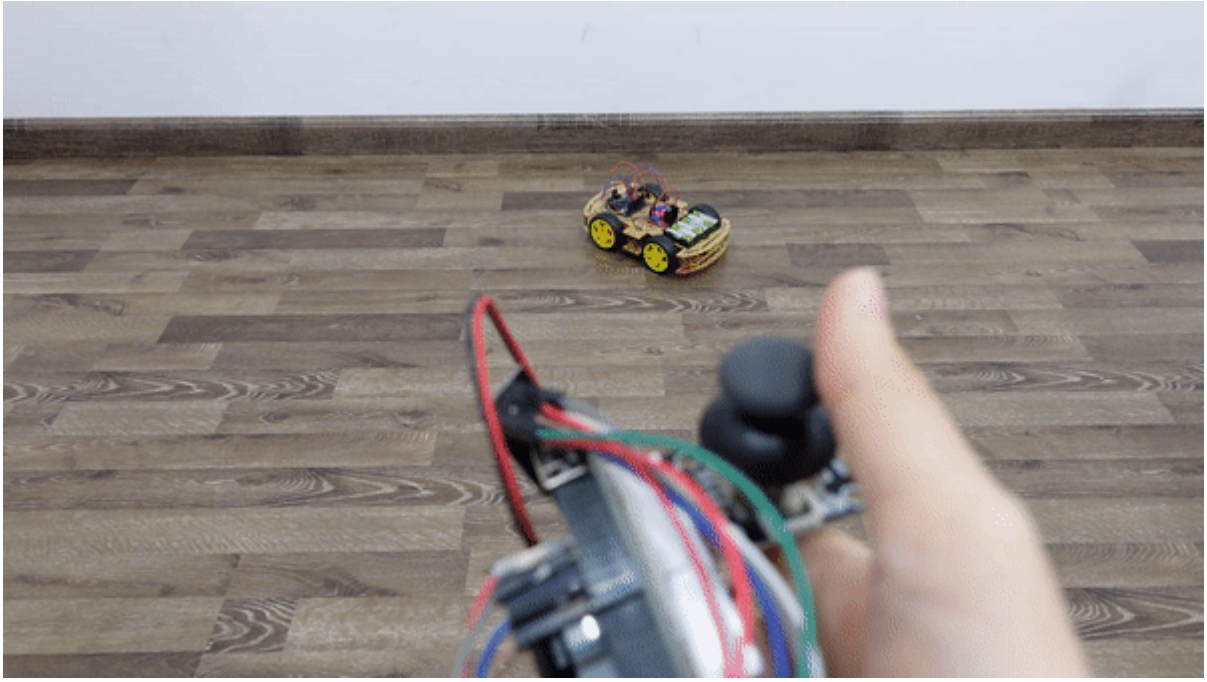
صنع جهاز للتحكم عن بعد بالروبوت باستخدام (NRF24L01)

تتطلب بعض المشاريع الإلكترونية، توصيل القطع لاسلكياً معاً سواء لإرسال البيانات او استقبالها او إرسال إشارات للتحكم في تشغيل أو إغلاق القطع أو قراءة بيانات من الحساسات، في هذا الدرس سنتعرف على وحدة (NRF24L01) والتي سنعمل على برمجتها مع الأردوينو وعصا التحكم لصنع جهاز تحكم عن بعد بالروبوت.

