

# Control Inalámbrico usando Arduino y módulos NRF24L01

## Material Y Equipo

- Kit Arduino Chasis De Carro de 2 llantas.
- 2 módulos inalámbricos 'NRF24L01' con antena SMA.
- 2 placas de desarrollo Arduino.
- Controlador de motores 'L293D'.
- Pila o batería portátil.
- 2 placas de pruebas.
- 4 botones.
- Cables.

## Armado y Conexiones

Para realizar la práctica se requiere el "Kit Arduino Chasis de Carro, 2 Llantas", el cual servirá como base para montar la tarjeta de pruebas, el Arduino y el módulo inalámbrico NRF24L01. El objetivo es contar con el receptor de la señal que controlará los dos motores encargados de mover las llantas.

El control se envía desde un transmisor mediante 4 botones, que permiten simular los movimientos: adelante, atrás, izquierda y derecha.

Una vez adquirido el kit, se procede a armar el carro de 2 llantas.

Primeramente, para la comunicación del control al carro, se conectan los módulos NRF24L01 a las placas Arduino UNO y Nano, formando las partes de transmisor y receptor. La conexión de los módulos se realiza de la siguiente manera:

Tabla 1. Conexión del módulo NRF24L01 a las placas Arduino.

NRF24L01	Arduino UNO, Nano
GND	Pin GND
3.3 V	Pin 3.3 V
CE	Pin 9
CSN	Pin 8
SCK	Pin 13
MOSI	Pin 11
MISO	Pin 12

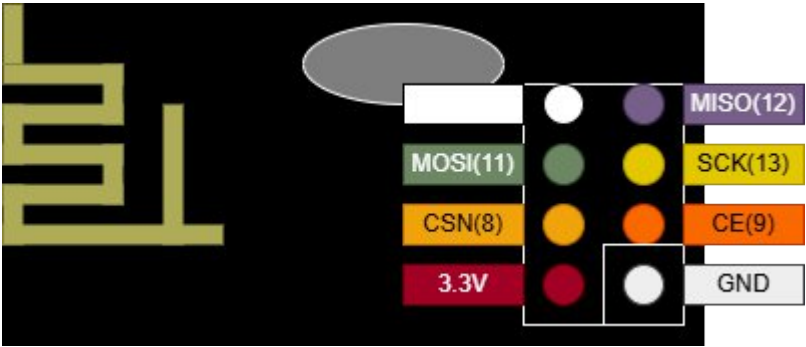


Ilustración 1. Conexión del módulo NRF24L01 a la placa de desarrollo Arduino.

Posteriormente, para la parte transmisora, se conectan los 4 botones en los pines digitales 2, 3, 4 y 5 del Arduino UNO:

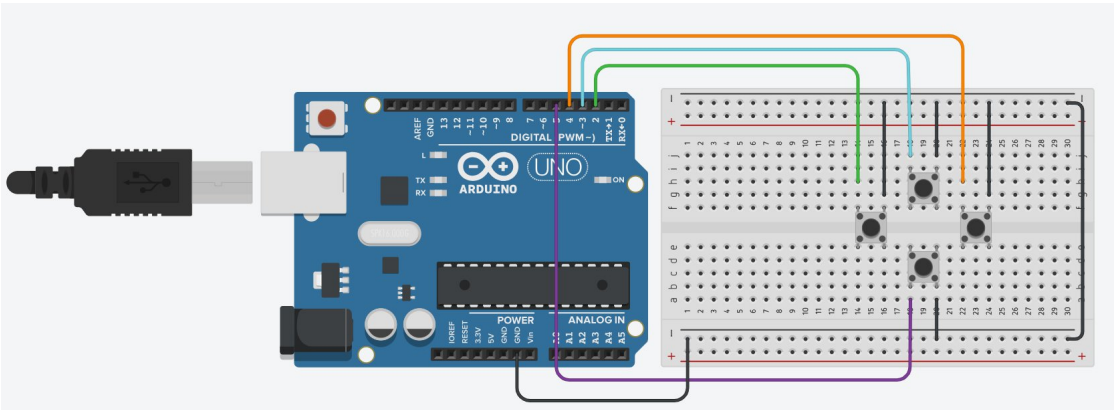
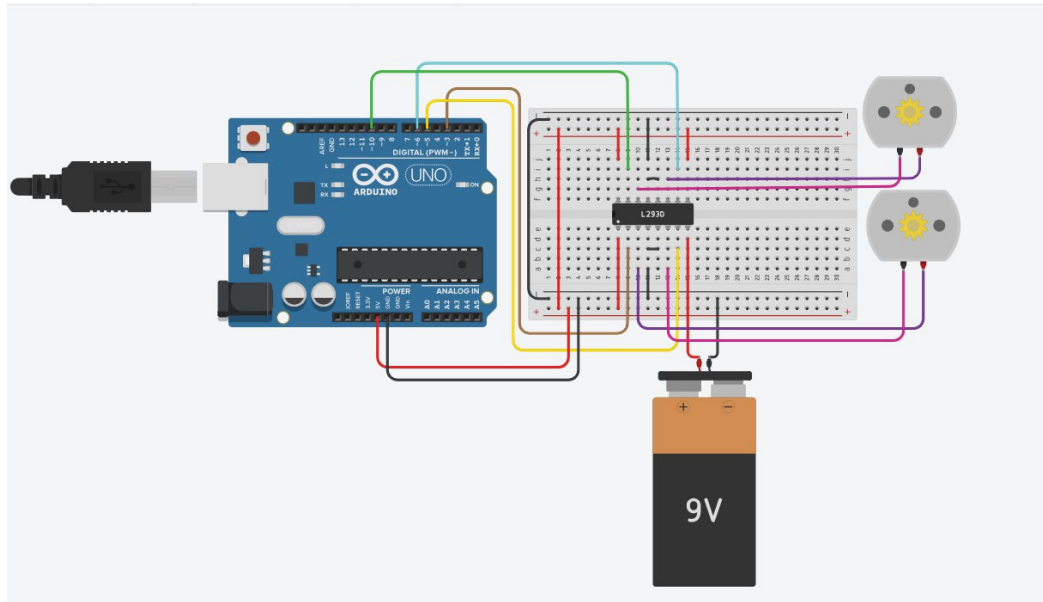


Ilustración 2. Parte transmisora de botones al Arduino UNO, realizado en Tinkercad.

Para la parte receptora se utiliza el circuito integrado L293D para controlar los dos motores del carro, siguiendo la conexión correspondiente y seleccionando las salidas PWM digitales para el control de ambos motores:



*Ilustración 3. Parte receptora de controlador de motores al Arduino UNO, realizado en Tinkercad.*

## Código de programación en IDE de Arduino

Se procede a establecer los códigos para la parte transmisora y receptora. En la sección transmisora se implementa un control con cuatro botones para direccionar el movimiento hacia adelante, atrás, izquierda y derecha, asignando cada función según la posición física de los botones en la tabla de pruebas.

En la parte receptora se reciben cuatro señales PWM que determinan el giro de los motores según la acción requerida.

Con la lógica antes descrita, se desarrollan los códigos necesarios, incluyendo la configuración de comunicación del módulo NRF24L01 mediante la instalación de su librería correspondiente en la IDE de Arduino para establecer la conexión inalámbrica.

Para la parte transmisora:

```
#include <SPI.h> //Se incluye la librería SPI.h
```

```

#include <nRF24L01.h> //Se incluye la libreria nRF24L01.h

#include <RF24.h> //Se incluye la libreria RF24.h


#define buttonPin1 2 //Boton izquierda

#define buttonPin2 3 //Boton arriba

#define buttonPin3 4 //Boton derecha

#define buttonPin4 5 //Boton abajo


int buttonState1 = 0; //Estado del boton 1

int buttonState2 = 0; //Estado del boton 2

int buttonState3 = 0; //Estado del boton 3

int buttonState4 = 0; //Estado del boton 4

RF24 radio(9, 8); // pines CE y CSN respectivamente


const byte address[6] = "00002"; //Direccion para la comunicacion entre modulos


void setup() {

    pinMode(buttonPin1, INPUT_PULLUP); //entrada con resistencia interna modo pullup activada

    pinMode(buttonPin2, INPUT_PULLUP); //entrada con resistencia interna modo pullup activada

    pinMode(buttonPin3, INPUT_PULLUP); //entrada con resistencia interna modo pullup activada

    pinMode(buttonPin4, INPUT_PULLUP); //entrada con resistencia interna modo pullup activada

    Serial.begin(9600); //Inicializa a 9600 baudios

    radio.begin(); //Inicializa la libreria de radio

    radio.openWritingPipe(address); //Busca la direccion de comunicacion establecida en modo transmisor

    radio.setPALevel(RF24_PA_MIN); //Potencia de transmision (minimo)

    radio.stopListening(); //Establece el modulo como transmisor

}

