



Tecnológico Nacional de México
Instituto Tecnológico de La Laguna



Sistemas Programables
INGENIERÍA EN SISTEMAS COMPUTACIONALES

Proyecto Final

Presenta:

18131278 - Iván Romero Canaán

18131273 - José Ángel Rocha García

18131252 - Jesús Daniel López Hernández

18131222 - José Luis Carreón Reyes

18131210 - Jesús Francisco Aguilera Guajardo

Torreón, Coahuila
26 de mayo del 2022

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
Descripción general del proyecto	1
Descripción del problema	2
Objetivos	2
Principal	2
Secundarios	3
Justificación	3
Alcances y limitaciones	3
COMPETENCIA A DESARROLLAR	4
METODOLOGÍA	4
Metodología de construcción	4
Diagrama del receptor (carro)	4
Diagrama del control de inclinación	5
Lista y descripción del material y equipo empleados	15
Costos (gastos que se realizaron)	19
Desarrollo del proyecto	20
Explicación del código del receptor	20
Explicación del código de la antena	23
Explicación del código esp32	25
Agregamos cambios al código del Arduino en el carro	30
Creando la app	32
PROBLEMAS DURANTE EL DESARROLLO DEL PROYECTO	34
RESULTADOS	35
CONCLUSIONES Y RECOMENDACIONES	36
REFERENCIAS	37

INTRODUCCIÓN

Descripción general del proyecto

Este proyecto consiste en un carro a control remoto, el cual es manejado por gestos de inclinación; es decir, dependiendo de la dirección de inclinación de dicho control (el cuál será manipulado de manera similar a un volante) será la dirección en la que el carro se dirija.

Este proyecto se conforma de 2 componentes principales, que son el carro y el guante de control remoto. Ambos están construidos con materiales electrónicos y están codificados para que se puedan comunicar entre sí para su funcionamiento esperado. Aparte de esto, el proyecto también cuenta con una aplicación móvil en la que podemos controlarlo por Wifi como un control remoto convencional. Además de todo esto, el proyecto también cuenta con su propia página web en donde se hará la simulación de promoción de dicho producto y el espacio para poder conseguir este mismo. En esta página se incluyen fotos, videos y descripciones para convencer al usuario de adquirir o desarrollarlo por su propia cuenta.

Teniendo en cuenta los materiales que necesitamos comprar, y los materiales que ya teníamos, el costo total del proyecto fue de: \$ 2419 MXN.

Descripción del problema

Actualmente no es difícil encontrar kits de robótica que permitan construir un carro desde cero, pero por lo general el control del carro se da por medio de sensores ya sea de proximidad o de seguimiento de una línea y aquellos que cuentan con control remoto no permiten realizar inclinaciones para control el carrito, también cuentan con el problema de que son prediseñados para ciertos componentes por lo cual el espacio para expandir capacidades es muy limitado o en todo caso nulo. En cuanto al código también es difícil modificar puesto que los componentes que se conectan ya sea arduino o el dispositivo que utilicen como procesador no permiten la modificación de sus parámetros, sólo permiten la lectura por lo cual el aprendizaje comparado contra realizar el carro desde cero a comprar un kit es muy grande resultando ganador el carro creado por propia mano.

Objetivos

Principal

- Poner a prueba los conocimientos adquiridos de las prácticas realizadas a lo largo de las 5 unidades de la materia.

Secundarios

- Realización del carro.
- Realización del volante a control remoto.
- Lograr la comunicación entre carro y control.
- Que el carro pueda ser controlado con el volante.
- Realizar la aplicación móvil con funcionalidades extra.
- Realizar la página web con información del proyecto realizado.
- Haber obtenido los conocimientos esperados dentro de la materia.

Justificación

Se escogió realizar este proyecto debido a que cumple en su mayoría de los requisitos establecidos para entregar como proyecto final, el cual es el uso de sensores y actuadores. Además, al ser un proyecto sencillo y popular, tuvimos bastantes fuentes de información disponibles en las cuales basarnos y poder realizar nuestra propia versión a la medida de nuestros gustos, capacidades y juicio. En adición, gracias a todo el soporte que hay en las fuentes de información, nos fue posible terminar el proyecto en un tiempo prudente y conveniente para poder realizar las otras partes del proyecto, las cuales serían la realización de la aplicación móvil y la página web.

Al final, hemos puesto a prueba y reforzado los conocimientos adquiridos en clase mediante prácticas realizando este proyecto, por lo que concluimos que ésta fue una buena elección como proyecto final de la materia.

Alcances y limitaciones

Nuestro coche a control remoto es un dispositivo sencillo el cual se mueve hacia adelante, hacia atrás y a los lados. No posee la capacidad de hacer maniobras avanzadas como piruetas como otros coches prearmados que están disponibles en el mercado.

El rango del control del coche se encuentra en el rango de los 5 metros cuando éste está alimentado por el Arduino, y el rango se extiende cuando se le provee de

energía externa hasta 1000 metros. En este aspecto, nuestro coche no se queda muy atrás en comparación con otros productos similares.

COMPETENCIA A DESARROLLAR

- Elaborar un proyecto en el que presenta soluciones de interfaces hombre-máquina o máquina-máquina aplicando microcontroladores, dispositivos de E/S e interfaces de comunicación de datos en un sistema programable.

METODOLOGÍA

Metodología de construcción

Diagrama del receptor (carro)

Este circuito se compone de un arduino que se encarga de procesar toda la información que reciba mediante el nrf24l01, después enviará señales al L298N motor driver para activar el motor correspondiente con su sentido de giro hacia adelante o atrás, las baterías alimentan el driver y el arduino.

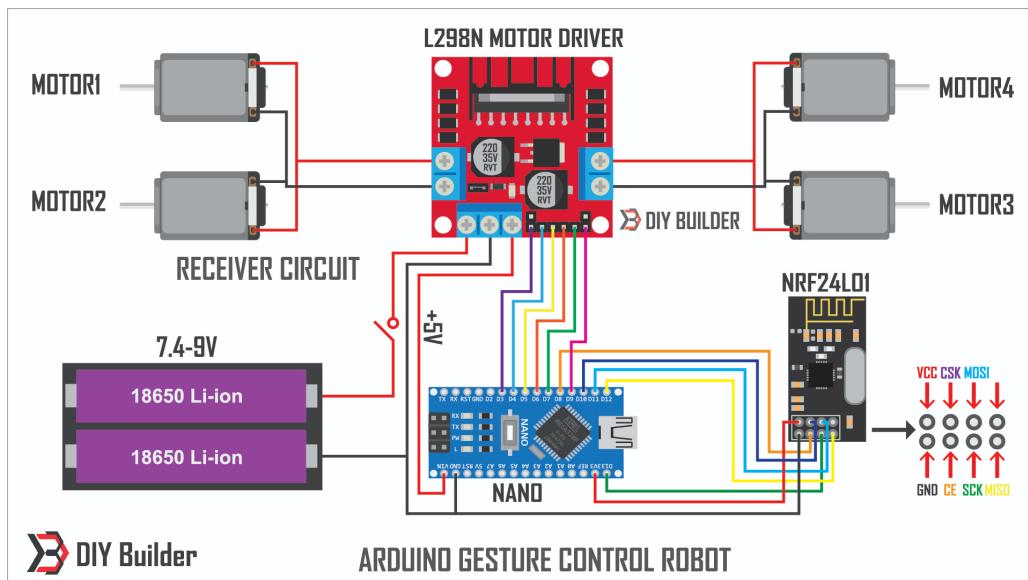
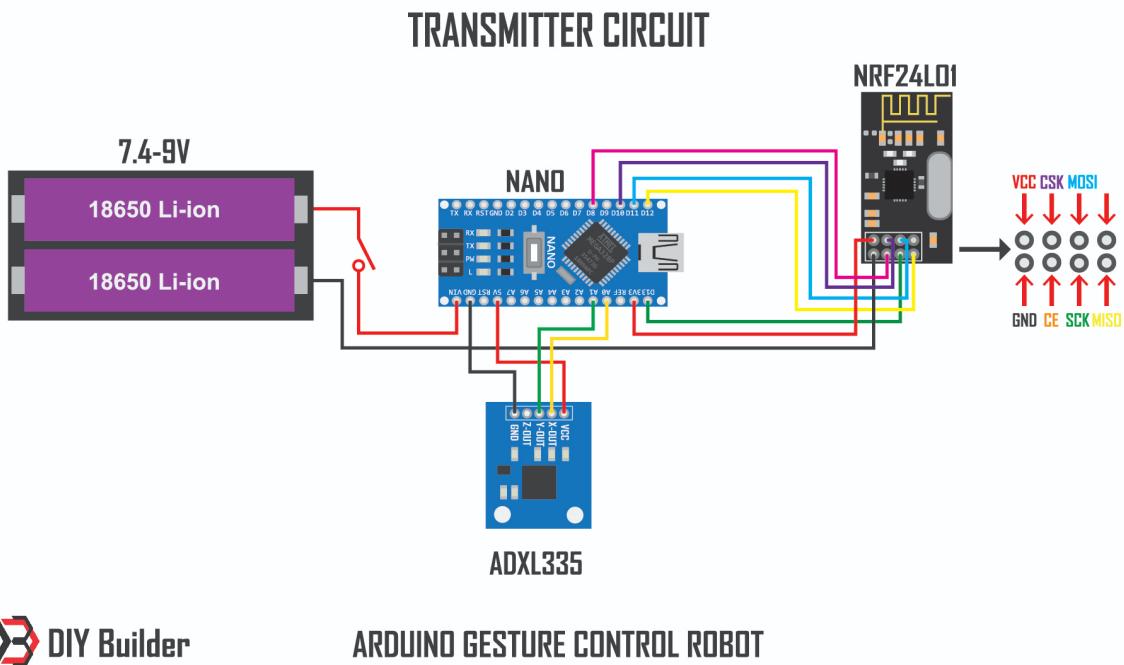