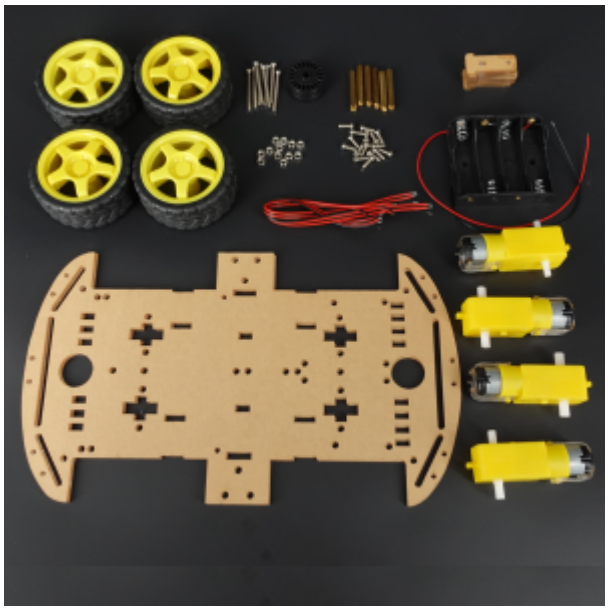


المواد و الأدوات

2X اردوينو أونو

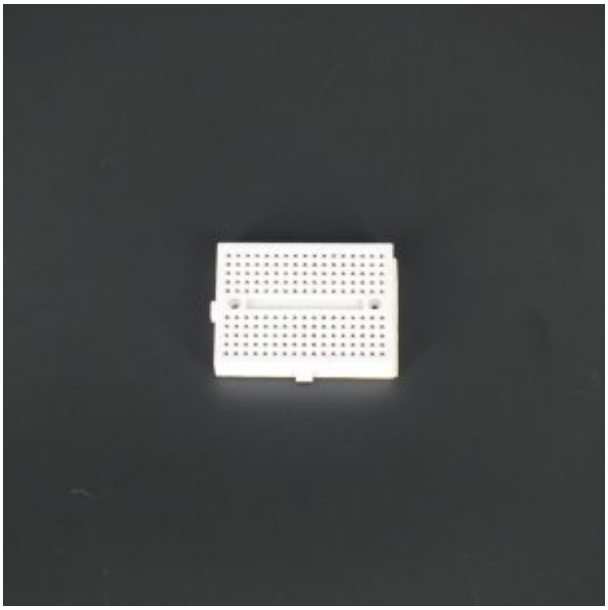
X1 سلك أردوينو

2X وحدة إرسال و استقبال (NRF24L01)



