




Lesson 3 ライン追跡車





Points of the section

このレッスンでは、車を滑走路に沿って移動するように制御する方法を学習します。

Learning parts:

-  ライントラッキングモジュールの使い方を学ぶ
-  ライントラッキングの原則を学ぶ
-  プログラミングによるライントラッキングの実装方法を学ぶ

準備:

-  カー（バッテリーを装備）
-  USB ケーブル
-  3つのライントラッキングモジュール
-  黒いテープのロール

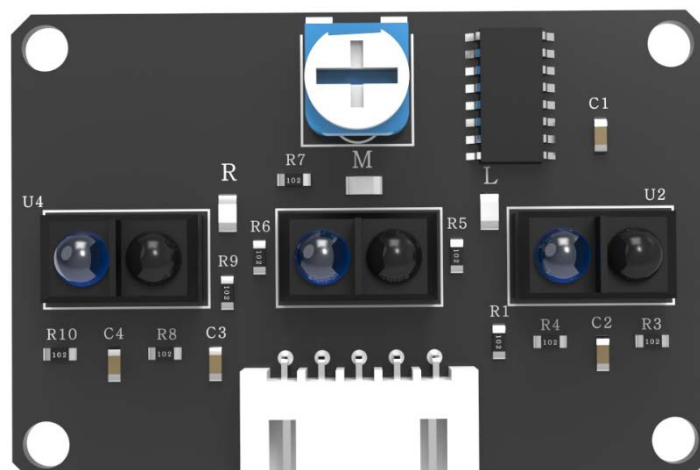
I . Making runway

材料：電気絶縁テープ（黒色）

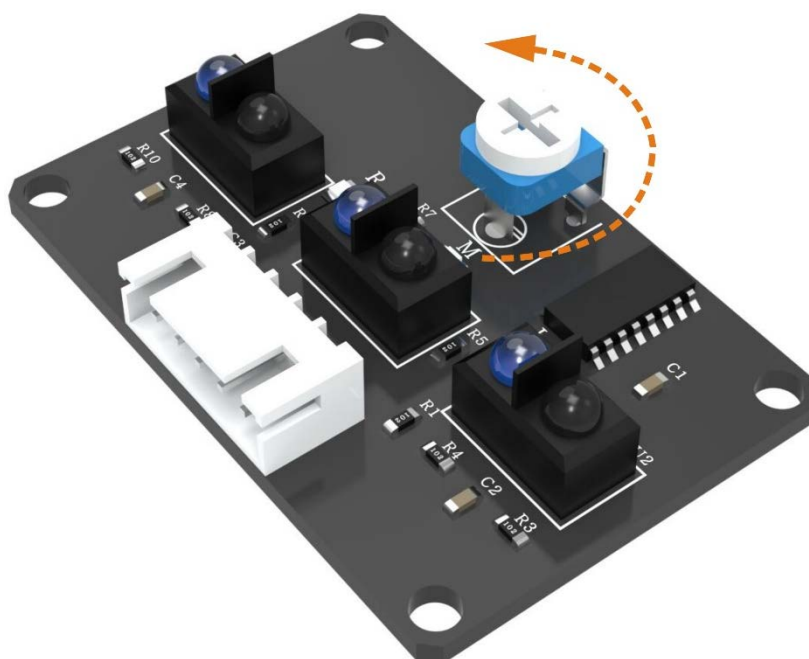
まず第一に、自分たちで滑走路を作る必要があります。適切な紙や地面に黒いテープを貼り付けることで回路を作ることができます。ペーストする前に、走路をペンで描き、電気粘着テープで貼り付けることができます。コーナーを可能な限り滑らかにするように注意してください。角度が小さすぎると車がラインから出ますが、もっと難しくしたい場合は小さくすることができます。走路のサイズは一般に 40×60cm 以上である。