Diagrama del control de inclinación

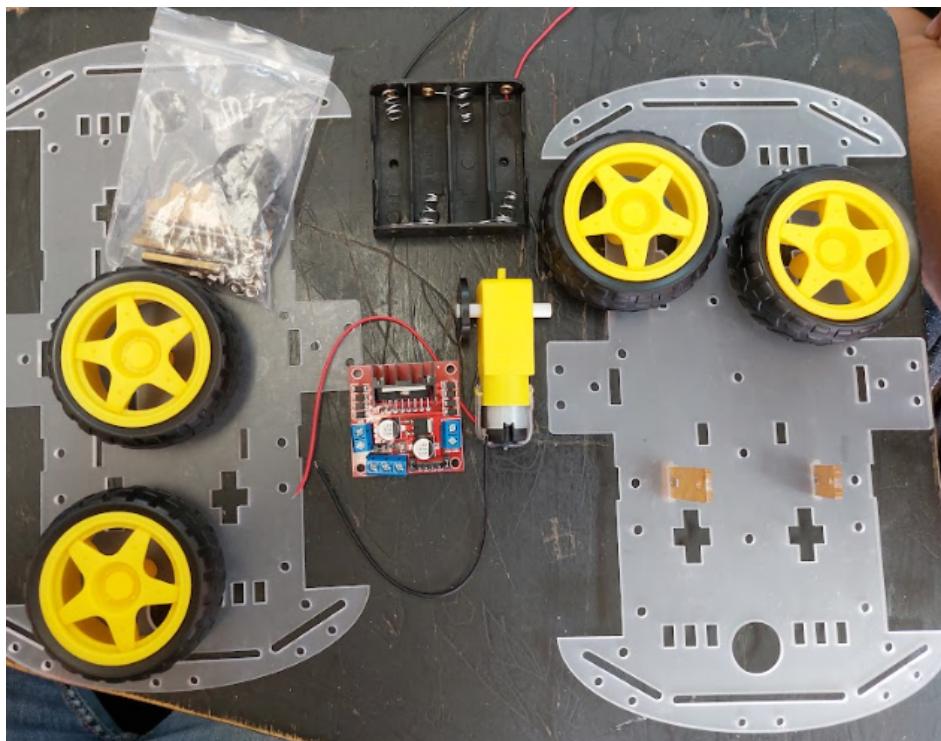
Al igual que el diagrama anterior, el arduino se encarga de leer las señales, en este caso leerá el acelerómetro Adxl335 para después enviar información por medio del nrf24l01, las baterías alimentan el arduino y este a su vez alimenta al adxl335 y la nrf24l01.



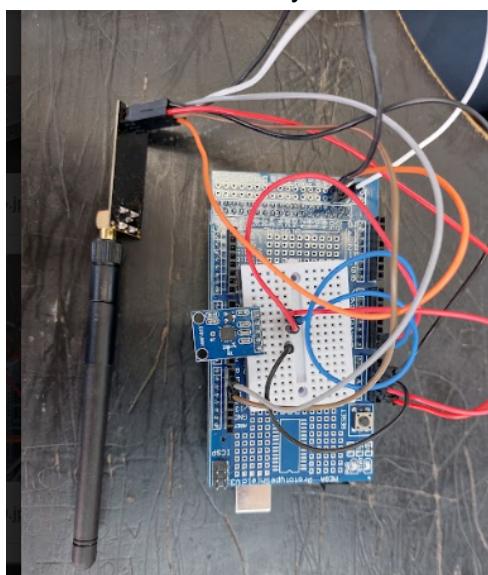
Los componentes mostrados en los diagramas anteriores cambiaron durante la construcción del proyecto para ahorro de materiales (baterías, versiones de arduino).

Componentes usados

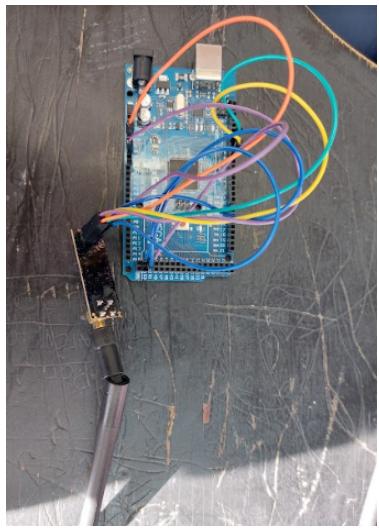
Base del carro con motores, driver y llantas



Componentes del controlador
Antena, acelerómetro y arduino



Arduino en el receptor (carro) y su antena, el driver se probó por separado

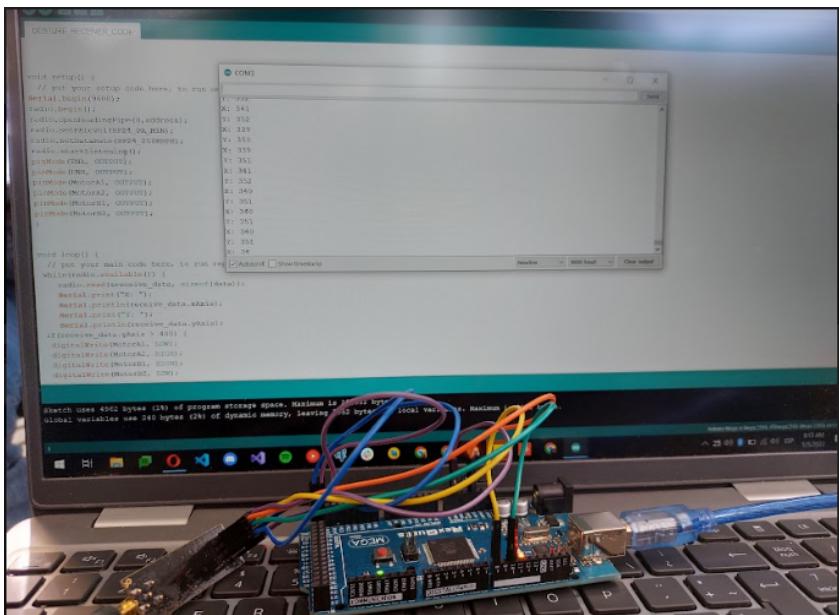


Proceso de construcción

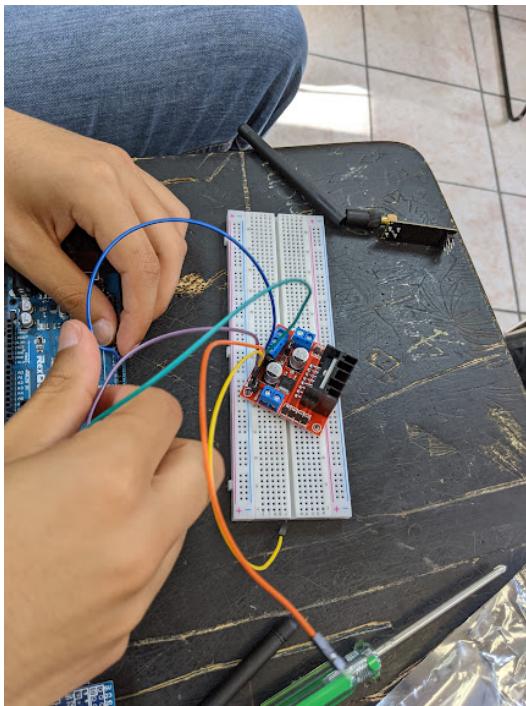
Para comenzar montamos los componentes sin una base o en protoboard para comprobar el funcionamiento de cada uno de ellos.