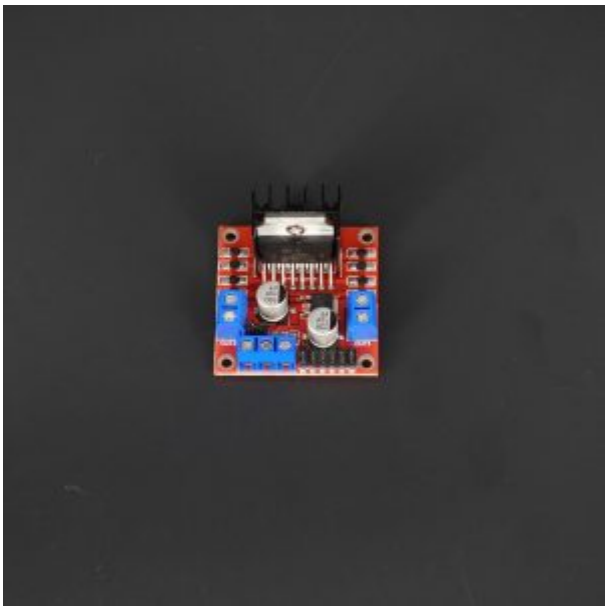
1X هيكل روبوت بأربع عجلات

1X عصا تحكم

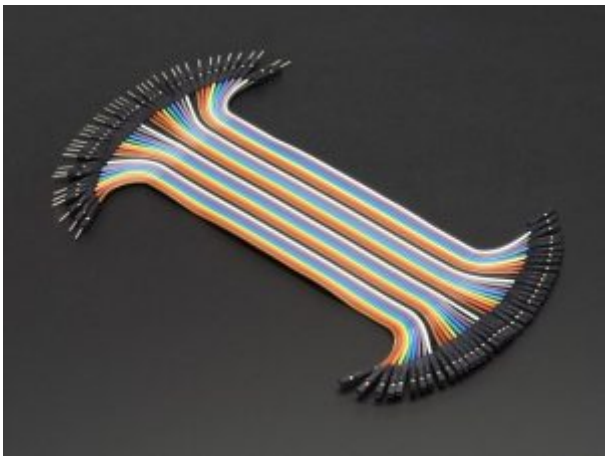


1X لوحة تجارب

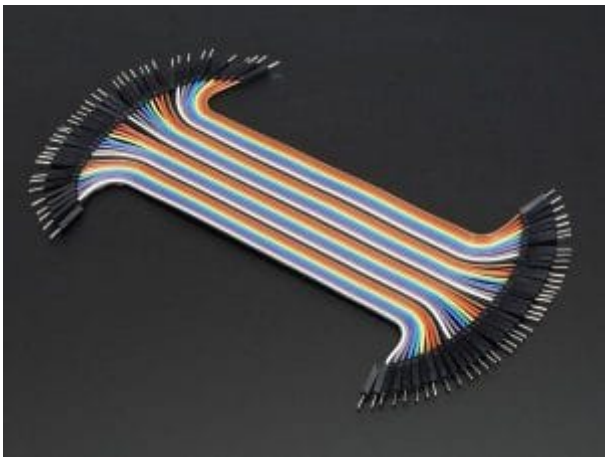
6X بطاريات (يفضل توفير حامل بطاريات لعدد 6)



1X دائرة التحكم بالمحركات (L298)



أسلاك توصيل (ذكر/ أنثى)

