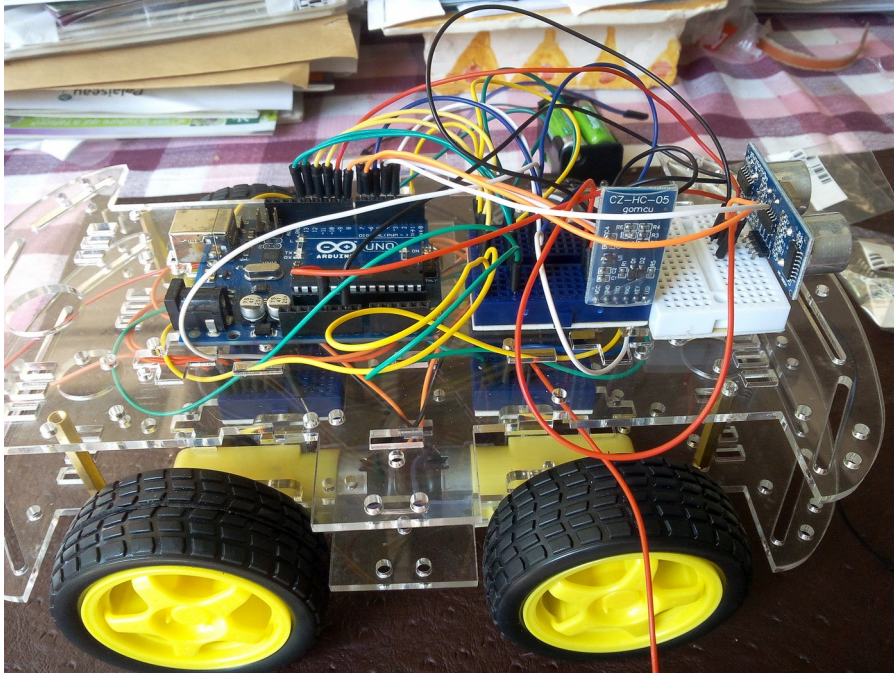


Construire sa voiture bluetooth-commandée avec arduino

I Le resultat final

Voici la voiture que vous allez être sur le point de construire



Mais pour cela il va nous falloir un peu de materiel en voici la liste :

- > un chassis avec 4 moteurs CC*
- > 5 mini breadboards*
- > 2 L293D*
- > un speaker*
- > un module bluetooth HC-05*
- > un module ultrason SR-04*
- > 2 blocs de 4 piles AA*
- > une pile 9V ou une alimentation USB*
- > des fils*
- > un arduino (ici un uno)*

Le code complet de la voiture et des différents composants est disponible ici :

https://github.com/maximouth/voiture_arduino

Mais avant de commencer à tout brancher, nous allons voir comment fonctionnent les différents composants de cette voiture.

II Le bluetooth

Nous allons commencer par voir le fonctionnement du module HC-05 qui est un port serie bluetooth.

Voici à quoi il ressemble :

« photos »

Il possède 6 broches, mais seulement 4 nous seront utiles ici.

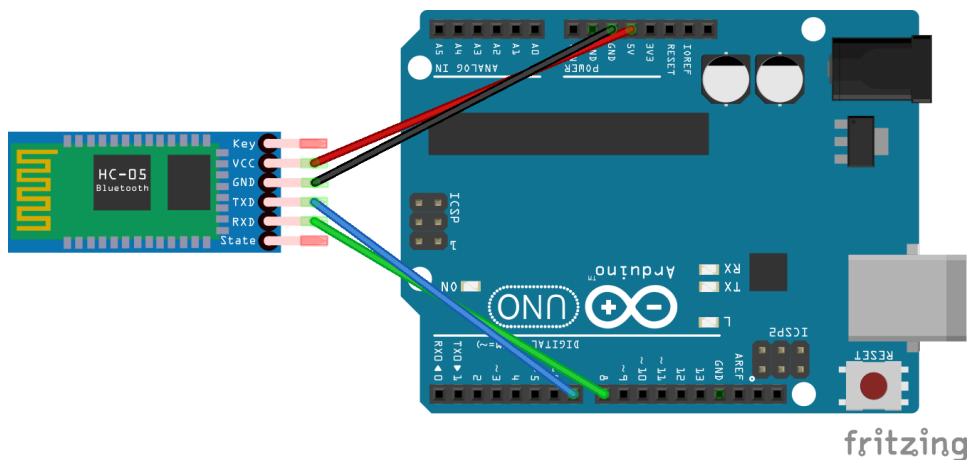
-> VCC, GND correspondent aux bornes positives et negatives de la carte.

-> RX et TX sont les deux broches qui servent à envoyer et recevoir les informations transmissent.

-> Key sert à configurer le module (nom,mdp...) nous ne nous ne servirons pas ici.

-> Led est utiliser pour brancher le module sur une led et ainsi observer les echanges de données, on ne s'en servira pas non plus.

Une fois cela vu nous allons nous attaquer au montage, il est assez simple à réaliser.



VCC sur 5V

GND sur GND

TXD sur la broche 7

RXD sur la broche 8

Nous allons maintenant nous attaquer au code !

Pour ce faire nous utiliserons la librairie « softwareserial » qui va nous permetttre de controler ce module. Ce sera l'unique librairie utilisée lors la création de cettre voiture.

On va commencer par cette ligne à rajouter en haut du fichier

```
#include <SoftwareSerial.h>
```

qui permet d'inclure la librairie citée au dessus.