Controlador de motores y antena

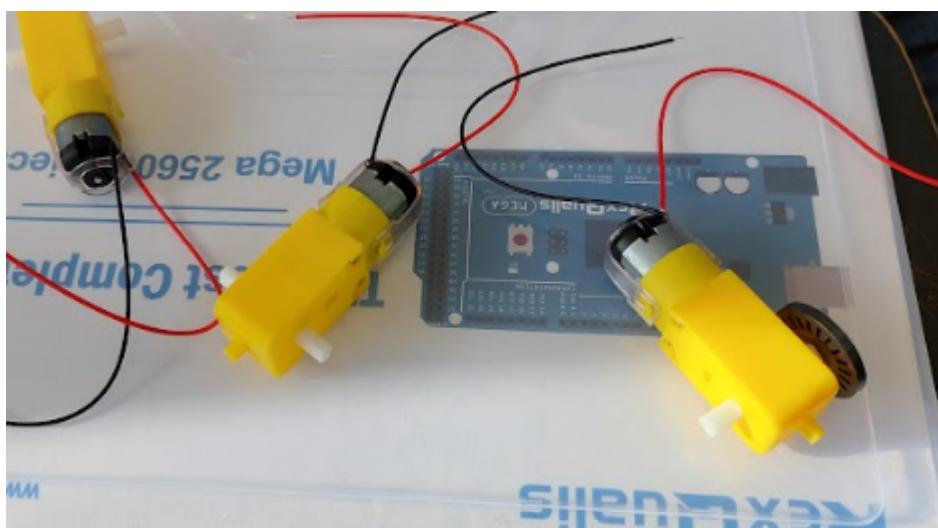
Se probó que el driver activará los puertos donde están conectados los motores además de comprobar que la antena recibe los datos del control de inclinación para asegurarnos de montar, la velocidad a la que lee los datos es demasiado rápida por lo cual pueden existir algunos problemas a la hora de enviar señales a los motores por lo cual decidimos agregar un pequeño retraso, la antena fue cambiada de pines debido a que la versión de arduino mega usa diferentes puertos para enviar señales en comparación a la versión nano o normal.



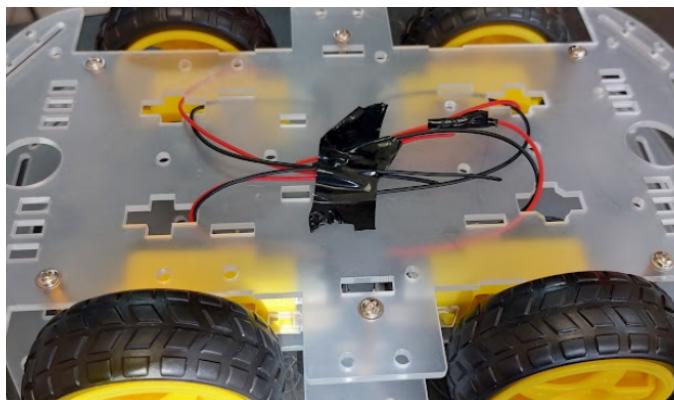
El driver activaba los puertos según el cable que tenía corriente, no fue necesario conectar los motores para verificar la entrada ya que es posible ver la corriente pasar conectando un multímetro a los puertos en cada lado.



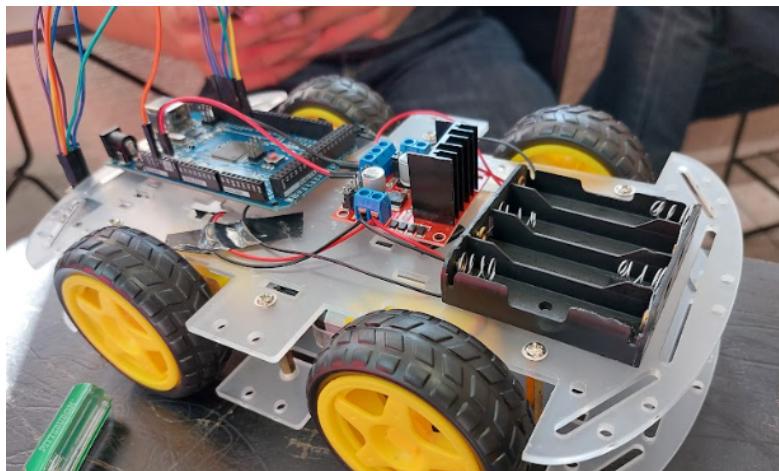
Después se procedió a soldar cables en los motores para montarlos sobre la base de acrílico, este proceso fue muy rápido debido a que los motores ya cuentan con un orificio por donde es posible pasar el cable y asegurarla con soldadura.



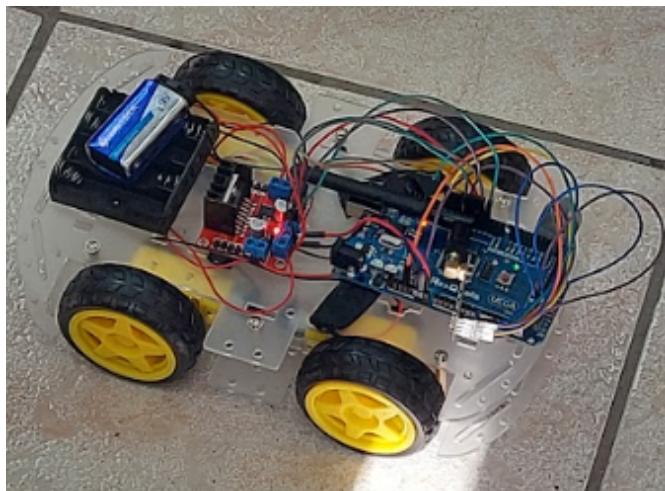
Se montaron los motores en la base de acrílico y se colocaron las ruedas en los motores, la base es resistente por lo cual aguantara el peso del arduino con el driver y algunos otros componentes de ser necesario, para evitar que los cables se perdieran dentro del chasis se aseguran con cinta adhesiva.



Se montaron el arduino y la antena (colocada por debajo de la base) y el porta pilas (que después cambió a uno de 9v debido a la falta de corriente de las pilas AA), al probar el arduino tuvimos el problema de que los tornillos del driver en ocasiones no sujetan bien el cable pero se soluciono dando un poco más de fuerza con el desarmador.

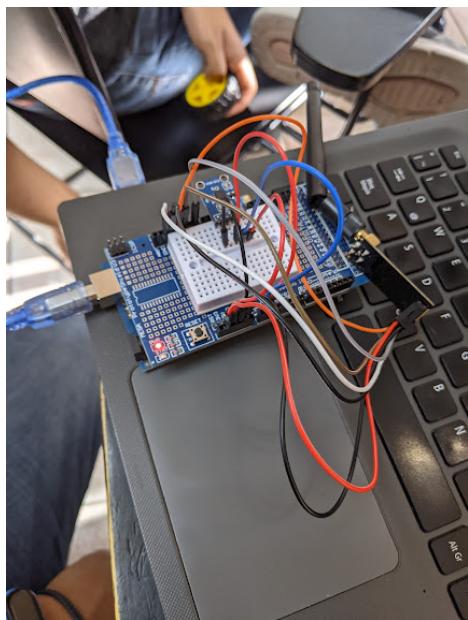


Esta fue la primera versión del carro aun con los cables desordenados



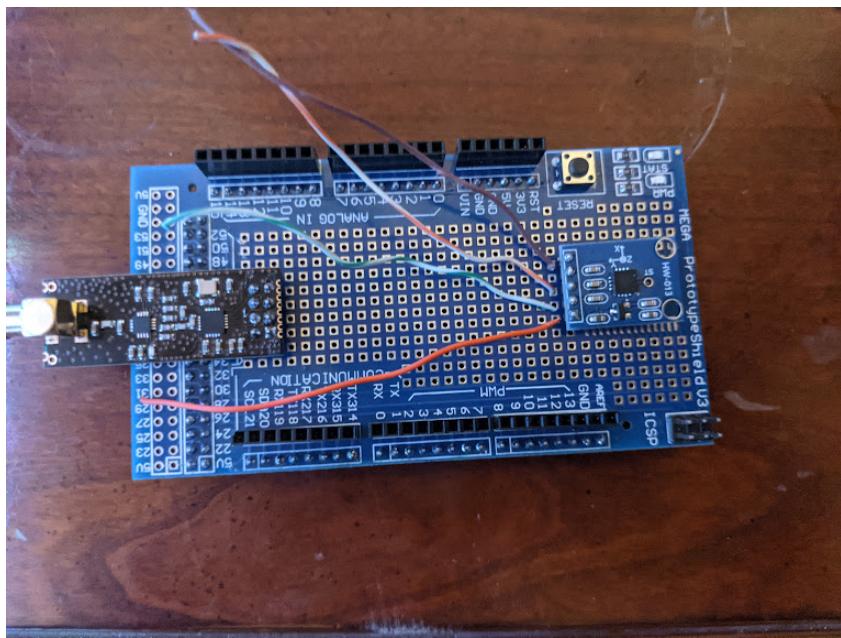
Control de mano (acelerómetro y antena)