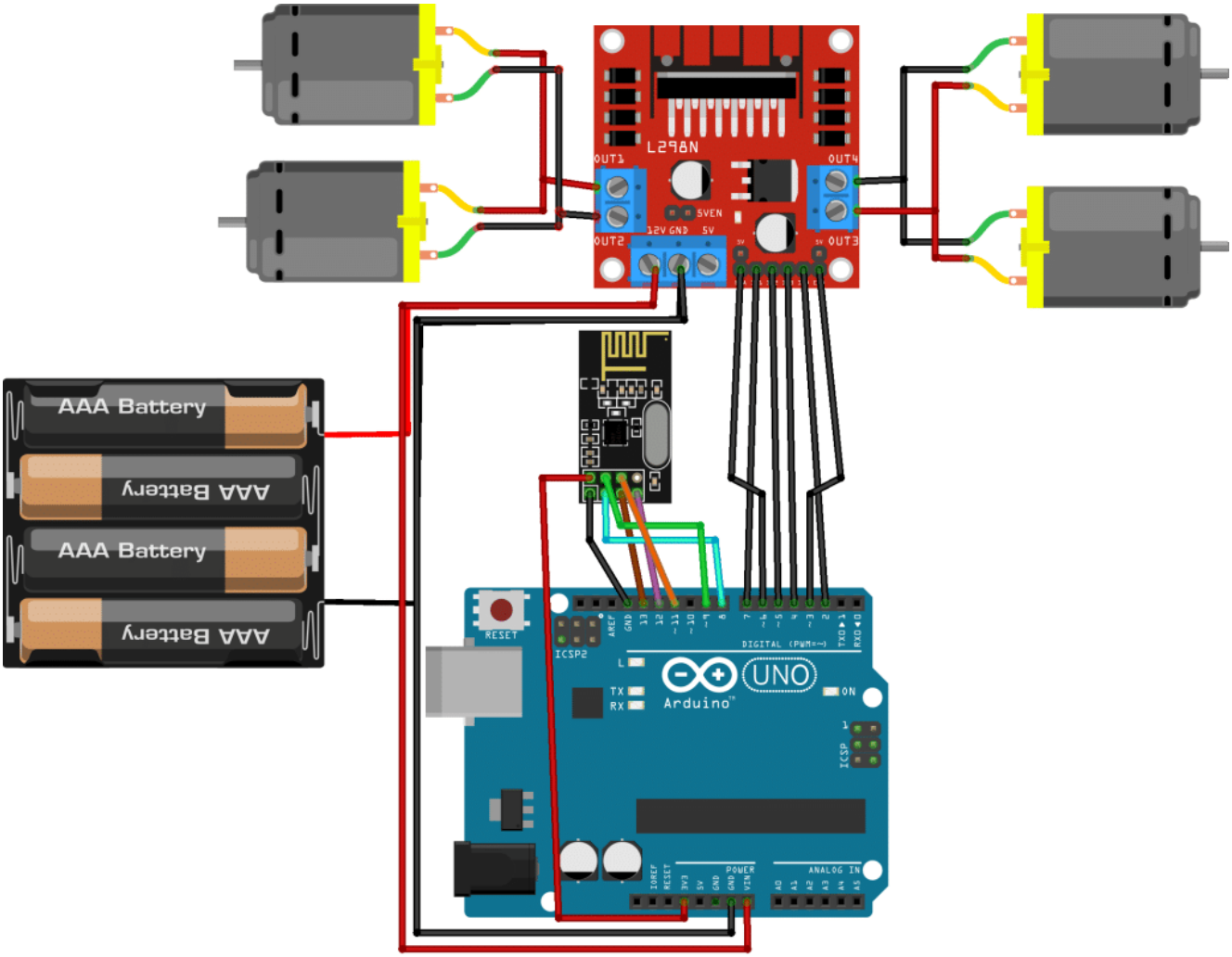


أسلاك توصيل (ذكر/ ذكر)

توصيل الدائرة

بالبداية قم بالاطلاع على الدرس لمعرفة كيفة تثبيت هيكل الروبوت

أولا: توصيل دائرة المستقبل



توصيل المحركات :

أسلاك المحركات	L298H-bridge
الأسلاك ذات اللون الأسود على جهة اليسار (ـ)	OUT 2
الأسلاك ذات اللون الأحمر على جهة اليسار (+)	OUT 1
الأسلاك ذات اللون الأسود على جهة اليمين (ـ)	OUT 4
الأسلاك ذات اللون الأحمر على جهة اليمين (+)	OUT 3

توصيل المنافذ الرقمية من الاردوينو مع L298H-bridge :

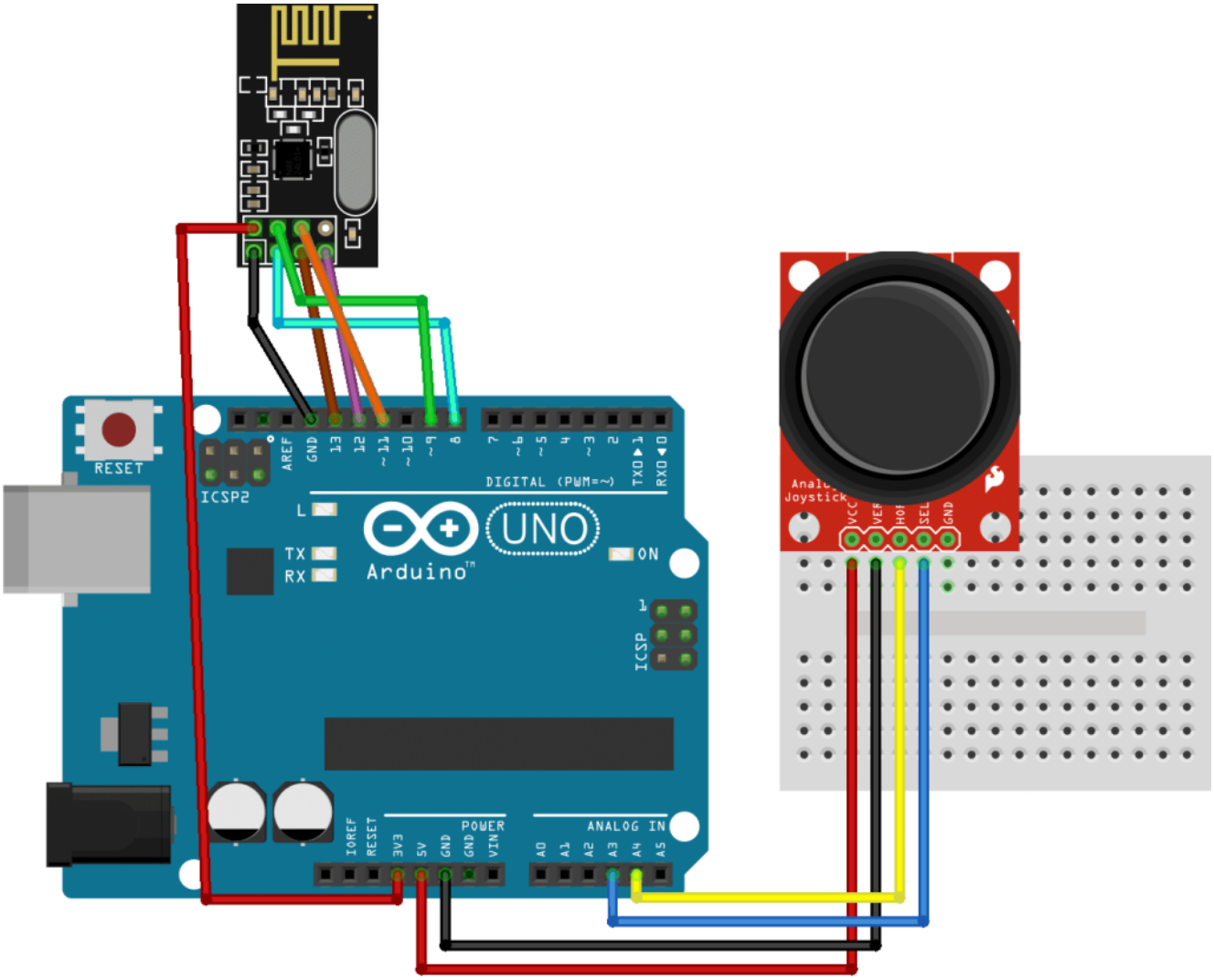
الأردوينو	L298H-bridge
6	enA
7	IN 1
5	IN 2
4	IN 3
2	IN4
3	enB