Et maintenant nous allons créer notre objet qui va nous permettre d'utiliser le HC-05.

```
SoftwareSerial bluetooth_card(8,7);
```

8 pour l'endroit où est branché RXD et 7 pour TXD.

Notre objet btsp est donc créé et va nous permettre de dialoguer entre un objet connecté en bluetooth au HC-05 et notre arduino.

Il faut maintenant initialiser notre objet avec la ligne suivante

```
bluetooth_card.begin(9600);
```

Notre prochain objectif est de lire un caractère envoyé depuis un téléphone et de l'afficher à l'écran grâce au port série de l'arduino.

Le code va donc être maintenant écrit dans la fonction loop()

```
if (bluetooth_card.available()) {  
  data=bluetooth_card.read();  
  Serial.println (data) ;  
  delay (100) ;  
}
```

Le code suivant fait :

Si il y a quelque chose à lire -> le lire , l'afficher et attendre 100 ms

Sinon rien

Nous savons maintenant comment récupérer un caractère envoyé par bluetooth et l'afficher.

Voici le code complet du programme

```
#include <SoftwareSerial.h>
```

```
char data;
```

```
SoftwareSerial bluetooth_card (8,7);
```

```
void setup() {
```

```
  bluetooth_card.begin(9600); // initialise le hc-05
```

```
  Serial.begin(9600); // initialise le port série de l'arduino
```

```
}  
  
void loop() {  
  
    if (bluetooth_card.available()) {  
        data = bluetooth_card.read();  
        Serial.println (data);  
        delay (100);  
    }  
  
}
```

disponible ici :

https://github.com/maximouth/voiture_arduino/tree/master/code/bt

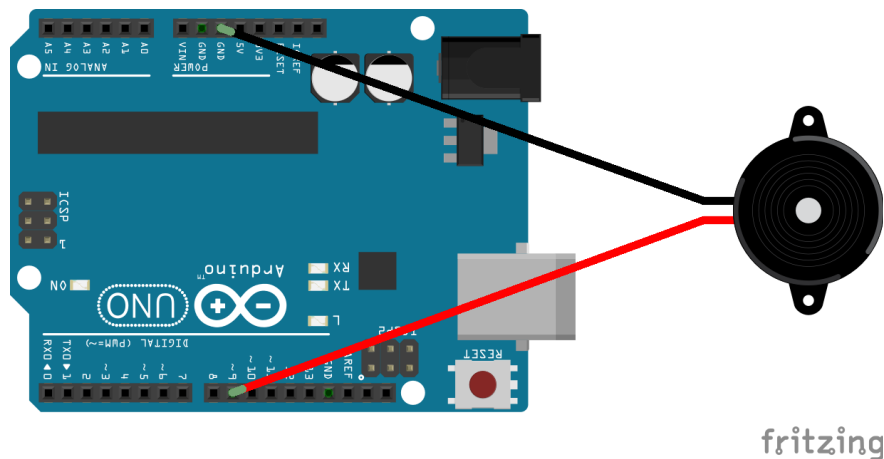
III Le speaker

Dans cette partie nous allons voir comment utiliser un petit speaker, très facile d'utilisation que voici

« photo »

Il ne possède que deux broches, VCC et GND,

Brancher VCC sur la broche 9 de votre arduino et GND sur GND de l'arduino.



Nous allons utiliser le pwm pour generer un signal qui va être transformer en son, en utilisant `analogWrite (9, x)`

-9 car nous avons branché le speaker sur la broche 9

- x une valeur entre 0 et 255, la valeur que vous allez mettre à x changera le son qui sortira, prenez celui qui vous conviens le mieux comme klaxon.

Nous allons tester toutes les sons possible de generer avec notre speaker dans un petit programme.

On va commencer par declarer notre speaker avec cette ligne en haut du fichier :

```
const int spk = 9;
```

puis notre x qui va servir de compteur.

```
int x = 0;
```

nous pouvons maintenant ecrire notre fonction beep() qui servira a produire un son pendant 1 seconde.

```
void beep(){
  analogWrite(spkr,x); // produire un son
  // x va prendre une valeur differente à chaque tour
  delay (1000); // attendre 1 seconde
  analogWrite(spkr,0); // arreter le son
}
```

Mais à quel moment la valeur de x change elle me direz vous ? Et bien dans la fonction loop, on va appeler beep() puis incrementer x(modulo 256 pour le pas dépasser 255) et enfin attendre 2 secondes avant de recommencer.

Voici le code complet :

```
const int spkr = 9;
int x = 0;
```

```
void setup() {
  // declare pin 9 to be an output:
  pinMode(spkr, OUTPUT);
}
```

```
void loop() {
  beep();
  x = (x+1)%256;
  delay (2000);
}
```

```
void beep(){
  analogWrite(spkr,x); // x va prendre une valeur differente à
chaque tour
  delay (1000);
  analogWrite(spkr,0);
}
```

disponible ici :

