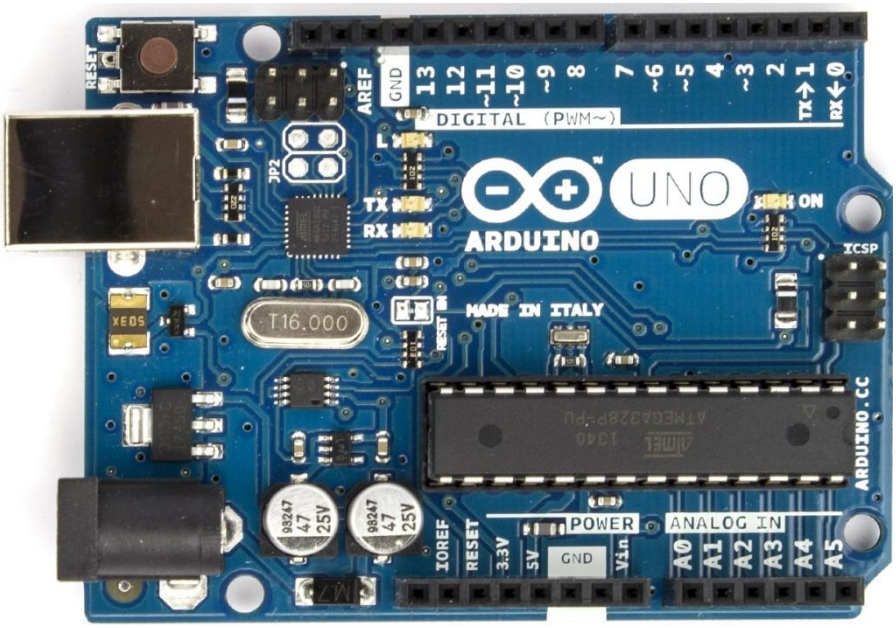
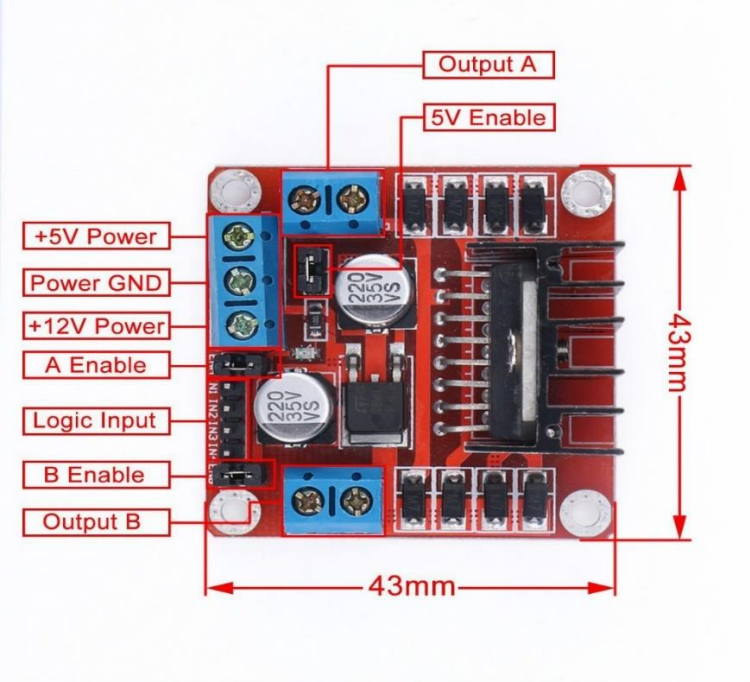
1.Le matériel utiliser pour monter la voiture :

1.1. La carte Arduino :

Pour assurer la réalisation du projet une carte Arduino de type Arduino uno R3 a était mis en place pour manipuler, installer, configurer et programmer la voiture grâce à des multiples instruction et commande.



1.2. Le module L298N :

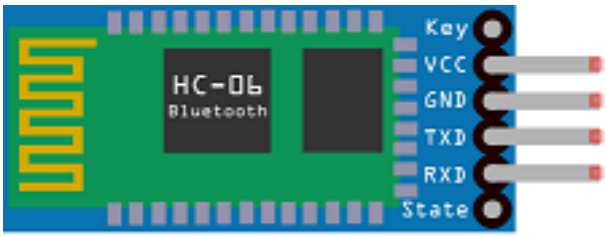


Le module représenter dans la figure précédente est le module L298N il est un module de commande de moteur qui peut être utilisé pour contrôler la vitesse et la direction de deux moteurs à courant continu (DC). Il est couramment utilisé dans les projets de voiture robotique car il peut gérer les exigences de courant élevé des moteurs et dispose d'une protection intégrée contre la surchauffe et les surintensités. Le module peut être contrôlé à l'aide d'un microcontrôleur, tel qu'un Arduino, pour envoyer des signaux PWM pour contrôler la vitesse des moteurs et des signaux numériques pour contrôler la direction de rotation.