Se comprobó que los ejes del acelerómetro detectaron correctamente el movimiento que hacíamos y que la antena para después soldar los componentes en la shield de expansión

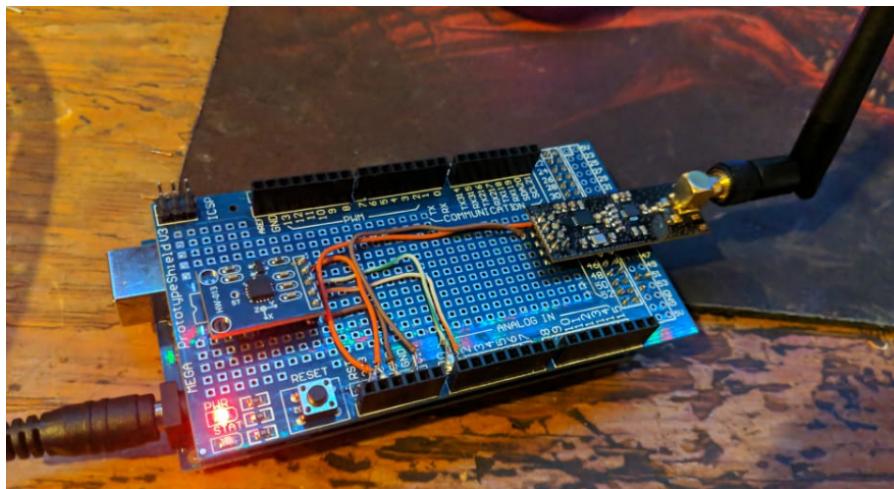


Usamos una expansión proto shield para montar los componentes y después soldarlos ya que es mucho más fácil soldar sobre ella

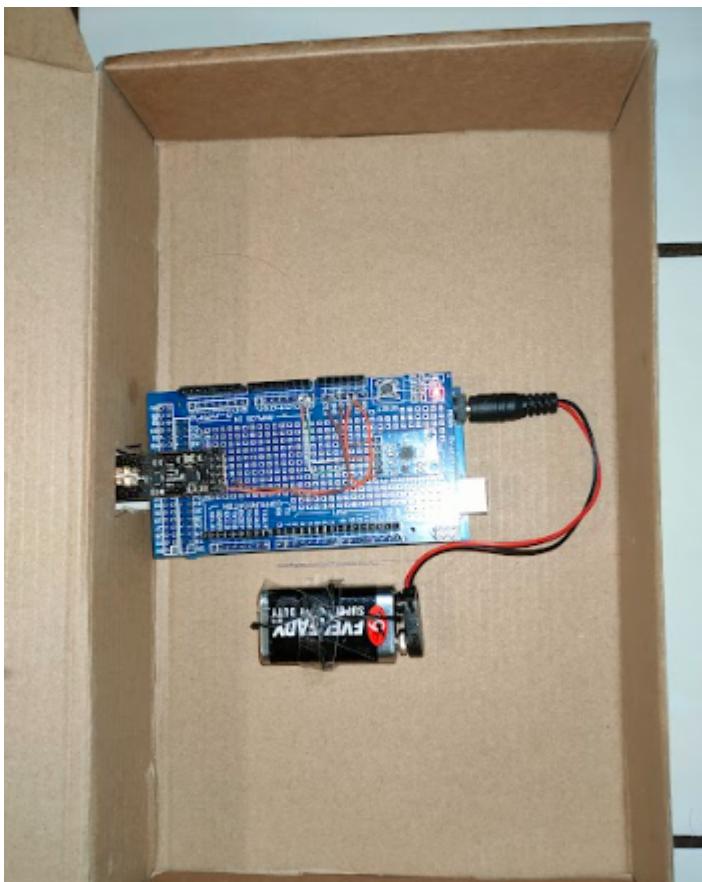




Esta es la versión final , es posible usarla tomándola con la mano e inclinando pero debido a los pines expuestos se puede provocar un corto en alguno de los cables o componentes

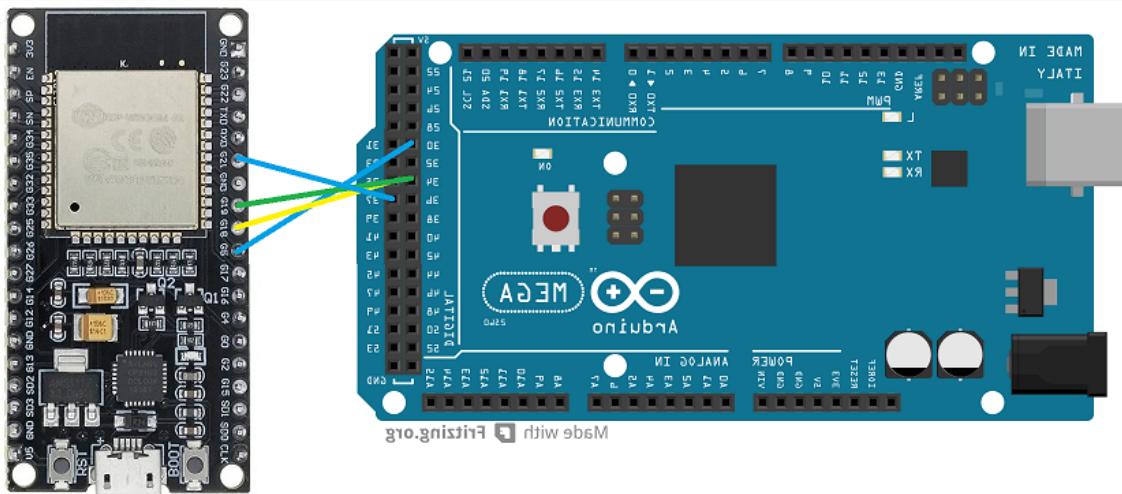


Despues se metio a una caja para proteger el circuito y tener mejor agarre, se aseguró la batería y se hizo un agujero en la parte del lomo para poder pasar la antena

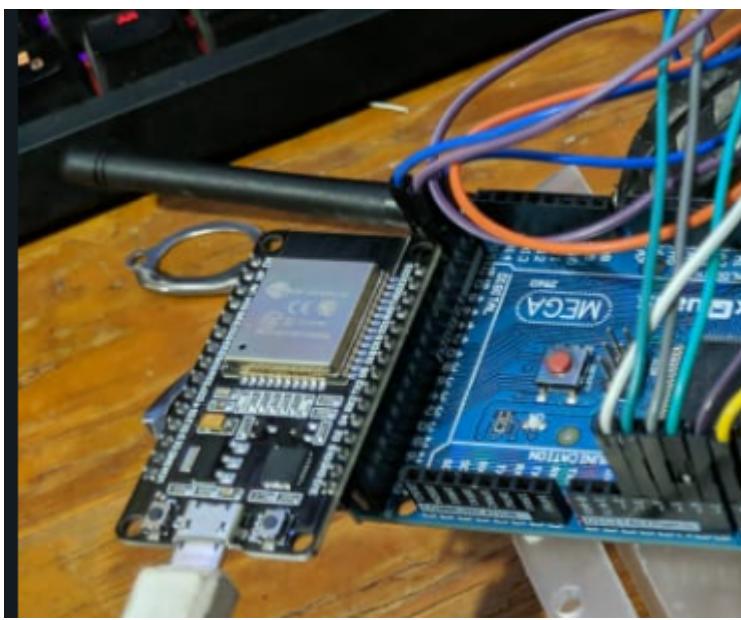


Agregando el esp32

Para poder controlar el carro desde el celular fue necesario agregar otro microcontrolador (esp32) con wifi y bluetooth para realizar una aplicación, el diagrama para comunicar el esp32 con arduino fue el siguiente