توصيل وحدة إرسال و استقبال (NRF24L01)

الأردوينو	NRF24L01
3.3	VCC
9	CSN
11	MOSI
GND	GND
8	CE
13	SCK
12	MISO

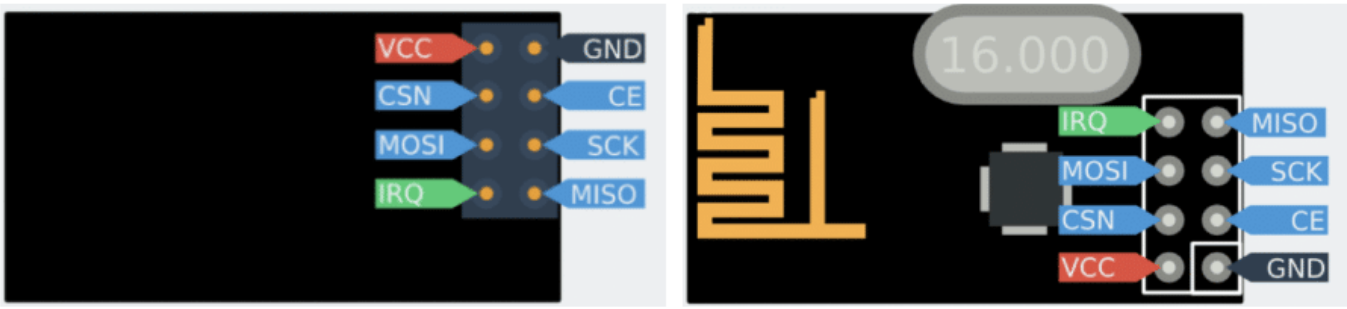
ثانيا: توصيل دائرة المرسل

سيتم توصيل وحدة اللإرسال و الاستقبال (NRF24L01) بنفس الربط في المستقبل



وحدة الارسال و الاستقبال (NRF24L01)

تسمح وحدة الإرسال والاستقبال nRF24L01 لمتحكمين أو أكثر بالتواصل مع بعضهما البعض لاسلكيًا وتعمل على بروتوكول SPI.
ونطاق ترددها 2.4 جيجا هرتز ونطاق الاتصال يصل إلى 100 متر.