1.3. Module de voiture :

Un kit de robot à deux roues pour voiture est une collection de composants qui peuvent être utilisés pour construire un petit robot autonome qui se déplace sur deux roues.

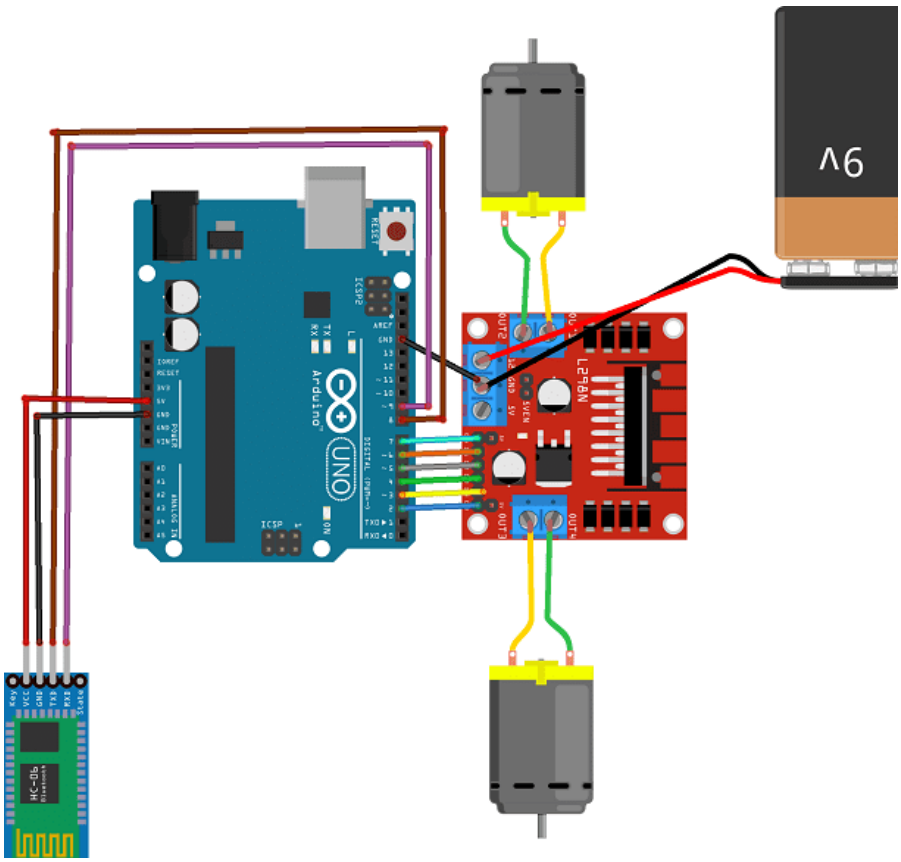
1.4 Le module Bluetooth HC-06:



Le module Bluetooth HC-06 est un choix populaire pour ajouter la fonctionnalité Bluetooth à un robot de voiture basé sur Arduino. C'est un module Bluetooth peu coûteux, à faible consommation d'énergie et facile à utiliser qui permet à une carte Arduino de communiquer sans fil avec d'autres dispositifs compatibles Bluetooth. Le HC-06 peut être configuré pour fonctionner en mode esclave ou maître. En mode esclave, le HC-06 peut se connecter à un dispositif maître tel qu'un ordinateur et recevoir des commandes pour contrôler les mouvements du robot. En mode maître, le HC-06 peut se connecter à d'autres dispositifs esclaves tels que des capteurs et recevoir des données. Le module peut être connecté à la carte Arduino via les broches de communication série (RX, TX) et à l'aide de la bibliothèque SoftwareSerial, il peut être configuré pour utiliser d'autres broches numériques. Le HC-06 est une solution simple et économique pour ajouter une communication sans fil à un robot de voiture basé sur Arduino, permettant le contrôle à distance et la surveillance des mouvements du robot et des données de capteurs.