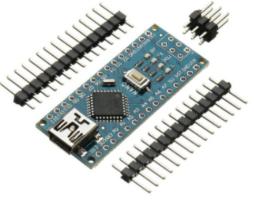
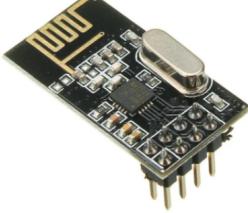
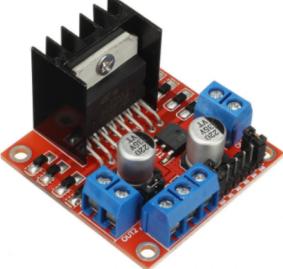
el esp 32 no utilizó cables como tal ya que es posible montarlo directamente en el arduino mega

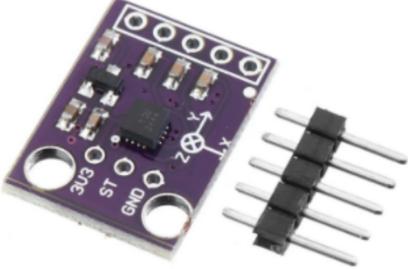


El funcionamiento del esp 32 se limita a recibir las señales que el celular manda vía wifi para después transmitir un HIGH o LOW hacia los pines de entrada del arduino y mediante código activar los diferentes motores con los que contamos, así mismo

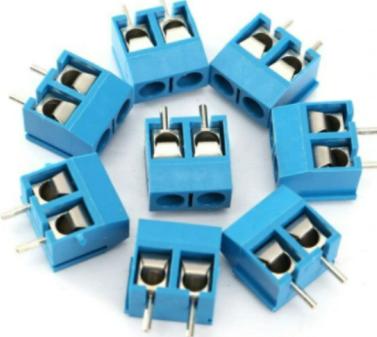
cuenta con una página web integrada en el dispositivo el cual realiza cálculos de las revoluciones por minuto que es capaz de generar el dispositivo.

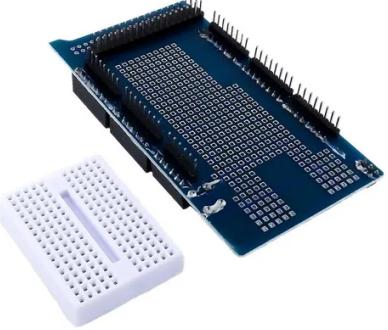
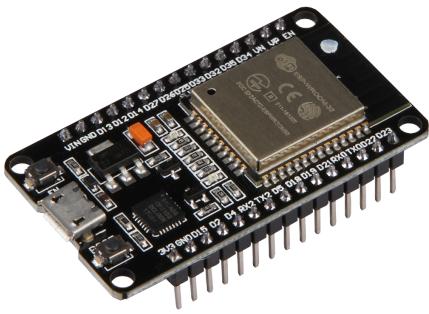
Lista y descripción del material y equipo empleados

	Arduino nano o mega (2x) Es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra.
	NRF24L01+ RF Module (2x) Módulo de transmisión, indispensable para la comunicación entre el guante y el carro.
	L298N Motor Driver El módulo controlador de motores L298N H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla, gracias a los 2 los dos H-bridge que monta.

	<p>ADXL335 Module</p> <p>Este módulo tiene un acelerómetro ADXL335 de 3 ejes, incluye condensadores de desacoplo para un óptimo funcionamiento. Está listo para su uso en cualquier tipo de proyecto con microcontroladores o para su uso en robótica.</p> <p>Para cada eje devuelve una tensión proporcional a la aceleración, las cuales se pueden convertir a datos digitales con la ayuda de algún microcontrolador PIC, Atmel o Arduino y sus entradas analógicas. Se recomienda implementar algún filtro digital para tener una lectura más estable.</p>
	<p>TT Gear Motor (4x)</p> <p>Es una fuente de poder con engranajes de plástico que se puede utilizar para mover las ruedas del carro. Este motor engranado TT encaja perfectamente con la polea de sincronización de plástico 622 y con la polea de sincronización de plástico 90T para el sistema de ruedas.</p>

	<p>Rubber wheels (4x)</p> <p>Estas irán en los motores TT para poder mover el carro en la dirección deseada</p>
	<p>18650 li-ion battery (2x) o batería de 9V (2x)</p> <p>Para darle energía al carro.</p>
	<p>18650 battery holder</p> <p>Aquí es donde irán las baterías para poder transferir la energía de estas al carro.</p>

	<p>Screw Terminal</p> <p>Para conectar cables y apretarlos.</p>
	<p>Jumpers</p> <p>Para realizar las conexiones entre los puertos de los componentes.</p>
	<p>Chasis</p> <p>Soporte para los componentes y las conexiones entre ellos.</p>

	<p>Shield de expansión para arduino mega</p> <p>Esta placa permite montar componentes y conectarse al arduino</p>
	<p>ESP32</p> <p>El esp32 es un dispositivo programable para proyectos que cuenta con wifi y bluetooth integrados</p>

Costos (gastos que se realizaron)

Componente(s)	Precio
Arduino mega (2x)	\$1178
Esp32	\$209
Motor cc con llanta (4x)	Incluidos con el chasis de carro
Chasis de carro	\$290
Llantas para motor cc (4x)	Incluidas con el chasis de carro
Antena nrf24l01 (2x)	\$182
Adxl335 (acelerometro)	\$109
L298N motor driver	\$75
Batería (2x)	\$200

Shield de expansión para arduino mega	\$76
Materiales extra (cautin, cables, soldadura, madera)	\$100

Desarrollo del proyecto

Explicación del código del receptor

Se incluyen las librerías necesarias para el funcionamiento de la antena, estas fueron descargadas desde el administrador de librerías del IDE Arduino.

```
#include<SPI.h>
#include<nRF24L01.h>
#include<RF24.h>
```

Se declaran las salidas en el arduino, las salidas ENA, ENB sirven para que el driver de motor nos permita manipular los motores, si estos pines no son activados los motores no girarán, las siguientes entradas A1,A2,B1,B2 son los motores disponibles.

```
int ENA = 3;
int ENB = 9;
int MotorA1 = 4;
int MotorA2 = 5;
int MotorB1 = 6;
int MotorB2 = 7;
```

Declaración de los pines que usará la antena rf para transmitir, además de declarar la estructura y objeto para enviar información, la antena se comunica mediante un arreglo de 5 posiciones

```

RF24 radio(8, 10);

const byte address[6] = "00001";

struct data {
    int xAxis;
    int yAxis;

};

data receive_data;

```

Inicialización de los modos de los pines, seriales, opciones de la antena para transmitir en el canal deseado, el pin serial se utilizó únicamente durante las pruebas ya que en la versión final no se utilizan conexiones a computadora, la antena se comunicará transmitiendo datos a 250KBPS

```

Serial.begin(9600);
radio.begin();
radio.openReadingPipe(0,address);
radio.setPALevel(RF24_PA_MIN);
radio.setDataRate(RF24_250KBPS);
radio.startListening();
pinMode(ENA, OUTPUT);
pinMode(ENB, OUTPUT);
pinMode(MotorA1, OUTPUT);
pinMode(MotorA2, OUTPUT);
pinMode(MotorB1, OUTPUT);
pinMode(MotorB2, OUTPUT);

```

Se crea en el loop el ciclo while para comprobar que la antena está funcionando y disponible, se imprime un mensaje para informar acerca del control y se leen los datos que la antena recibe

```

//Pausa entre lecturas, ~1 ms
while(radio.available()) {
    Serial.println("Operando con el control manual");
    radio.read(&receive_data, sizeof(data));
}

```

Se comprueban los datos recibidos en los ejes

Si la inclinación en el eje y es mayor a 400 el carro avanzara hacia adelante activando los pines en el driver que hacen que se genere un giro en sentido horario y poniendo los ena, enb en 150 para mover los motores

```

analogWrite(ENB, 150);
}
if(receive_data.yAxis > 400) {
    digitalWrite(MotorA1, LOW);
    digitalWrite(MotorA2, HIGH);
    digitalWrite(MotorB1, HIGH);
    digitalWrite(MotorB2, LOW);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);
}