الجدول يوضح وظائف المنافذ للوحدة

المنفذ	الوظيفة

VCC	يستخدم هذا المنفذ لتزويد الوحدة بالطاقة. يمكن أن يتراوح الجهد الكهربائي من 1.9 إلى 3.9 فولت
CSN (Chip Select Not)	وهو عبارة عن منفذ نشط منخفض. نحتاج إلى إبقائه مرتفعًا إلا عندما نرسل للجهاز أمر SPI أو عندما نتلقى بيانات على ناقل SPI من وحدة التحكم.
MOSI (Master Out Slave In)	مدخل SPI إلى nRF24L01. يتم استخدامه لتلقي البيانات من متحكم
IRQ	إرسال إشارة تنبيه عند توفر بيانات جديدة للمعالجة.
MISO (Master In Slave Out)	إنه خرج SPI من nRF24L01. يتم استخدامه لإرسال البيانات إلى وحدة التحكم
SCK (Serial Clock)	يقبل نبضات الساعة التي يوفرها مدير ناقل SPI.
CE (Chip Enable)	منفذ التمكين الخاص بالوحدة المستخدمة لتحديد وضع nRF24L01 مرسل أو مستقبل، اعتمادًا على الوضع الموجود فيه حاليًا.
GND	الأرضي

البرمجة

بالبداية تحتاج إلى تحميل مكتبة (RF24 master) موجدة في إدارة المكتبات في بيئة تطوير الأردوينو،

يمكن معرفة اضافة المكتبات من خلال الرابط

الشفرة البرمجية للمرسل

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(8,9); // CE, CSN
const byte address[6] = "00001";
char xyData[32] = "";
int joystick[2];
void setup() {
Serial.begin(9600);
radio.begin();
radio.openWritingPipe(address);
```

```
radio.setPALevel(RF24_PA_MAX);
radio.stopListening();
}
void loop() {

joystick[0] = analogRead(A4);
joystick[1] = analogRead(A3);

radio.write( joystick, sizeof(joystick) );
}
```

الشفرة البرمجية للمستقبل

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define enA 6
#define in1 7
#define in2 5
#define enB 3
#define in3 4
#define in4 2

RF24 radio(8,9); // CE, CSN
const byte address[6] = "00001";
char receivedData[32] = "";
int xAxis, yAxis;
int motorSpeedA = 0;
int motorSpeedB = 0;
int joystick[2];

void setup() {
pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
Serial.begin(9600);
radio.begin();
radio.openReadingPipe(0, address);
radio.setPALevel(RF24_PA_MAX);
radio.startListening();

digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}
void loop() {

if (radio.available()) { // If the NRF240L01 module received data
```

```
radio.read( joystick, sizeof(joystick) );

radio.read(&receivedData, sizeof(receivedData));
yAxis = joystick[0];
xAxis = joystick[1];

Serial.println(yAxis);
Serial.println(xAxis);

}

if (yAxis < 470) {

digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);

digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);

motorSpeedA = map(yAxis, 470, 0, 0, 255);
motorSpeedB = map(yAxis, 470, 0, 0, 255);
}
else if (yAxis > 550) {

digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);

digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);

motorSpeedA = map(yAxis, 550, 1023, 0, 255);
motorSpeedB = map(yAxis, 550, 1023, 0, 255);
}

else {
motorSpeedA = 0;
motorSpeedB = 0;
}

if (xAxis < 470) {

int xMapped = map(xAxis, 470, 0, 0, 255);

motorSpeedA = motorSpeedA - xMapped;
motorSpeedB = motorSpeedB + xMapped;
// Confine the range from 0 to 255
if (motorSpeedA < 0) {
motorSpeedA = 0;
}
if (motorSpeedB > 255) {
motorSpeedB = 255;
}
}
```

```
if (xAxis > 550) {

int xMapped = map(xAxis, 550, 1023, 0, 255);

motorSpeedA = motorSpeedA + xMapped;
motorSpeedB = motorSpeedB - xMapped;

if (motorSpeedA > 255) {
motorSpeedA = 255;
}
if (motorSpeedB < 0) {
motorSpeedB = 0;
}
}

if (motorSpeedA < 70) {
motorSpeedA = 0;
}
if (motorSpeedB < 70) {
motorSpeedB = 0;
}
analogWrite(enA, motorSpeedA); // Send PWM signal to motor A
analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
}
```

شرح الشفرة البرمجية للمرسل

تضمين المكتبات الضرورية مثل (SPI,nRF24L01,RF24)