2. La réalisation et montage :

Maintenant qu'on a le matérielle on peut le monter suivant le schéma suivant :



Grâce au schéma suivant on peut commencer maintenant la phase du codage pour faire rouler la voiture et on peu ajouter après notre module Bluetooth. D'abord pour commencer note code on doit déclarer des variables multiples qui vont contrôler les moteurs de notre voiture comme suit :

```
//moto 1
int enA = 3;
int in1 = 2;
int in2 = 4;
//moto 2
int enB = 6;
int in3 = 5;
int in4 = 7;
```

La broche enA et enB est l'entrée du signal PWM (modulation de largeur d'impulsion) pour le moteur. Elle est utilisée pour contrôler la vitesse du moteur. En fournissant un signal PWM à la broche enA et enB, la vitesse du moteur peut être ajustée en variant le rapport cyclique du signal PWM. La broche In1, in2, in3 et in4 son-les l'entrée de commande de direction pour le moteur. ils sont utilisée pour inverser la direction de rotation du moteur A ou B. En fournissant un signal HIGH ou LOW.

Apres on peut programmer des multiples fonctions comme des fonctions d'avance et de marche arrière :

```
void f(){
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  }
```

```
void b(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);}
```

Et des fonctions pour tourner à gauche et à droite :

```
void r(){
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);}
```

```
void l(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);}
```

Maintenant après avoir écrire nos fonctions on peut tout simplement les faires déclarés dans notre code et les moteurs vont commencer à tourner comme voulu.

3. L'ajout du module Bluetooth :

Pour ajouter le module Bluetooth il suffit d'ajouter des instructions `Serial.begin(9600);` afin d'assurer initialise la communication série pour transmettre et recevoir des données à un taux de transmission de 9600 bits par seconde. La communication série permet à une carte Arduino de communiquer avec un ordinateur ou avec d'autres périphériques tels que des capteurs, des afficheurs, des modules GPS, des modules Bluetooth, etc.

4. L'ajout de détection d'obstacle :

4.1. Le capteur HC-SR04

Le capteur HC-SR04 est un capteur à ultrasons utilisé pour mesurer des distances dans les projets électroniques. Il est largement utilisé dans les projets Arduino pour mesurer des distances jusqu'à 4 mètres avec une précision de 3 mm.

Pour utiliser le capteur il faut déclarer tout d'abords la bibliothèque suivante `#include <NewPing.h>` et grâce à l'instruction suivante `NewPing sonar(A0, A1, 20);` on initialise un objet sonar de type NewPing qui est connecté au capteur à ultrasons en utilisant les broches A0 et A1 pour envoyer et recevoir les signaux d'ultrasons.

4.2 Le capteur de touche :

On a fait appelle encore aux capteurs de touche suivants pour faire allumer des LED lors de touche et ça en le branchons dans notre carte Arduino

5. Le code finale :

```
#include <NewPing.h>
char incomingByte;

//moto 1
int enA = 3;
int in1 = 2;
int in2 = 4;
//moto 2
int enB = 6;
int in3 = 5;
int in4 = 7;

NewPing sonar(A0, A1, 20);

void setup() {
  Serial.begin(9600);
  delay(50);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(12, INPUT);
```

```
  pinMode(13, OUTPUT);
  pinMode(8, OUTPUT);
}
void f(){
  if (sonar.ping_cm() >= 1 && sonar.ping_cm() <5){
    s();
    digitalWrite(8, HIGH);
    delay(200);
    digitalWrite(8, LOW);
    b();
    delay(500);
    s();
  }
  else {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  }}
void b(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  digitalWrite(9, 255);
  digitalWrite(8, HIGH);
  delay(20);
  digitalWrite(9, 0);
  digitalWrite(8, LOW);}
void r(){
  digitalWrite(in1,HIGH );
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);}
void l(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);}
void s(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);}
```

```
void loop() {
  while(Serial.available()){
    incomingByte = Serial.read();
    if(digitalRead(12) == HIGH){
    digitalWrite(13, HIGH);
    digitalWrite(8, HIGH);
    }
    else if (incomingByte == 'F') {
    f();
    digitalWrite(8, LOW);
    digitalWrite(13, LOW);
    }
    else if (incomingByte == 'B') {
    b();
    }
    else if (incomingByte == 'S'){
    s();
    }
    else if (incomingByte == 'L'){
    r();
    }
    else if (incomingByte == 'R'){
    l();
    }
    else if (incomingByte == 'W'){
    digitalWrite(13, HIGH);
    }
    else if(incomingByte == 'w'){
    digitalWrite(13, LOW);
    }
    else if(incomingByte == 'V'){
    digitalWrite(8, HIGH);
    }
    else if(incomingByte == 'v'){
    digitalWrite(8, LOW);
    }
    else if(incomingByte == 'U'){
    digitalWrite(9, 255);
    }
    else if(incomingByte == 'u'){
    digitalWrite(9, 0);
    }
    else if(incomingByte == '0'){
    analogWrite(enA, 0);
    analogWrite(enB, 0);
```

```
    }
    else if (incomingByte == '1' || incomingByte == '2'){
    analogWrite(enA, 75);
    analogWrite(enB, 80);
    }
    else if(incomingByte == '3' || incomingByte == '4'){
    analogWrite(enA, 85);
    analogWrite(enB, 110);
    }
    else if (incomingByte == '5' || incomingByte == '6'){
    analogWrite(enA, 155);
    analogWrite(enB, 160);
    }
    else if(incomingByte == '7' || incomingByte == '8'){
    analogWrite(enA, 195);
    analogWrite(enB, 200);
    }
    else if (incomingByte == '9' || incomingByte == 'q'){
    analogWrite(enA, 255);
    analogWrite(enB, 255);
    }
  }
}
```