https://github.com/maximouth/voiture_arduino/tree/master/code/klaxon

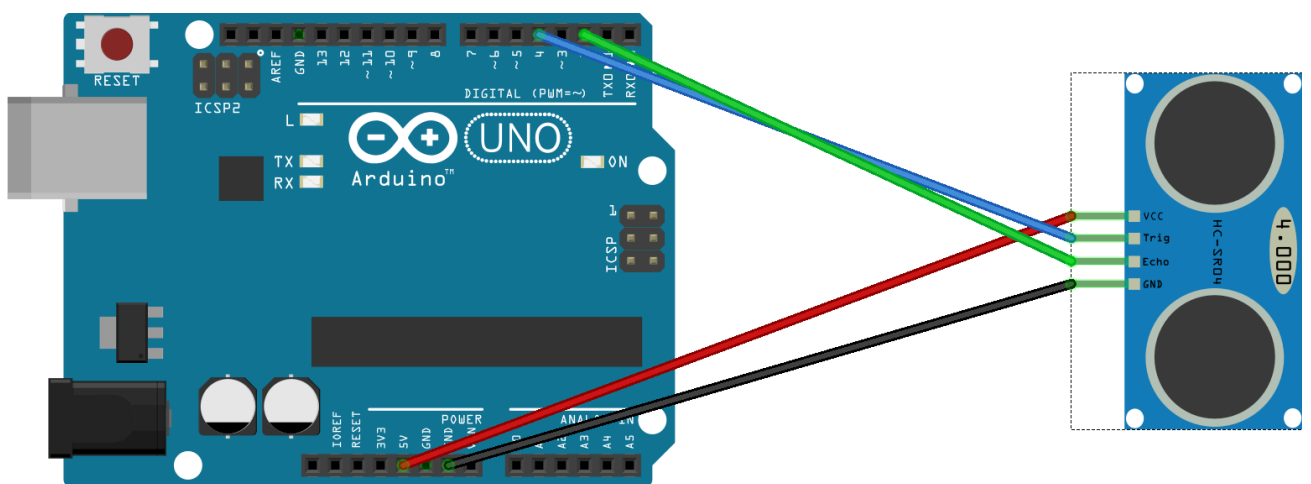
IV Le module ultrason

Nous allons maintenant nous intéresser à notre module ultrason SR-04, et à comment récupérer la distance entre le module et un objet.

Pour commencer voici le module en question :

« photo »

Ce module possède 4 broches, comme d'habitude VCC et GND pour les deux bornes '+' et '-', Trig sert à commander l'émetteur du module et Echo le récepteur. Voici de quel façon nous allons le brancher à notre arduino.



fritzing

Une fois les branchements faits nous allons pouvoir commencer à l'utiliser.

Ouvrez donc un nouveau fichier, et commencer par déclarer les deux broches sur lesquelles sont branchées votre capteur.

```
const int trig = 4;  
const int echo = 2;
```

Et dans la setup() n'oubliez pas de définir si ces broches sont en entrée ou sortie sur votre carte.

Trig étant l'émetteur, sa broche sera en sortie,

```
pinMode(trig, OUTPUT);
```

et Echo l'e receptrer nous auront donc besoin des informations recueillies, sa broche sera en entrée,

```
pinMode(echo, INPUT);
```

REGARDER COMMENT MARCHE EXACTEMENT LE CALCUL DE DISTANCE, ET LA FONCTION PULSE IN() ET POURQUOI ON MET HIGH SUR TRIG PENDANT 10 uS

voir cette page :

<https://itechnofrance.wordpress.com/2013/03/12/utilisation-du-module-ultrason-hc-sr04-avec-larduino/>

V Les Moteurs CC

Voici l'une des parties les plus difficiles, non pas à réaliser, mais à comprendre, faire tourner un moteur n'importe qui en serait capable, mais ici nous allons voir comment régler la puissance du moteur, son sens de rotation, et le plus important pour notre voiture, comment en utiliser plusieurs en même temps pour faire avancer une voiture.

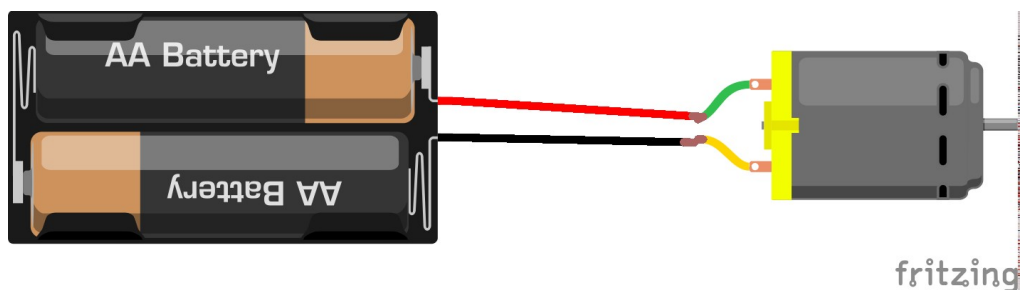
Nous allons avoir besoin dans cette partie d'un peu plus de matériel qu'avant : un moteur, un L293D et notre arduino, avant de commencer nous allons voir le fonctionnement d'un L293D, et de quelle façon l'utiliser avec l'arduino.

Le voici :

« photos »

Un moteur CC ne possède que deux broches, une pour la borne positive l'autre pour la négative d'une alimentation. Branché dans un sens le moteur tournera dans un sens, branché dans l'autre il ira dans le sens inverse.

Vous pouvez tester de cette façon en utilisant un de vos bloc de pile :



et refaire la même chose en inversant les bornes.

Vous voyez ce n'est pas trop difficile jusque là.

Mais comment brancher le moteur avec l'arduino et les piles pour que le moteur ne fasse pas que de tourner même avec une carte non alimentée.