II. モジュールを接続してデバッグする



指し示されているコンポーネントはポテンショメータです。 ライトラッキングモジュールの感度を調整するには、抵抗値を変更します。



III. Upload program

走路とモジュールを接続したら、コードファイル “¥ Lesson 3 Line Tracking Car ¥ Line_Tracking_Car ¥ Line_Tracking_Car.ino”を開き、プログラムを UNO コントローラボードにアップロードするだけです。

Code preview:

```
//www.elegoo.com

//Line Tracking IO define
#define LT_R !digitalRead(10)
#define LT_M !digitalRead(4)
#define LT_L !digitalRead(2)

#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11

#define carSpeed 150

void forward(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go forward!");
}

void back(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go back!");
}
```

```
}

void left(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go left!");
}

void right(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go right!");
}

void stop(){
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);
  Serial.println("Stop!");
}

void setup(){
  Serial.begin(9600);
  pinMode(LT_R, INPUT);
  pinMode(LT_M, INPUT);
  pinMode(LT_L, INPUT);
}

void loop() {
  if(LT_M){
    forward();
  }
  else if(LT_R) {
    right();
    while(LT_R);
  }
  else if(LT_L) {
```

```

left();
while(LT_L);
}
}

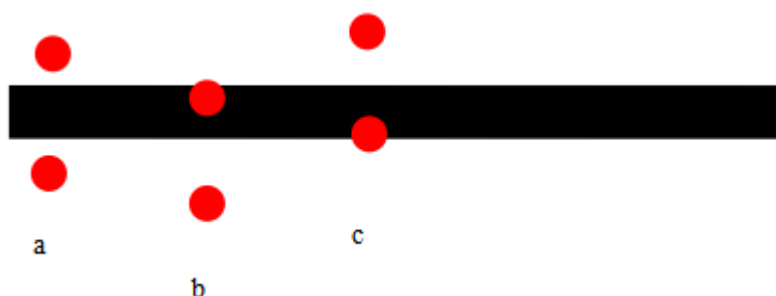
```

車をコンピュータに接続した後、電源スイッチをオンにして走路に乗ることができます。その後、車はラインをたどります。期待どおりに動かない場合は、ライントラッキングモジュールのポテンショメータを調整してください。