Le code suivant va lire une variable nommer incomingByte grâce a une application qui envoie des instructions en Bluetooth et il va la lire, cette dernière vas réagir selon la condition donner. Et il va s'arrêter devant des obstacles grâce à la commande

```
if (sonar.ping_cm() >= 1 && sonar.ping_cm() <5){
                        s();}
```

6. Manipuler la voiture avec des commandes vocaux :

Pour manipuler la voiture grâce a des commandes vocales il suffit de trouver une application comme Arduino BlueControl qui vas envoyer des commandes selon la commande vocale que ta transmis comme exemple si je prononce AVANCE elle l'application vas envoyer la lettre F qui vas selon le code faire avancer la voiture.

7.La détection des obstacles :

Pour que notre voiture réussie a détecter des obstacle on a programmer le code suivant :

```
#include <NewPing.h>
//motor 1
int enA = 3;
int in1 = 2;
int in2 = 4;
//motor 2
int enB = 6;
int in3 = 5;
int in4 = 7;
NewPing sonar(A0, A1, 5);

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(12, INPUT);
}

void av(){
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void ar(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void ga(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
```

```
  digitalWrite(in4, LOW);
}
void ad(){
  digitalWrite(in1,HIGH );
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void st(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void loop() {
  int analogValue = analogRead(A3);
  int speedvalue = map(analogValue, 0, 1023, 0, 255);
  analogWrite(enA, speedvalue);
  analogWrite(enB, speedvalue);
  analogWrite(9, speedvalue);
  int etat = digitalRead(12);
  while(etat == HIGH){
  av();
  if (sonar.ping_cm() >= 1 && sonar.ping_cm() <= 3){
    st();
    delay(1000);
    ar();
    delay(800);
    st();
    analogWrite(enA, 150);
    delay(1000);
    ad();
    delay(600);
    st();
    delay(1000);
    analogWrite(enA, speedvalue);
    if (sonar.ping_cm() >= 1 && sonar.ping_cm() <= 3){
      analogWrite(enB, 150);
      delay(1000);
      ar();
      delay(800);
      st();
      delay(1000);
      ga();
      delay(900);
```

```
      st();
      delay(1000);
      analogWrite(enB, speedvalue);
      st();
      delay(500);
    }
  }
  }
}
```

Dans le code suivant on a ajouté un potentiomètre qui est brancher a la pin A3 et qui vas contrôler la vitesse des moteurs

8. L'ajout d'autre module :

On peut ajouter d'autres module comme le scanner RFID ou une écran LCD mais ca vas tuer notre batterie pour cela on les a ajouter indépendamment.

8.1 Ecran LCD et le scanner RFID

Pour connecter une écran LCD a notre programme il faut tout d'abord télécharger la Bibliothèque LiquidCrystal_I2C est couramment utilisée pour afficher des informations telles que des messages, des numéros, des graphiques et des jauges sur un écran LCD. Et la déclarer avec le format de LCD sans oublions la bibliothèque MFRC522 est une bibliothèque qui s'intéresse à fournir des fonctions et des constantes pour communiquer avec des lecteurs RFID

```
1  #include <SPI.h>
2  #include <MFRC522.h>
3  #include <Wire.h>
4  #include <LiquidCrystal_I2C.h>
5  #include<NewPing.h>
```