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

إنشاء كائن (RF24) يحتوي على معاملين و هي منافذ (CE, CSN)

```
RF24 radio(8,9); // CE, CSN
```

نحتاج إلى إنشاء مصفوفة تمثل العنوان الذي ستتواصل من خلاله وحدة الإرسال و الاستقبال. يمكننا تغيير قيمة العنوان إلى أي سلسلة مكونة من 5 أحرف وهذا يتيح اختيار أي جهاز استقبال سيوصل مع المرسل، سيكون نفس العنوان في كل من جهاز الاستقبال وجهاز الإرسال.

```
const byte address[6] = "00001";
```

نحدد حجم ونوع البيانات التي سيتم إرسالها

```
char xyData[32] = "";
```

نعرف مصفوفة من متغيرين ستمثل قيمة محاور عصا التحكم (x,y)

```
int joystick[2];
```

في دالة التهئية نهيئ الاتصال التسلسلي و الراديو

```
void setup() {
Serial.begin(9600);
radio.begin();
```

نحتاج إلى تهيئة كائن الراديو باستخدام وظيفة radio.openWritingPipe () ونعين عنوان جهاز الاستقبال الذي سنرسل إليه البيانات

```
radio.openWritingPipe(address);
```

ثم باستخدام وظيفة radio.setPALevel () ، نعين مستوى مضخم الطاقة ، وفي حالتنا سأقوم بتعيينه على الحد الأعلى حتى نتمكن من التحكم بالروبوت على مسافات أكبر.

```
radio.setPALevel(RF24_PA_MAX);
```

نستخدم دالة radio.stopListening(); لتحديد أن الوحدة هي وحدة الإرسال

```
radio.stopListening();
}
```

في دالة void loop نحدد مصفوفة لقيم محور y والتي تحدد حركة الماتور إلى الأمام والخلف ومصفوفة لقيم محور xالتي تحدد اتجاه الحركة إلى اليمين أو اليسار

```
void loop() {

joystick[0] = analogRead(A4);
joystick[1] = analogRead(A3);
```

نستخدم دالة radio.write لإرسال البيانات و تحتوي الدالة على معيارين الأول يمثل قيمة عصا التحكم و الثاني حجم البيانات التي سيتم نقلها و حددناها (sizeof) بمعنى العدد الفعلي من البايتات (Byts) لمصفوفة (Joystick))

```
radio.write( joystick, sizeof(joystick) );
}
```

شرح الشفرة البرمجية للمستقبل

تضمين المكتبات الضرورية مثل (SPI) و (nRF24L01) و(RF24)

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

تعريف منافذ وحدة L298

```
#define enA 6
#define in1 7
#define in2 5
#define enB 3
#define in3 4
#define in4 2
```

إنشاء كائن (RF24) يحتوي على معاملين و هي منافذ (CE, CSN)

```
RF24 radio(8,9); // CE, CSN
```

نحتاج إلى إنشاء مصفوفة تمثل العنوان سيكون نفس العنوان في كل من جهاز الاستقبال وجهاز الإرسال.

```
const byte address[6] = "00001";
```

نحدد حجم ونوع البيانات التي سيتم استقبالها

```
char receivedData[32] = "";
```

نعرف متغيران لقيمة (X,Y)

```
int xAxis, yAxis;
```

نحدد قيمة افتراضية تساوي 0 لسرعة المحركات motorSpeedA و motorSpeedB

```
int motorSpeedA = 0;
int motorSpeedB = 0;
```

نعرف مصفوفة من متغيرين ستمثل قيمة محاور عصا التحكم (x,y)

```
int joystick[2];
```

في دالة التهيئة نعرف منافذ الدخل و الخرج لوحدة L298

```
void setup() {
pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
```

نهيئ الإتصال التسلسلي و الراديو

```
Serial.begin(9600);
```

```
radio.begin();
```

نحتاج إلى تهيئة كائن الراديو باستخدام وظيفة radio.openReadingPipe () ونعين عنوان جهاز الإرسال الذي سنستلم منه البيانات

```
radio.openReadingPipe(0, address);
```