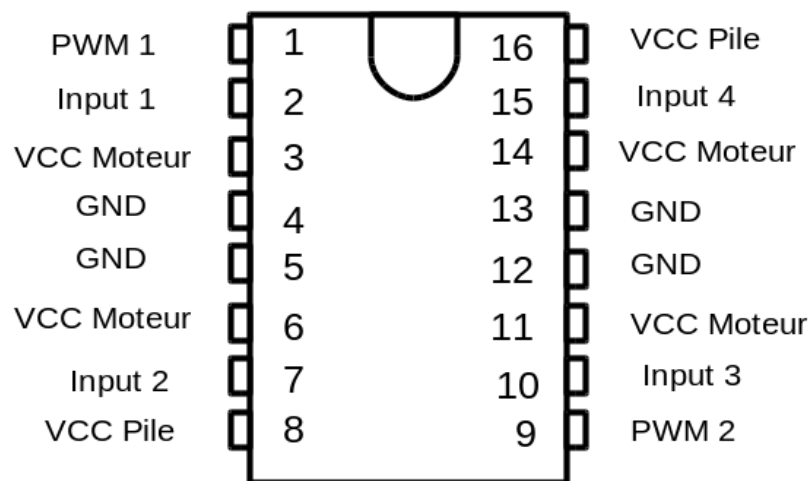
Le L293D me direz vous ? Et bien c'est tout à fait cela !

L'arduino ne pouvant delivrer assez de puissance pour faire fonctionner les moteurs sans risque de griller votre carte, nous sommes obligé d'utiliser un alimentation exterieure ainsi qu'un petit composant pour relier le tout ensemble.

Et le voici sa fichetechnique pour connaitre les tensions maximales du composant ainsi que ces caracteristiques techniques :

<http://www.ti.com/lit/ds/symlink/l293.pdf>

ainsi que la façon de le brancher pour nous, Nous utiliserons un coté du L293D par moteur.



Nous allons maintenant expliquer comment se servir de chaque entrées.

Le PWM va nous servir à contrôler la vitesse à laquelle tournera le moteur, cela va marcher un peu de la même façon qu'avec notre speaker, nous allons utiliser la fonction `analogWrite(pin PWM, vitesse)`.

VCC pile sert à brancher la borne plus de votre alimentation dessus et les différentes sorties GND iront sur la borne moins, éviter d'inverser ces deux bornes sur votre L293D par risque de faire enflammer le composant (si si c'est possible), et peut être votre voiture si vous ne vous en rendez pas compte.

(bon le composant brûlé à été vécu, mais la voiture n'a subi aucun dégats, juste une breadboard qui a un peu fondu mais rien de grave)

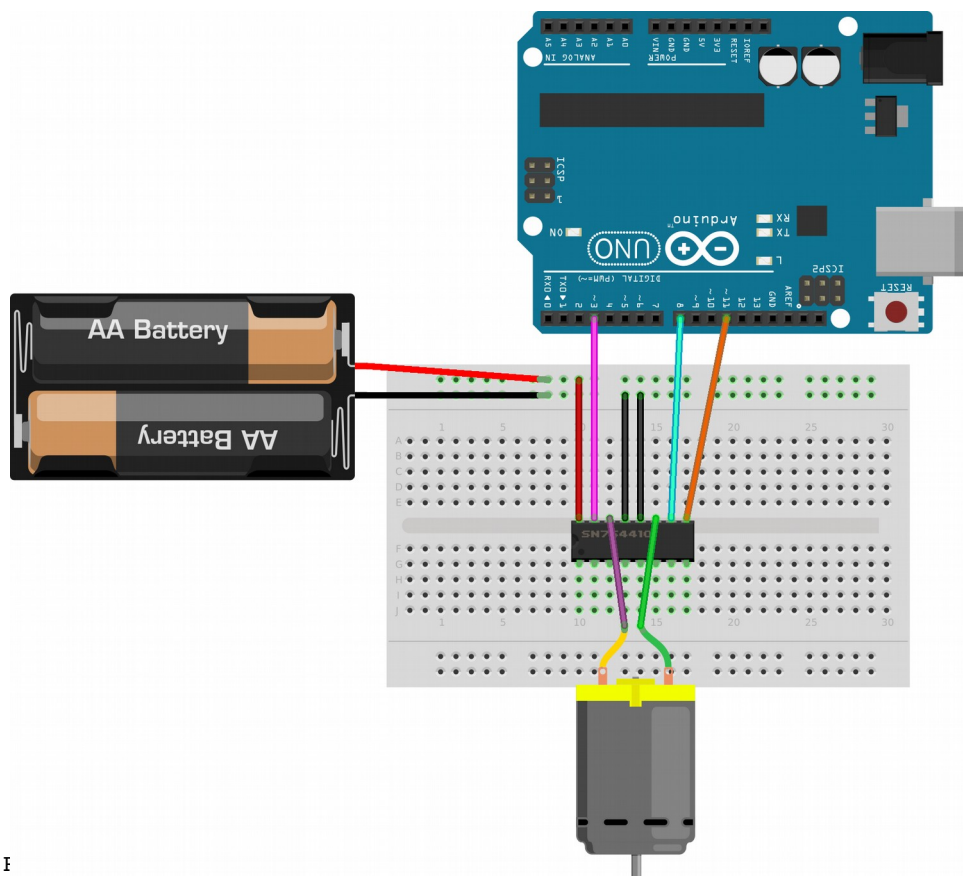
Les broches 3-6 et 11-14 servent à brancher les moteurs dessus, pas de sens précis, essayer juste de tous les brancher dans le même sens si vous en utilisez plusieurs cela évitera d'avoir à trop jouer avec les entrée 2-7 et 10-15

Nous allons donc voir de quelle façon nous servir de ces 4 broches, elles servent à contrôler le moteur, c'est à dire le faire tourner dans un sens, dans l'autre ou le stopper.