La première biblio #include <SPI.h> La bibliothèque SPI fournit des fonctions et des constantes pour communiquer avec des périphériques en utilisant le protocole d'interface périphérique série (SPI). Le protocole SPI est un protocole de communication série synchrone utilisé pour échanger des données entre des microcontrôleurs et des périphériques tels que des capteurs, des écrans et des puces mémoire. La deuxième bibliothèque `#include <MFRC522.h>` est une bibliothèque qui s'intéresse a fourni des fonctions et des constantes pour communiquer avec des lecteurs RFID (Radio Frequency Identification) utilisant le chipset MFRC522. `#include <Wire.h>` est une bibliothèque qui fournit des fonctions pour communiquer avec des périphériques I2C. L'utilisateur accède à un ensemble de fonctions et de méthodes qui peuvent être utilisées pour initialiser et contrôler le LCD. La bibliothèque LiquidCrystal_I2C est couramment utilisée pour afficher des informations telles que des messages, des numéros, des graphiques et des jauges sur un écran LCD. `#include<NewPing.h>` la bibliothèque NewPing est utilisée en programmation Arduino pour contrôler des capteurs à ultrasons

```
void setup() {
  Serial.begin(9600);  // initialize serial communication at 9600 baud
  delay(50);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  analogWrite(enA, 255);
  analogWrite(enB, 255);
  lcd.begin();
  lcd.backlight();
  SPI.begin();    // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  delay(4);
  Serial.println(F("Scan PICC to see UID..."));
  lcd.print("Scan your key ");

  }
```

On a commencé par Serial.begin(9600) qui représente une fonction d'Arduino utilisée pour initialiser la communication entre la carte Arduino et un ordinateur ou un autre périphérique via une connexion série. La fonction définit le début de données en bits par seconde (débit en bauds) pour la transmission de données série. Dans ce cas, elle définit le débit en bauds à 9600 bits par seconde, qui est un débit en bauds couramment utilisé pour la communication série entre les cartes Arduino et d'autres périphériques.

On a déclaré enA et enB ainsi que les quatre autres broches en tant que sorties et avec une fonction analogWrite on a alimenté les deux broches enA et enB en écrivant 255.

lcd.begin() est une fonction d'initialisation pour les écrans LCD (Liquid Crystal Display) compatibles avec les bibliothèques LiquidCrystal ou LiquidCrystal_I2C d'Arduino. Elle est utilisée pour configurer et initialiser l'affichage pour une utilisation avec la carte Arduino. Elle configure les broches d'entrée/sortie utilisées pour communiquer avec l'écran LCD, détermine le nombre de colonnes et de lignes de l'affichage, et initialise l'affichage avec des paramètres tels que la position du curseur, la couleur de fond, etc.

La commande lcd.backlight() est utilisée pour activer ou désactiver le rétroéclairage de l'écran LCD. Cette commande est souvent utilisée pour économiser de l'énergie lorsque le rétroéclairage n'est pas nécessaire ou pour améliorer la lisibilité de l'écran dans des conditions de faible luminosité.

SPI.begin() est une fonction d'initialisation utilisée dans le code Arduino pour configurer l'interface de communication SPI (Serial Peripheral Interface) du microcontrôleur. Elle permet à la carte Arduino de communiquer avec d'autres périphériques compatibles avec le protocole SPI, tels que des capteurs, des écrans ou des cartes SD et mfrc522.PCD_Init() est une méthode de la bibliothèque MFRC522 pour Arduino qui initialise le module de lecteur de cartes MFRC522 pour la communication avec les cartes RFID (Radio-Frequency Identification). `Serial.println(F("Scan PICC to see UID..."));` cette ligne de code est utilisée pour afficher un message sur le moniteur série. Le message affiché est "Scan PICC to see UID..." et la lettre "F" avant le message indique que le message est stocké en mémoire programme plutôt qu'en mémoire vive. Cela permet de libérer de la mémoire vive pour d'autres fonctions du microcontrôleur.

Les deux fonctionnalité suivante pour mieux scanner et lire les donner enregistrer dans notre carte RFID, et les vérifier après grâce au lignes suivantes de code :

```
if(mfrc522.uid.uidByte[0] == UID[0] && mfrc522.uid.uidByte[1] == UID[1] && mfrc522.uid.uidByte[2] == UID[2] && mfrc522.uid.uidByte[3] == UID[3] ){
  lcd.clear();
  lcd.print("Bienvenue a HEM");
  delay(4000);
}else {
  lcd.clear();
  lcd.print("Pas la bonne cle");
  delay(4000);
```

Ici il vas checker si le code fournie par la carte et le même UID fournit au début qui est valable et qui vas lancer le programmes qui vas lire la variable transmit par Bluetooth et exécuter les taches selon la variables partager.