```

```
void loop() {

    buttonState1 = digitalRead(buttonPin1); //Lectura del valor del boton de izquierda

    buttonState2 = digitalRead(buttonPin2); //Lectura del valor del boton de arriba

    buttonState3 = digitalRead(buttonPin3); //Lectura del valor del boton de derecha

    buttonState4 = digitalRead(buttonPin4); //Lectura del valor del boton de abajo


    if (buttonState1 == 1) //Si no se presiona el boton de izquierda

    {

        buttonState1 = 1;

    }

    else if (buttonState1 == 0) //Si se presiona el boton de izquierda

    {

        buttonState1 = 0;

    }


    if (buttonState2 == 1) //Si no se presiona el boton de arriba

    {

        buttonState2 = 3;

    }

    else if (buttonState2 == 0) //Si se presiona el boton de arriba

    {

        buttonState2 = 2;

    }


    if (buttonState3 == 1) //Si no se presiona el boton de derecha

    {

        buttonState3 = 5;
```

```
}
```

```
else if (buttonState3 == 0) //Si se presiona el boton de derecha
```

```
{
```

```
    buttonState3 = 4;
```

```
}
```

```
if (buttonState4 == 1) //Si no se presiona el boton de abajo
```

```
{
```

```
    buttonState4 = 7;
```

```
}
```

```
else if (buttonState4 == 0) //Si se presiona el boton de abajo
```

```
{
```

```
    buttonState4 = 6;
```

```
}
```

```
Serial.print(buttonState1); //Imprime en monitor serial el valor del boton de izquierda
```

```
Serial.print("\t"); //separacion
```

```
Serial.print(buttonState2); //Imprime en monitor serial el valor del boton de arriba
```

```
Serial.print("\t"); //separacion
```

```
Serial.print(buttonState3); //Imprime en monitor serial el valor del boton de derecha
```

```
Serial.print("\t"); //separacion
```

```
Serial.print(buttonState4); //Imprime en monitor serial el valor del boton de abajo
```

```
Serial.print("\t"); //separacion
```

```
radio.write(&buttonState1, sizeof(buttonState1)); //Manda el valor del boton de izquierda, estableciendo el tamaño en bytes de la variable
```

```
radio.write(&buttonState2, sizeof(buttonState2)); //Manda el valor del boton de arriba, estableciendo el tamaño en bytes de la variable
```

```

    radio.write(&buttonState3, sizeof(buttonState3)); //Manda el valor del boton de derecha, estableciendo el tamaño en
bytes de la variable

    radio.write(&buttonState4, sizeof(buttonState4)); //Manda el valor del boton de abajo, estableciendo el tamaño en bytes
de la variable

}

```

## Para la parte receptora:

```

#include <SPI.h> //Se incluye la libreria SPI.h

#include <nRF24L01.h> //Se incluye la libreria nRF24L01.h

#include <RF24.h> //Se incluye la libreria RF24.h


#define izqA 3 //Avanza llanta izquierda

#define izqB 5 //Retroceso llanta izquierda

#define derA 6 //Avanza llanta derecha

#define derB 10 //Retroceso llanta derecha

int vel = 255; // Velocidad de los motores (0-255)

int paro = 0; //Determina cuando se deja de presionar un boton


int buttonState = 0; //Señal captada del modulo NRF24L01


RF24 radio(9, 8); // pines CE y CSN respectivamente

const byte address[6] = "00002"; //Direccion para la comunicacion entre modulos


void setup() {

    Serial.begin(9600); //Inicializa a 9600 baudios