Pour cela nous allons utiliser la fonction `digitalWrite(num Input, HIGH/LOW)` qui selon les différentes que peuvent prendre les deux inputs 2 à deux va faire quelque chose, voici les différentes possibilités (1 -> HIGH) (0 -> LOW)

Input 1 (2-10)	Input 2 (7-15)	Effet
0	0	Frein
0	1	Tourne sens des aiguilles d'une montre
1	0	Tourne dans le sens inverse
1	1	Frein

Nous allons maintenant pouvoir réaliser ce montage et comprendre de quelle façon l'utiliser.



Nous allons faire augmenter puis diminuer la vitesse de rotation du moteur sur toutes les valeurs possibles dans ce nouvel exemple que nous écrirons dans un fichier motor.ino.

Pour commencer, nous allons comme d'habitude déclarer nos différentes broches :

```
const int input1 = 8;  
const int input2 = 3;  
const int pwm = 11;
```

car la broche 8 de notre arduino est branchée sur input1, la 3 sur input2, et la 11 qui permet l'utilisation du pwm en 11.

Il va aussi nous falloir deux variables, une pour la valeur qui l'on va donner à la fonction analogWrite() et une autre pour indiquer si on est entrain d'augmenter ou de réduire la vitesse de notre moteur.

```
int x = 0;  
int sens = 0; // 0 pour incrementer - 1 pour decrementer
```

Dans setup() nous allons devoir déclarer nos 3 broches comme broches de sortie.

```
pinMode(input1, OUTPUT);  
pinMode(input2, OUTPUT);  
pinMode(pwm, OUTPUT);
```

Une fois ces déclarations faites, nous allons nous attaquer au gros de la fonction à l'intérieur de loop.

Voici le principe de notre fonction.

À chaque tour de boucle nous allons écrire la nouvelle vitesse à laquelle va tourner notre moteur de cette façon analogWrite (pwm,x); , puis nous occuper de changer la valeur de ce x.

Si sens est égal à 0, c'est que l'on est entrain de monter la vitesse, du coup on augmente x de 1, on vérifie si notre x n'a pas dépassé 255 (valeur max) sinon on inverse le sens en le mettant à un et on modifie x pour lui donner 255 comme valeur. Puis attendre 500 ms avant de recommencer.

Si sens est égal à 1, on fait le symétrique de si sens est égal à 0.

Voici le code complet :

```
const int input1 = 8;
const int input2 = 3;
const int pwm = 11;

int x = 0;
int sens = 0; // 0 pour incrementer - 1 pour decrementer

void setup() {

    pinMode(input1, OUTPUT);
    pinMode(input2, OUTPUT);
    pinMode(pwm, OUTPUT);

}

void loop() {

    analogWrite (pwm,x);
    if (sens = 0) {
        x++;
        if (x > 255) {
            sens = 1;
            x = 255;
        }
    }
    else {
        x--;
        if (x < 1) {
            sens = 0;
            x = 1;
        }
    }

    delay (500);

}
```

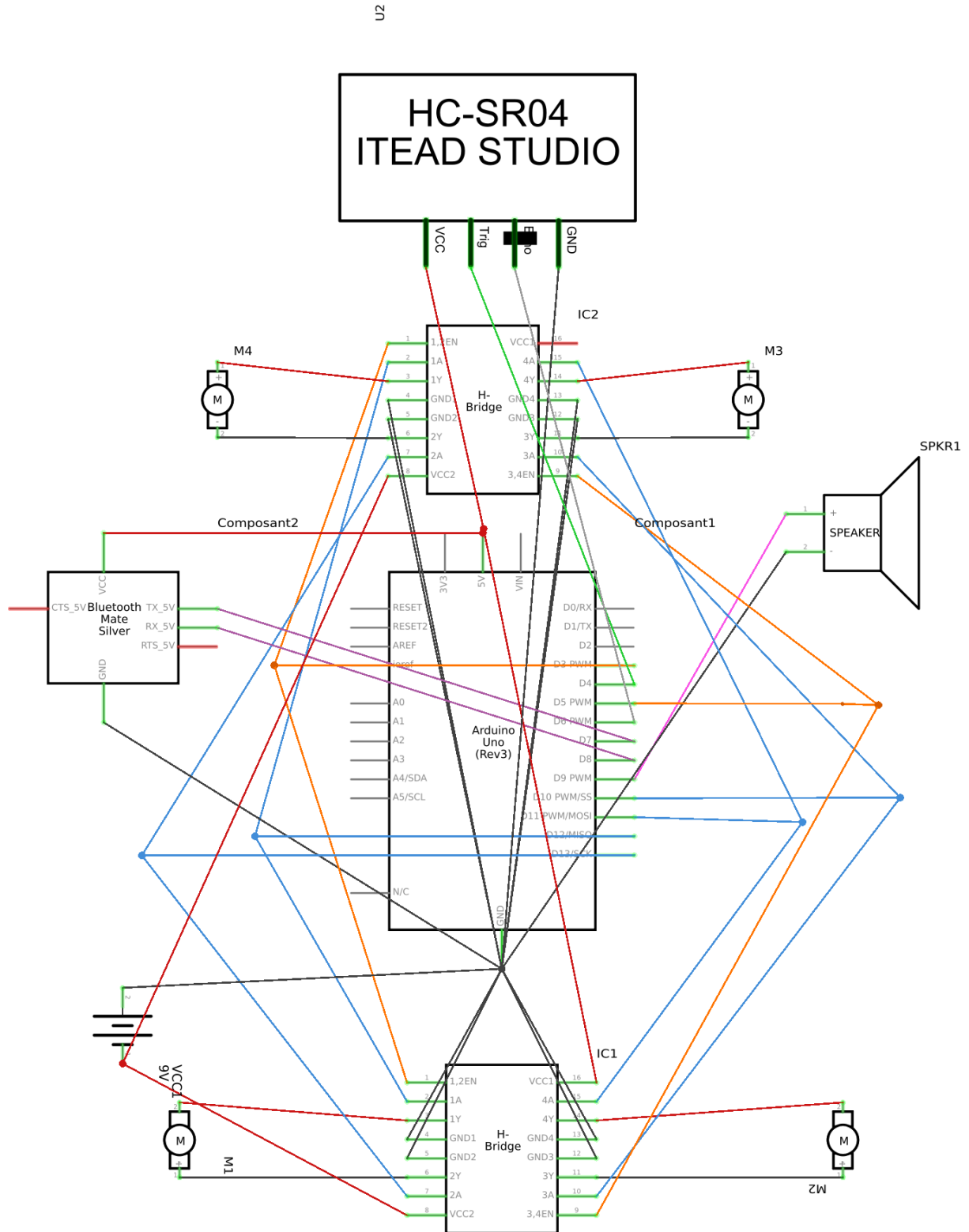
disponible ici :