```

Si es menor a 320 avanzará hacia atrás y repetirá el proceso de activar los pines para generar un giro en este caso en sentido contrario, al igual que lo que hizo con los pines ena y enb en el proceso anterior.

```

} else if(receive_data.yAxis < 320) {
    digitalWrite(MotorA1, HIGH);
    digitalWrite(MotorA2, LOW);
    digitalWrite(MotorB1, LOW);
    digitalWrite(MotorB2, HIGH);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);
}

```

Si el eje x es mayor a 400, hará un giro a la derecha, en este caso las llantas giraran en sentido contrario a las del lado opuesto lo cual provoca que el carro gire en su propio eje.

```

analogWrite(ENB, 150);
} else if(receive_data.xAxis > 400) {
    digitalWrite(MotorA1, HIGH);
    digitalWrite(MotorA2, LOW);
    digitalWrite(MotorB1, HIGH);
    digitalWrite(MotorB2, LOW);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);
}

```

Si el eje x es menor a 320 hará un giro hacia la izquierda. este proceso es similar al anterior.

```

analogWrite(ENB, 150);
else if(receive_data.xAxis < 320) {
    digitalWrite(MotorA1, LOW);
    digitalWrite(MotorA2, HIGH);
    digitalWrite(MotorB1, LOW);
    digitalWrite(MotorB2, HIGH);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);
}

```

Si ninguno de estos datos se cumple entonces significa que el carro esta detenido por lo cual se apagan todos los motores, los pines `ena`, `enb` se apagan para evitar que los motores se muevan.

```
else {
    digitalWrite(MotorA1, LOW);
    digitalWrite(MotorA2, LOW);
    digitalWrite(MotorB1, LOW);
    digitalWrite(MotorB2, LOW);
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
}
```

Explicación del código de la antena

Se incluyen las librerías para hacer funcionar, estas librerías se instalaron desde el ide de arduino en la parte de administración de librerías

```
#include<SPI.h>
#include<nRF24L01.h>
#include<RF24.h>
```

Se declaran los pines de entrada para el acelerómetro, el sensor emite valores análogos por lo cual se asignan a las entradas A0 y A1

```
const int x_out = A0;
const int y_out = A1;
```

Se declara la estructura de los datos igual al del receptor y se declara objeto

```
struct data{
    int xAxis;
    int yAxis;

};

data send_data;
```

Se declaran los pines que utiliza la antena rf para comunicarse y la dirección a la que enviará los datos, esta dirección es un arreglo de bytes de 5 dígitos de longitud

```
RF24 radio(8,10);
const byte address[6] = "00001";
```

En el setup declaramos el funcionamiento de la antena (canal de radiofrecuencia).

Aquí no es necesario establecer el modo de pin ya que por defecto toma los pines analógicos como entradas.

En el setup la antena transmite a la misma frecuencia que la del carro para evitar problemas al transmitir-leer, de igual forma el serial se utiliza únicamente para pruebas

```
void setup() {

    // put your setup code here, to run once:
Serial.begin(9600);
radio.begin();
radio.openWritingPipe(address);
radio.setPALevel(RF24_PA_MIN);
radio.setDataRate(RF24_250KBPS);
radio.stopListening();
}
```

En el loop se leen los los valores analogicos desde el las entradas A0 y A1, estos valores se asignan a las variables de la estructura send_data en su eje correspondiente y después son enviados a través de la antena, tambien imprimimos su valor mediante el puerto serial.

```
void loop() {
    // put your main code here, to run repeatedly:
send_data.xAxis = analogRead(x_out);
send_data.yAxis = analogRead(y_out);

radio.write(&send_data, sizeof(data));
Serial.println("x: ");
Serial.println(send_data.xAxis);
Serial.println("y: ");
Serial.println(analogRead(send_data.yAxis));
delay(500);

}
```

Explicación del código esp32

Inclusión de la librería para wifi

```
| #include <WiFi.h>
```

Configuraciones para la red wifi que se utiliza para comunicación , (direcciones ip, puerta, dns), estas configuraciones son estáticas y deben modificarse si se cambia de red wifi.

```
const char* ssid = "realme 7 Pro";
const char* password = "1234567890";

// Configuración de la IP estática.
IPAddress local_IP(192, 168, 236, 12);
IPAddress gateway(192, 168, 236, 123);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8); //opcional
IPAddress secondaryDNS(8, 8, 4, 4); //opcional
```

Definición de los pines utilizados, estos pines envían una señal para hacer que el carro se mueva o cambie de tipo de control.

```
define ADL 5
define IZQ 18
define DER 19
define ATR 21
define VEL 22
define CTRL 2
```

Variables para transmisión pwm, el esp32 no puede transmitir directamente la señal pwm por lo que necesita usar 4 variables para generar una de ellas,a diferencia de arduino que solo necesita la función digitalRead().

```
const int canalAnalogo = 0;
const int limiteBits = 8;
const int frecuencia = 5000;
int velocidad = 150;
```

Inicialización del puerto del servidor wifi para la página mostrada.

```
WiFiServer server(80);
```

Establecimiento del puerto serial y modos de pin de las salidas usadas, todo será usado como salida puesto que el esp32 no lee ningún dato.

```
void setup() {
    Serial.begin(115200);
    pinMode(ADL, OUTPUT);
    pinMode(IZQ, OUTPUT);
    pinMode(DER, OUTPUT);
    pinMode(ATR, OUTPUT);
    pinMode(VEL, OUTPUT);
    // ...
}
```

Código para conectar a la red wifi configurada previamente, mostraremos un mensaje de acuerdo a si se logró conectar al wifi o no.

```
// Establecimiento de la IP estática.
if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("Fallo en la configuración.");
}

// Conecta a la red wifi.
Serial.println();
Serial.print("Conectando con ");
Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("Conectado con WiFi.");
```

Inicio del servidor web, también se muestra la ip que se utilizará para conectarse, se muestra un mensaje informando que el servidor fue iniciado de forma correcta e indicamos la dirección ip que usa el esp32.

```

// Inicio del Servidor web.
server.begin();
Serial.println("Servidor web iniciado.");

// Esta es la IP
Serial.print("Esta es la IP para conectar: ");
Serial.print("http://");
Serial.println(WiFi.localIP());

```

En el loop lo primero que se hace es intentar conectarse a la red wifi configurada y mostrar los clientes que se conecten a ella los cuales pueden enviar y dejar de enviar datos.

```

void loop() {
    // Consulta si se ha conectado algún cliente.
    WiFiClient client = server.available();
    if (!client) {
        return;
    }

    Serial.print("Nuevo cliente: ");
    Serial.println(client.remoteIP());

    // Espera hasta que el cliente envíe datos.
    while(!client.available()){ delay(1); }

    /////////////////////////////////
    // Lee la información enviada por el cliente.
    String req = client.readStringUntil('\r');
    Serial.println(req);
}

```

Después se configura la página web para mostrar el estado del carro en RPM, tomando la lectura de la app, la página se genera mediante strings que contienen el código html.

```

////////// Página WEB. ///////////
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;}");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
client.println(".button2 {background-color: #555555;}</style></head>");

client.println("<body>");
client.println("<h1>ESTADO DEL CARRO</h1>");
client.println("<h2>Calculo de velocidad</h2>");

//Leer la velocidad del slider
if(req.indexOf("valor") >= 0){
    velocidad = req.substring(req.indexOf("/") + 6, req.indexOf("/") + 9).toInt();
}

```

Se realiza el cálculo de las rpm con la información disponible de los motores para hacer una regla de 3 ya que el valor que se lee varía desde 0 al 255, mientras que las revoluciones tienen un valor entre 0 a 200.