ثم باستخدام وظيفة radio.setPALevel () ، نعين مستوى مضخم الطاقة ، وفي حالتنا سأقوم بتعيينه على الحد الأعلى حتى نتمكن من التحكم بالروبوت على مسافات أكبر.

```
radio.setPALevel(RF24_PA_MAX);
```

نستخدم دالة radio.startListening(); لتحديد أن الوحدة هي وحدة الاستقبال

```
radio.startListening();
```

اعطاء إشارة منخفضة لجميع منافذ التحكم في وحدة

```
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}
```

نحتاج العبارة المنطقية if لنتحقق من وصول إشارة

```
if (radio.available()) { // If the NRF240L01 module received data
```

نستخدم دالة radio.read لاستقبال البيانات و تحتوي الدالة على معيارين الأول يمثل قيمة عصا التحكم و الثاني حجم البيانات التي سيتم استلامها و حددناها (sizeof) بمعنى العدد الفعلي من البايتات (Byts) لمصفوفة (Joystick)

```
radio.read( joystick, sizeof(joystick) );
```

تخزين البيانات في مصفوفة joystick

```
radio.read(&receivedData, sizeof(receivedData));
yAxis = joystick[0];
xAxis = joystick[1];
```

نستخدمة دالة Serial.println لطباعة قيم عصا التحكم (ان احتجت للتحقق من القراءات)

```
Serial.println(yAxis);
Serial.println(xAxis);}
```

يستخدم المحور Y للتحكم بالحركة للأمام أو الخلف

```
if (yAxis < 470) {
```

يتحرك الروبوت إلى الخلف

```
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);

digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
```

قم بتحويل قراءات المحور Y المتدنية للعودة للخلف من 470 إلى 0 إلى قيمة 0 إلى 255 لإشارة PWM لزيادة سرعة المحرك

```
motorSpeedA = map(yAxis, 470, 0, 0, 255);
motorSpeedB = map(yAxis, 470, 0, 0, 255);
}
```

يتحرك الروبوت إلى الأمام

```
else if (yAxis > 550) {

digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);

digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
```

قم بتحويل قراءات المحور Y المتدنية للتحرك للأمام من 470 إلى 0 إلى قيمة 0 إلى 255 لإشارة PWM لزيادة سرعة المحرك

```
motorSpeedA = map(yAxis, 550, 1023, 0, 255);
motorSpeedB = map(yAxis, 550, 1023, 0, 255);
}
```

إذا بقيت عصا التحكم في المنتصف فإن المحركات لا تتحرك

```
else {
motorSpeedA = 0;
motorSpeedB = 0;
}
```

يتحكم محور X بالمحركات للتحرك باتجاه اليمين و اليسار

```
if (xAxis < 470) {
```

قم بتحويل قراءات المحور X المتزايدة من 0 إلى 470 إلى قيمة 0 إلى 255

```
int xMapped = map(xAxis, 470, 0, 0, 255);
```

يتحرك لليمين بتقليل سرعة المحرك اليمين و زيادة سرعة المحرك على اليسار

```
motorSpeedA = motorSpeedA - xMapped;
motorSpeedB = motorSpeedB + xMapped;
```

```
// Confine the range from 0 to 255
if (motorSpeedA < 0) {
motorSpeedA = 0;
}
if (motorSpeedB > 255) {
motorSpeedB = 255;
}
}
if (xAxis > 550) {
```

قم بتحويل قراءات المحور X المتزايدة من 550 إلى 1024 إلى قيمة 0 إلى 255

```
int xMapped = map(xAxis, 550, 1023, 0, 255);
```

يتحرك لليسار بتقليل سرعة المحرك اليسار و زيادة سرعة المحرك على اليمين

```
motorSpeedA = motorSpeedA + xMapped;
motorSpeedB = motorSpeedB - xMapped;

if (motorSpeedA > 255) {
motorSpeedA = 255;
}
if (motorSpeedB < 0) {
motorSpeedB = 0;
}
}

if (motorSpeedA < 70) {
motorSpeedA = 0;
}
if (motorSpeedB < 70) {
motorSpeedB = 0;
}
```

إرسال اشارة تضمين عرض النبضة

```
analogWrite(enA, motorSpeedA); // Send PWM signal to motor A
analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
}
```