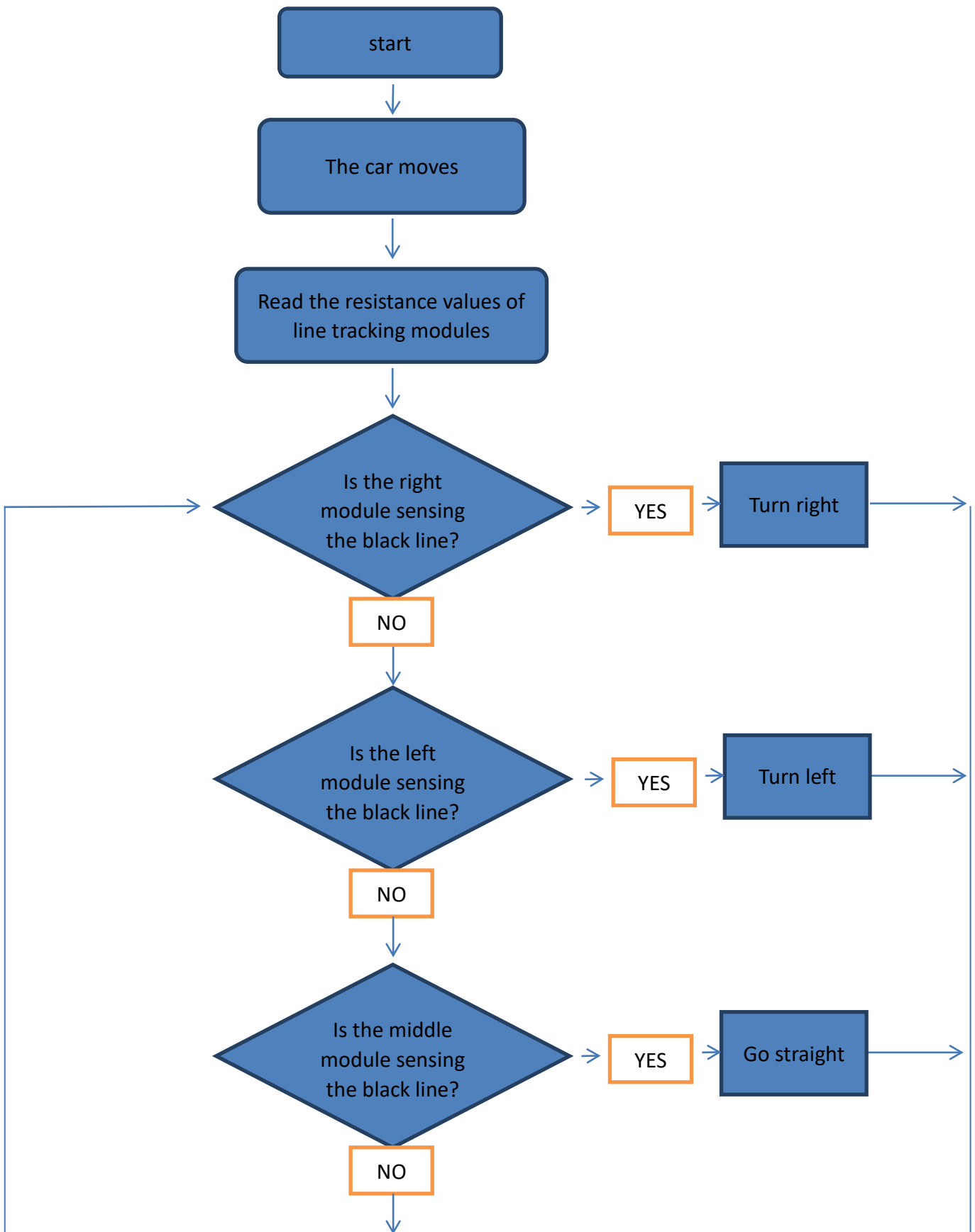
IV. 原則の導入

ライントラッキングモジュール

ライントラッキングセンサは、車の前に並んでいる。ライントラッキングセンサは、赤外線送信管と赤外線受信管で構成されています。前者は赤外線を透過するLEDであり、後者は赤外線を受光するだけの光抵抗である。黒色表面の光反射率は、白色表面の光反射率とは異なる。このため、黒路での車外受信赤外光の強度と白路での反射赤外光の強度が異なり、抵抗量も変化する。直列抵抗間の電圧分割の原理によれば、センサの電圧から車の下の道路の色を推定することによって動作経路を決定することができる。



- a → 車は黒線に沿って動く。ライントラッキングモジュールの1つはラインの左側にあり、もう1つは右側にあります。黒い線を検出することはできません。
- b → 車は右に動くことを学ぶ。左側のモジュールは黒線を検出し、コントローラボードに信号を送り、車は左に曲がります。
- c → 車は左に動くことを学びます。右側のモジュールは黒線を検出し、コントローラボードに信号を送信し、右折します。



上の図から、私達はライン追跡車の原則を見ることができます。車が始動した後、ライン追跡モジュールは、路面上の黒線を検知し、プログラムに従って対応する動作を行うだけでよい。

これは、車線追跡プログラムの簡単なアルゴリズムチャートです。PIDのようなもっと複雑なアルゴリズムがあります。したがって、ライントラッキングの機能を実現させた後、自分で車を制御するアルゴリズムをもっと学ぶことができます。

Small tips

- (1) ラインの曲げ部分はできるだけ滑らかでなければなりません。コーナリング半径が小さすぎると、車は走路を越えて移動する可能性が非常に高くなります。
- (2) ライントラッキングシーンは、白黒テープまたはパスと区別できる任意の色の用紙で作成できます。
- (3) ライントラッキングに加えて、モーションに関係なく走路内に車を閉じ込めるようなライントラッキングの原則に基づいて、想像力を広げて他のプログラムを開発することができます。