https://github.com/maximouth/voiture_arduino/tree/master/code/motor

VI Le montage final

Nous venons de voir de quelle façon ce servir des différents composants de notre voiture, nous allons donc passer à l'étape que vous attendez tous, le montage de la voiture !

Il va falloir tout assembler ensemble maintenant ! Et voici de quelle façon :



un pdf du montage est disponible ici :

https://github.com/maximouth/voiture_arduino/blob/master/voiture_telecommande_4X4_sch%C3%A9ma.pdf

Vous trouvez ça compliqué ?

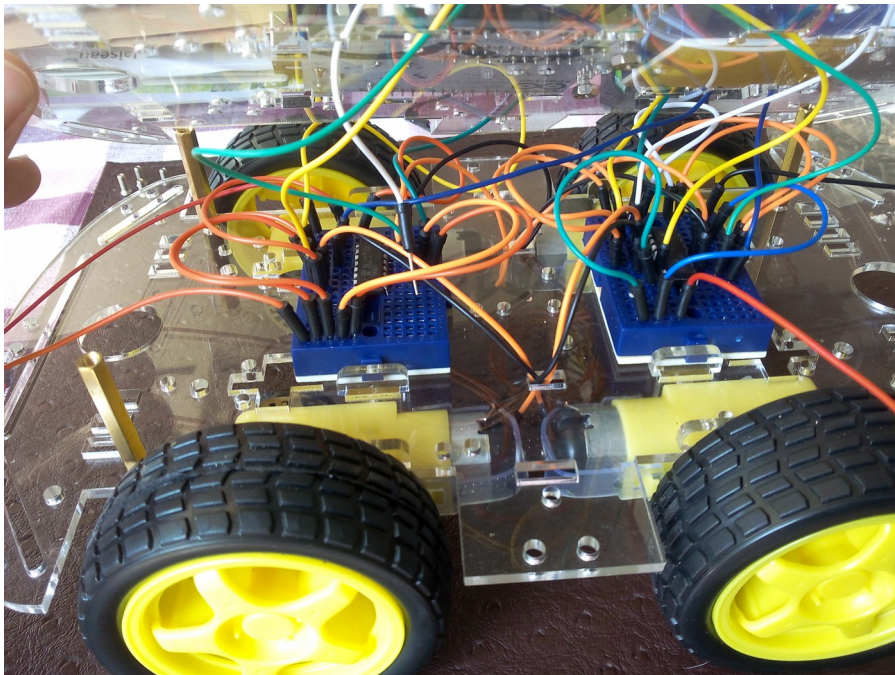
Ne vous inquiétez pas on va reprendre cela étape par étape, c'est assez simple si vous avez suivi toutes les parties précédentes.

Toute cette partie de montage sera basé sur ce châssis :

<http://www.banggood.com/4WD-Smart-Robot-Car-Chassis-Kits-With-Strong-Magneto-Speed-Encoder-p-917007.html>