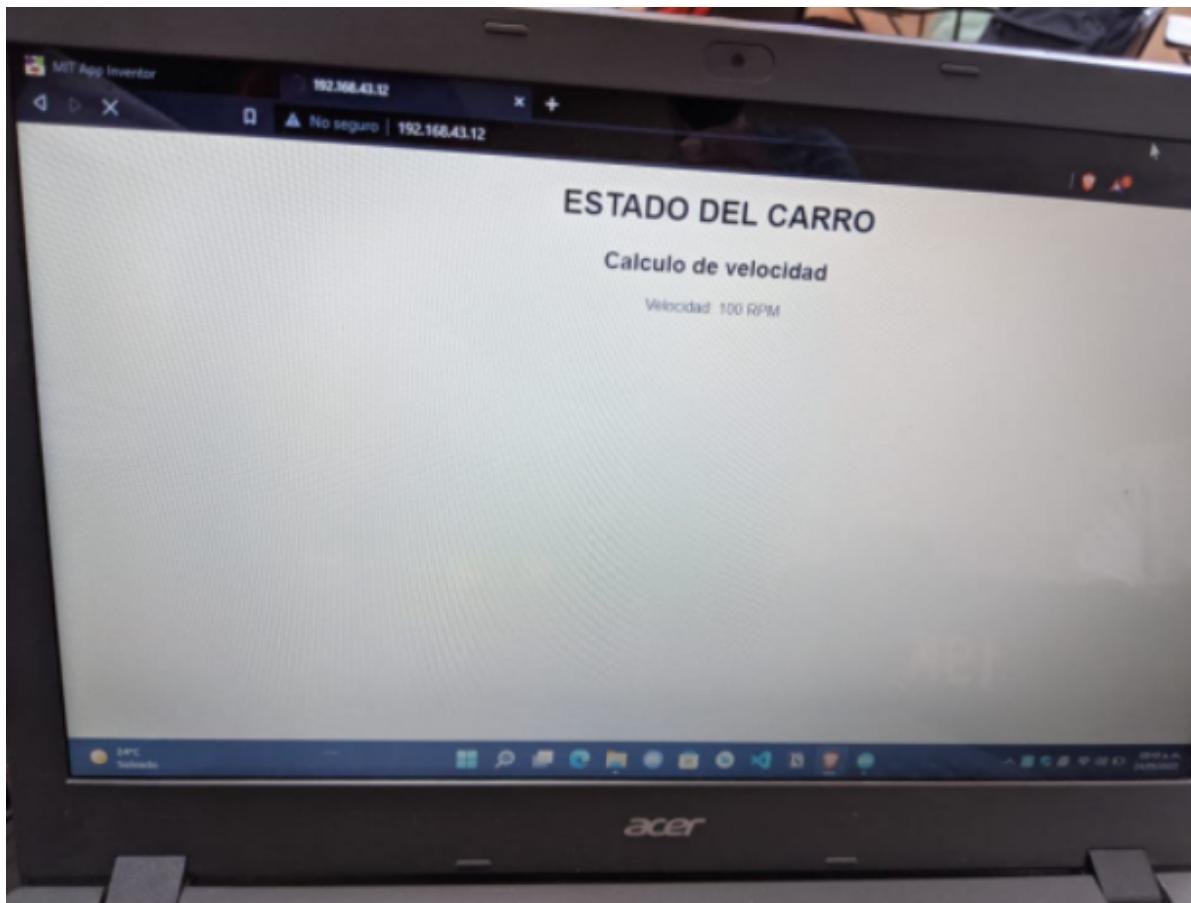
```

//La velocidad con 6V (9v de la pila menos 2 voltios que el Driver L293 quita) es de 200RPM, 200RPM es la velocidad maxima de los motores
//Las variables ENB y ENA controlan la velocidad de los motores del carro, por defecto son 150, el valor maximo de ENA o ENB es de 255
//Hacemos regla de 3
//velocidad = 100;
float porcentaje = (velocidad * 100 / 255);
int RPM = 200 * porcentaje;
client.println("Velocidad: ");
client.println(RPM / 100);
client.print(" RPM");
client.println("");

// Realiza la partición del cliente

```

La página que se muestra es la siguiente, es sencilla pero muestra el estado de las RPM.



Se leen las señales wifi enviadas desde la app para poder enviar una señal HIGH o LOW en el pin seleccionado, también se muestra el estado del pin

```
// Realiza la petición del cliente.
if (req.indexOf("onB") != -1) {
    digitalWrite(CTRL, HIGH);
    Serial.println("pin2ON");
}
if (req.indexOf("offB") != -1){
    digitalWrite(CTRL, LOW);
    Serial.println("pin2OFF");
}
if (req.indexOf("on5") != -1) {
    digitalWrite(ADL, HIGH);
    Serial.println("pin5ON");
}
if (req.indexOf("off5") != -1){
    digitalWrite(ADL, LOW);
    Serial.println("pin5OFF");
}
if (req.indexOf("on18") != -1) {
    digitalWrite(IZQ, HIGH);
    Serial.println("pin18ON");
}
if (req.indexOf("off18") != -1){
    digitalWrite(IZQ, LOW);
```

```

if (req.indexOf("on19") != -1) {
    digitalWrite(DER, HIGH);
    Serial.println("pin19ON");
}
if (req.indexOf("off19") != -1) {
    digitalWrite(DER, LOW);
    Serial.println("pin19OFF");
}
if (req.indexOf("on21") != -1) {
    digitalWrite(ATR, HIGH);
    Serial.println("pin21ON");
}
if (req.indexOf("off21") != -1) {
    digitalWrite(ATR, LOW);
    Serial.println("pin21OFF");
}

```

Por último se cierra la página web y se muestra que el cliente se desconectó, también se limpia el cliente y se detiene

```

client.println("</body></html>");

Serial.print("Cliente desconectado: ");
Serial.println(client.remoteIP());
client.flush();
client.stop();

```

Agregamos cambios al código del Arduino en el carro

Variables para leer los pines del esp32 e identificar si tiene que avanzar, retroceder, o girar hacia los lados, además un pin para identificar si se está usando la app o la antena para controlar el carro, en la parte de abajo se agregan las variables a las que se asigna el valor leído del pin correspondiente

```

int ADL = 30;
int IZQ = 32;
int DER = 34;
int ATR = 36;
int boton = 22;

int adelante = 0;
int izquierda = 0;
int derecha = 0;
int atras = 0;
int estadoBoton = 0;

```

En el código para funcionar desde la app primero leemos los pines, el resto del código es idéntico al del control de la antena con la excepción de que ahora en lugar de usar los ejes, se usan las variables

```
Serial.println("Operando con la app");
adelante = digitalRead(ADL);
izquierda = digitalRead(IZQ);
derecha = digitalRead(DER);
atras = digitalRead(ATR);

if(adelante == HIGH) {
    Serial.println("Adelante");
    digitalWrite(MotorA1, LOW);
    digitalWrite(MotorA2, HIGH);
    digitalWrite(MotorB1, HIGH);
    digitalWrite(MotorB2, LOW);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);

}else if(atras == HIGH) {
    Serial.println("Atras");
    digitalWrite(MotorA1, HIGH);
    digitalWrite(MotorA2, LOW);
    digitalWrite(MotorB1, LOW);
    digitalWrite(MotorB2, HIGH);
    analogWrite(ENA, 150);
    analogWrite(ENB, 150);
```

Por último se sustituye el código del loop, moviendo el código de la app y antena a dos funciones diferentes, para seleccionar uno de ellos se lee la entrada del pin 22 el cual indica por qué medio se está controlando

```
void loop() {
    estadoBoton = digitalRead(boton);
    if(estadoBoton == HIGH){
        app();
    }else{
        antena();
    }
}
```

Creando la app

Se utilizó la plataforma App inventor del MIT para crear la app ya que es fácil de usar y contiene los componentes necesarios para transmitir hacia el esp32 desde el celular

La plataforma permite realizar una interfaz mediante componentes ya definidos, solo es necesario arrastrarlos y colocarlos en la pantalla en la posición deseada, así mismo podemos editar propiedades como nombres del componente, tipo de letra, valores que transmite, etc.



Después en el apartado blocks podemos crear la programación mediante bloques, la cual se basa en enviar información a través de http el cual el esp32 leerá y ejecutará las operaciones necesarias.

El código utiliza las funciones touch down y touch up para detectar si el botón está siendo presionado o no, también se usa la función clic para ejecutar la acción con un solo toque sin necesidad de mantener presionado.

```

when [Button1 .Click]
do [set Web1 .Url to " http://192.168.43.12/onB "]
[call Web1 .Get]

when [Button2 .Click]
do [set Web1 .Url to " http://192.168.43.12/offB "]
[call Web1 .Get]

```

```

when [Botón1 .TouchDown]
do [set Web1 .Url to " http://192.168.236.12/on5 "]
[call Web1 .Get]

when [Botón1 .TouchUp]
do [set Web1 .Url to " http://192.168.236.12/off5 "]
[call Web1 .Get]

when [Botón2 .TouchDown]
do [set Web1 .Url to " http://192.168.236.12/on18 "]
[call Web1 .Get]

when [Botón2 .TouchUp]
do [set Web1 .Url to " http://192.168.236.12/off18 "]
[call Web1 .Get]

when [Botón3 .TouchDown]
do [set Web1 .Url to " http://192.168.236.12/on19 "]
[call Web1 .Get]

```

Por último usa el método positionchanged del slider para enviar los datos cada vez que la posición se mueva

```

when [Botón4 .TouchUp]
do [set Web1 .Url to " http://192.168.236.12/off21 "]
[call Web1 .Get]

when [Slider1 .PositionChanged]
  thumbPosition
do [set Label2 .Text to get thumbPosition]
[set Web1 .Url to [join " http://192.168.236.12/valor " get thumbPosition]]
[call Web1 .Get]

```

Para vincular la app al teléfono se utilizó su cliente de play store



PROBLEMAS DURANTE EL DESARROLLO DEL PROYECTO

El primer problema que enfrentamos fue la construcción de la base de acrílico debido a que el espacio para ajustar los tornillos y los motores era muy poco por lo cual tuvimos que conseguir un desarmador algo pequeño para que lograra entrar sin problema.