```

```

pinMode(derA, OUTPUT); //Salida digital PWM de avanza llanta izquierda

pinMode(derB, OUTPUT); //Salida digital PWM de retroceso llanta izquierda

pinMode(izqA, OUTPUT); //Salida digital PWM de avanza llanta derecha

pinMode(izqB, OUTPUT); //Salida digital PWM de retroceso llanta derecha


radio.begin(); //Inicializa la libreria de radio

radio.openReadingPipe(0, address); //Busca la direccion de comunicacion establecida en modo receptor

radio.setPALevel(RF24_PA_MIN); //Potencia de transmision (minimo)

}

void loop() {

    radio.startListening(); //Establece el modulo como receptor

    while (!radio.available()); //Mientras la comunicaci3n reciba una se1al entrante del modulo

    radio.read(&buttonState, sizeof(buttonState)); //Lee la se1al recibida y la almacena

    Serial.println(buttonState); //Muestra en monitor serial el valor recibido


    if (buttonState == 0 && paro == 0) { //Si el boton de izquierda esta presionado

        analogWrite(izqB, vel); //Retrocede la llanta izquierda

        analogWrite(derA, vel); //Avanza la llanta derecha

        paro = 1;

    } else if (buttonState == 1 && paro == 1){ //Si el boton de izquierda se deja de presionar

        analogWrite(izqB, 0); //Se detiene la llanta izquierda retroceso

        analogWrite(derA, 0); //Se detiene la llanta derecha avance

        paro = 0;

    }


    if (buttonState == 2 && paro == 0) { //Si el boton de arriba esta presionado

        analogWrite(izqA, vel); //Izquierda Avanza

```



```

analogWrite(derA, vel); //Derecha Avanza

paro = 2;

} else if (buttonState == 3 && paro == 2){ //Si el boton de arriba se deja de presionar

    analogWrite(izqA, 0); //Se detiene la llanta izquierda avance

    analogWrite(derA, 0); //Se detiene la llanta derecha avance

    paro = 0;

}

if (buttonState == 4 && paro == 0) { //Si el boton de derecha esta presionado

    analogWrite(izqA, vel); //Izquierda Avanza

    analogWrite(derB, vel); //Derecha Retrocede

    paro = 3;

} else if (buttonState == 5 && paro == 3){ //Si el boton de derecha se deja de presionar

    analogWrite(izqA, 0); //Se detiene la llanta izquierda avance

    analogWrite(derB, 0); //Se detiene la llanta derecha retroceso

    paro = 0;

}

if (buttonState == 6 && paro == 0) { //Si el boton de abajo esta presionado

    analogWrite(izqB, vel); //Izquierda retrocede

    analogWrite(derB, vel); //Derecha retrocede

    paro = 4;

} else if (buttonState == 7 && paro == 4){ //Si el boton de abajo se deja de presionar

    analogWrite(izqB, 0); //Se detiene la llanta izquierda retroceso

    analogWrite(derB, 0); //Se detiene la llanta derecha retroceso

    paro = 0;

}

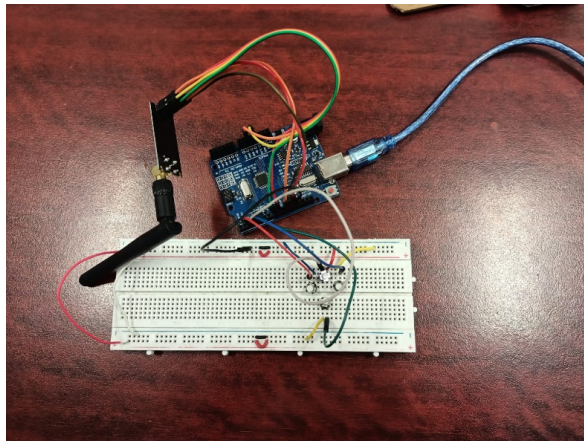
```

}

Se cargan los códigos correspondientes en los Arduino UNO designados. Posteriormente, se monta la placa de pruebas, el Arduino Nano y la pila en el chasis del carro de 2 llantas, verificando su funcionamiento.

## Resultados

Se obtiene una comunicación satisfactoria, gracias a la rápida respuesta del control de 4 botones hacia el receptor del carro, logrando un movimiento adecuado y acorde con la función asignada a cada botón.



*Ilustración 4. Parte transmisora de 4 botones para el movimiento del carro.*



*Ilustración 5. Parte receptora del chasis del carro de 2 llantas con control de motores inalámbrico.*