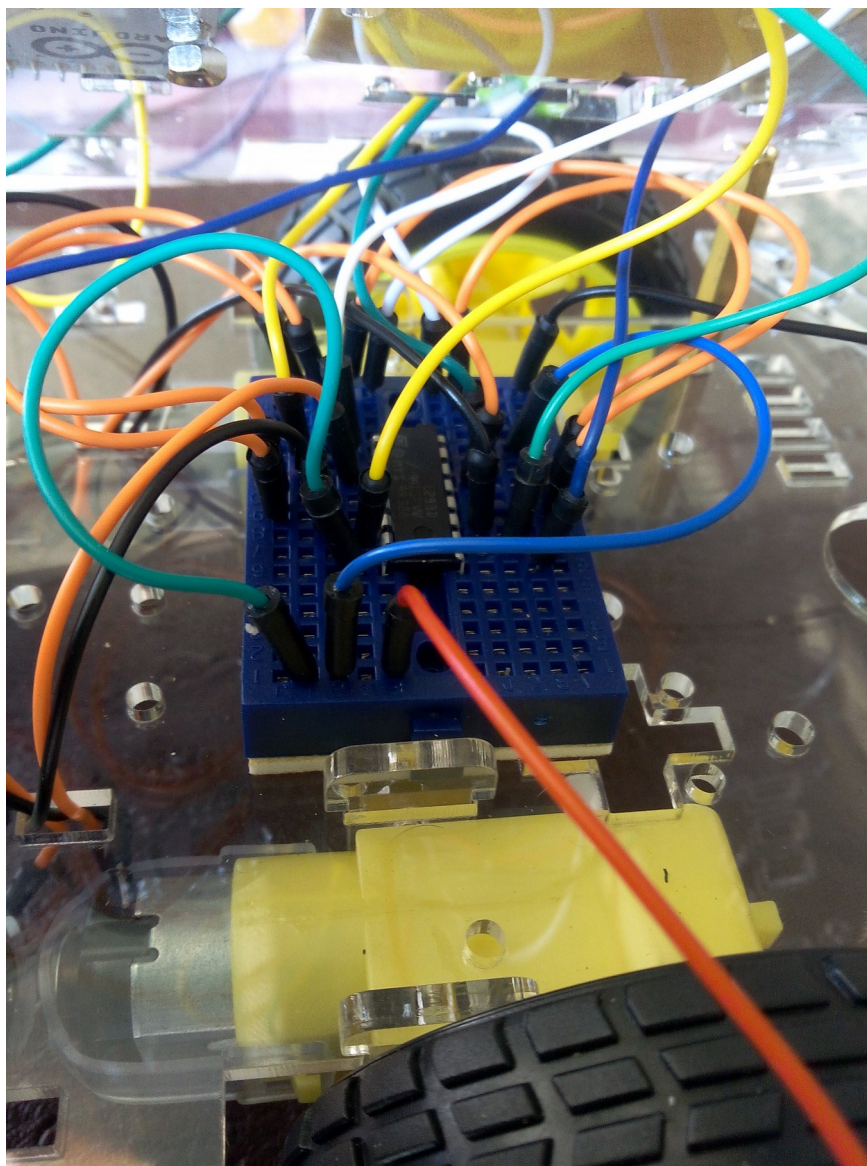
, si vous en avez un autre adaptez mon montage à votre matériel.

Une fois les moteurs fixés sur le premier support, collez deux mini breadboard de cette façon :



Et commencez à relier les moteurs comme vu dans la partie précédente et dans le schéma juste au dessus, pour l'instant laissez les fils devant être reliés à l'Arduino, et ne les branchez que sur le L293D.

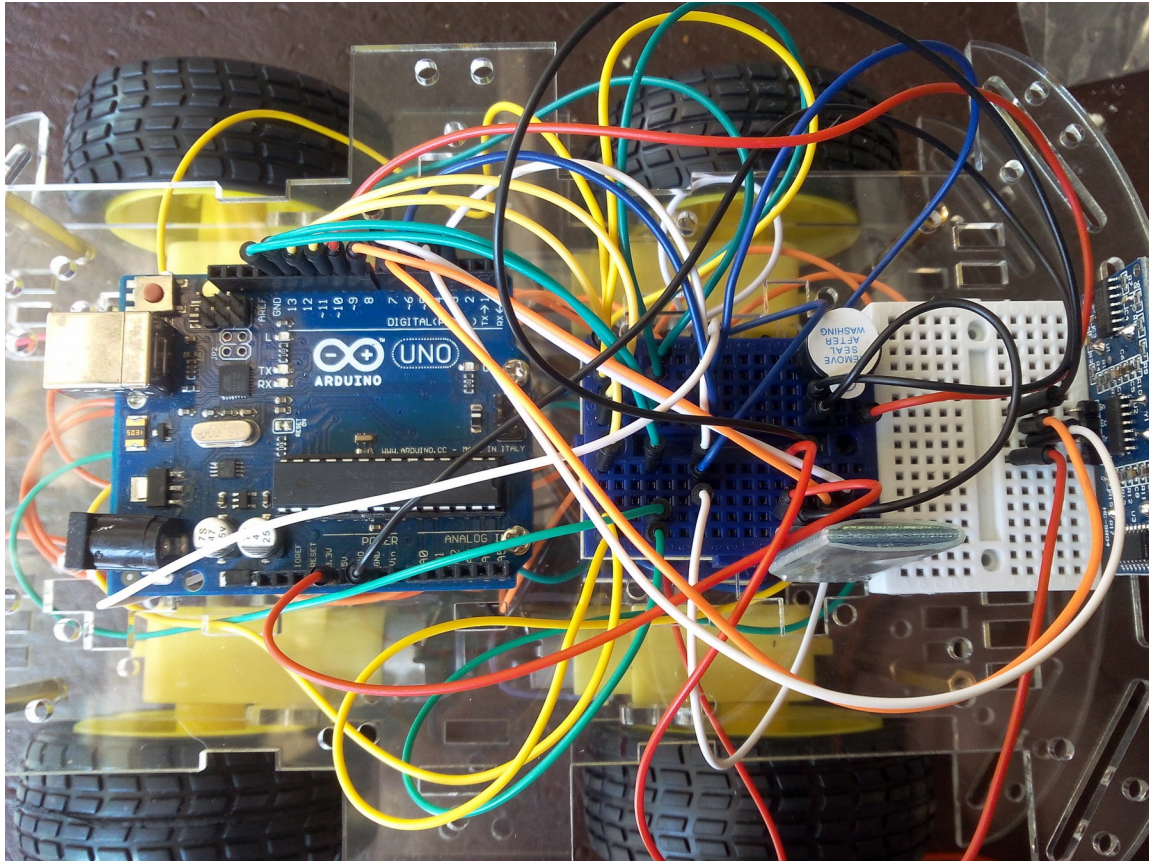
Vous devriez obtenir quelque chose de ressemblant à ça.