Después al construir y poner en funcionamiento el proyecto nos dimos cuenta que el sensor de inclinación no era tan preciso como necesitábamos por lo cual el carro tenía problemas para moverse debido a que los valores que leía por medio de la antena eran muy erráticos, para solucionar esta parte tuvimos que hacer pruebas únicamente con el control de inclinación para promediar los valores que generaba en cada eje y con esto ajustamos los valores que el carro necesitaba para moverse, después de realizar estas correcciones ya no tuvimos este problema.

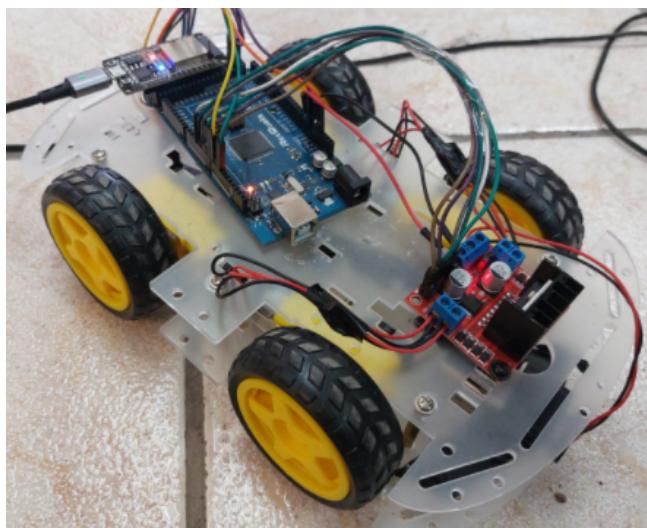
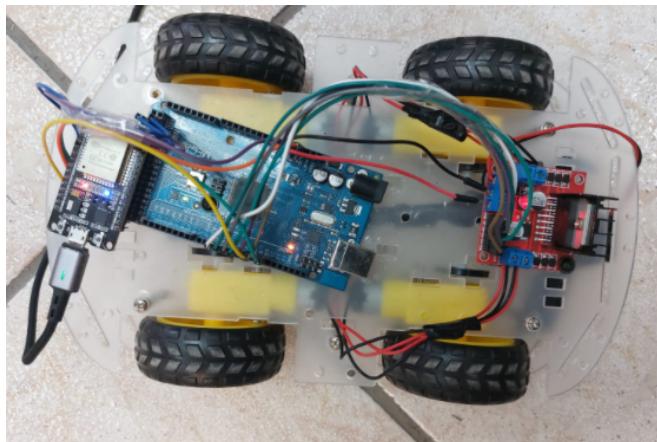
También fue necesario cambiar los pines de la antena ya que debido a la versión de arduino los pines para transmitir información están colocados de una manera diferente a arduino nano o uno.

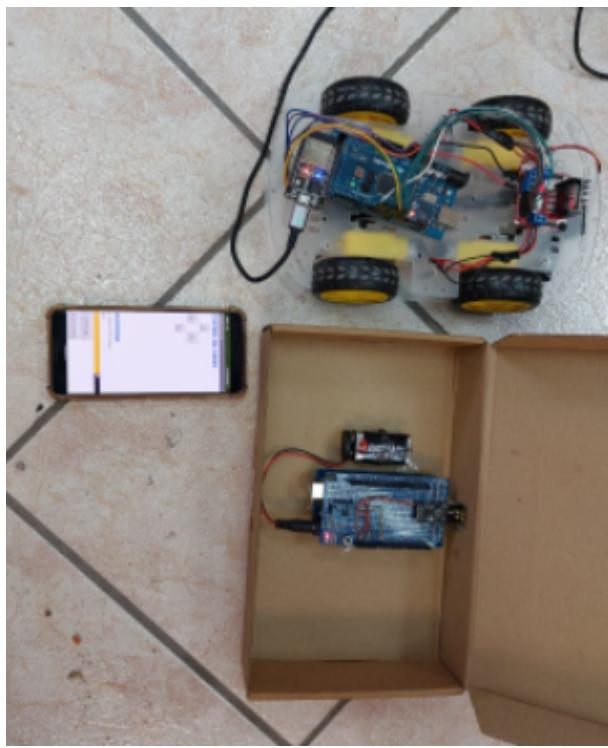
Luego más adelante tuvimos el problema de que al funcionar al mismo tiempo la app y el control de inclinación, el arduino en el carro tomaba los valores de ambos por lo cual no avanzaba al querer ir en dos diferentes direcciones que en ocasiones eran contrarias, para esto usamos un pin del esp32 para validar si se está usando la app o la antena.

El último problema que tuvimos fue con la batería del carro, debido a que utiliza cuatro motores cc consume muy rapido la bateria y llega el momento en el que la corriente ya no puede hacer que arranquen los motores, este problema pudo solucionarse consiguiendo una batería que sea de 9v recargable.

RESULTADOS

Este fue el proyecto final





El link para ver el video de funcionamiento es el siguiente

<https://youtu.be/FRsY2Ep1-BM>

Este proyecto nos ayudó a entender varios temas al mismo tiempo, primero entendimos como funciona arduino lo cual no fue difícil ya que su estructura de programación es muy sencilla por lo que no nos tomó mucho tiempo utilizarlo, también tuvimos que investigar acerca de diversos temas de electrónica ya que arduino va de la mano con ellos, tuvimos que comprender cómo se conectan estos componentes, que pines tienen y la función de ellos, cuanto voltaje utilizan, etc. para poder usarlos sin riesgo a quemarlo provocar un accidente.

Con este proyecto aunque parece algo sencillo nos abrió la puerta para poder crear otros sistemas de control.

CONCLUSIONES Y RECOMENDACIONES

Arduino es una plataforma muy fácil de entender y usar ya que cuenta con muchas librerías que ayudan a facilitar la programación de los diferentes componentes que

usa, además de tener una gran capacidad de expansión gracias a todos los puertos con los que cuenta (nosotros usamos la versión mega), esto nos permite agregar los componentes necesarios e incluso unir otro microcontrolador para combinar funciones, la parte “complicada” es el montaje de la electrónica debido a que se tiene que investigar como funciona cada pin, que salidas da, la forma en la que se comunica con arduino y si es necesario, las fórmulas para convertir los datos hacia algo que arduino pueda entender (como el caso de la antena).

Este proyecto nos resultó muy interesante ya que entendimos como es la base de los juguetes prearmados que venden, y el cómo utilizan las antenas de radio frecuencia para comunicarse sin causar interferencia entre ellos u otros dispositivos, además de poder realizar giros sin necesidad de mover las llantas hacia los lados.

REFERENCIAS

- [1] "Hand Gesture Controlled robot", Arduino Project Hub, 2022. [Online]. Available: https://create.arduino.cc/projecthub/abdelkader_ch/hand-gesture-controlled-robot-9e4282. [Accessed: 19- Apr- 2022].
- [2] "How To Make DIY Arduino Gesture Control Robot At Home", DIY Builder, 2019. [Online]. Available: <https://www.youtube.com/watch?v=svJwmjplm4c>. [Accessed: 19- Apr- 2022].
- [3] "ESP32 Useful Wi-Fi Library Functions (Arduino IDE) | Random Nerd Tutorials", *Random Nerd Tutorials*, 2022. [Online]. Available: <https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/>. [Accessed: 23- Apr- 2022].
- [4] "Primeros pasos con App Inventor 2". Código 21. <https://codigo21.educacion.navarra.es/autoaprendizaje/primeros-pasos-con-app-inventor-2/> (accedido el 22 de mayo de 2022).

[5] "ESP32 webserial: Web-based remote serial monitor | random nerd tutorials". Random Nerd Tutorials. <https://randomnerdtutorials.com/esp32-webserial-library/> (accedido el 22 de mayo de 2022).