra para evitar salirse
    while (irValueDel == 0)
    {
        vecesNegro++;
        if (vecesNegro > 5)
        {
            haciaAtras();
            delay(500);

            // Elegir aleatoriamente la dirección de giro para no salirse del tatami
            if (random(2) == 0)
            {
                girarIzquierdaMarchaAtras();
            }
            else
            {
                girarDerechaMarchaAtras();
            }

            delay(1000);
        }

        irValueDel = digitalRead(irPinDel);
    }

    if (irValueDel == 1)
    { // Si detecta que estamos en lo blanco, avanzar hacia delante
        haciaDelante();
    }

    // Si el sensor trasero detecta la línea negra, realizar maniobra para escapar de la situación
    if (irValueTra == 0)
    {
        girarDerecha();
        delay(500);
        haciaDelante();
        delay(1000);
    }

    getDistancia();

    while (distance < 30)
    {
        // Si detecta al enemigo muy cerca y el sensor infrarrojos trasero ha detectado la línea negra, hacer un movimiento para escapar
        if (distance < 17 && irValueTra == 0)
        {
            girarDerechaMarchaAtras();
            delay(1000);
            haciaDelante();
            delay(1000);
        }
        else
        {
            haciaDelante();
            delay(100);
            getDistancia();
        }
    }


}
```

PROBLEMAS

Los infrarrojos dan muchos problemas por los cables. A veces hacen contacto entre sí y fallan. Tienen que estar en una posición fija

Tuvimos problemas con el controlador del puente h un par de veces, que hacía que mandase mucha más corriente a la placa Arduino y salía humo por cortocircuito

Las ruedas no estaban bien unidas al servo y los cambiamos por unas piezas nuevas compradas en una tienda