Une fois ceci fait, laissez la partie basse pour le moment et prenez celle du haut.

Sur la plaque la plus haute, vous pouvez fixer votre arduino a l'aide des vis fournies, puis collez 2 breadboards de cette façon pour pouvoir y mettre le module bluetooth, le module ultrason et le speaker, ainsi que les différents branchements que vous allez centraliser en haut.

Voici une façon de faire



Vous pouvez donc voir l'arduino à gauche, sur la breadboard bleue des branchements, le speaker et le module bluetooth, et enfin sur la blanche, le module ultrason. Une fois les branchements faits à cet étage toujours en suivant le schéma vous allez pouvoir superposer les deux étages et faire passer les différents fils de l'étage inférieur au supérieur par les petites fentes, et finir le montage.

Voilà vous avez une voiture montée comme sur la première image de ce tutoriel, mais vous ne pouvez pas encore faire grand chose avec, c'est ce que nous allons voir maintenant.

Nous voulons une voiture qui avance, recule, tourne à droite et à gauche, klaxon sans bloquer l'entrée de nouvelles commandes, et s'arrete si il y a un obstacle devant elle à moins d'une distance donnée.

Pour commencer nous ferons juste rouler la voiture dans les 4 directions, une fois cela fait nous rajouterons le klaxon et enfin une fois tout cela en place nous pourrons arreter la voiture avec le capteur.

Il faut commencer par declarer nos différentes entrée et sortie, pour l'instant nous n'aurons que les 4 inputs et les 2 pwm.

Les moteurs sont reliés entre eux 2 par 2 (gauche et droite)

```
// pwm moteurs gauches
const int mt_g = 3;
//pwm moteurs droits
const int mt_d = 5;

//sens des moteurs gauches
const int b2 = 12;
const int b7 = 13;

//sens des moteurs droits
const int b10 = 10;
const int b15 = 11;
```

Nous allons s'inspirer de la partie sur le module bluetooth qui nous permet de recuperer le char qui est envoyé par le téléphone.

```
if (bluetooth_card.available()) {
    dir = bluetooth_card.read();
```

va nous permettre de recuperer le caractère envoyé, nous allons ensuite faire un gros switch qui nous permettra de réaliser une action selon les différentes entrée.

```
switch (dir) {
```

Ici nous avons choisi comme codage :

u -> pour avancer

d -> pour reculer

r -> tourner a droite

l -> tourner a gauche

K -> klaxonner

b -> arreter la voiture

Du coup le switch donnera quelque chose de ce style la :

```
switch (dir) {

    //arreter la voiture
case 'b' :
    arreter() ;
    break;

    //faire avancer la voiture
case 'u' :
    avancer() ;
    break;

    //tourner Ã gauche
case 'l' :
    gauche() ;
    break;

    //tourner Ã droite
case 'r' :
    droite() ;
    break ;

    //reculer
case 'd' :
    reculer() ;
    break;

    //klaxon
case 'k' :
    klaxonner() ;
    break;

default :
    // ne rien faire si autre
    break;
}
```

Mais comment faire pour faire bouger ou tourner la voiture ?

Il va falloir reprendre le tableau sur les moteurs donné dans la section moteur, pour avancer il faut que Input1 soit à LOW et input2 à HIGH.

Reculer l'inverse.

La où l'astuce ce passe c'est pour tourner, le chassis etant fixe et n'ayant pas de bras pour faire bouger les deux roues avant nous allons devoir faire autrement.

C'est à dire, diminuer un peu la vitesse des moteurs du coté ou l'on veut tourner et augmenter legerement celle des moteurs oposés, la voiture tournera donc.

Mais nous commençons à parler de vitesse, nous allons du coup introduire de nouvelles constantes pour avoir des vitesses fixes :

```
#define STOP 0
#define MARCHE 127
#define TURN 50
```

qui nous servirons de valeurs pour le pwm.

Voyons comment ecrire nos différentes fonctions définies au dessus.

```
Void Arreter() {
    analogWrite(mt_g, STOP);
    analogWrite(mt_d, STOP);

    digitalWrite (b2, LOW);
    digitalWrite (b7, LOW);
    digitalWrite (b10, LOW);
    digitalWrite (b15, LOW);
}
```

on réduit la vitesse des 4 moteurs à 0 avec analogWrite() pour on met les moteurs en positions frein en mettant input1 et input2 à LOW.

Vous pouvez maintenant arreter votre voiture, mais cela ne sert pas à grand chose si elle ne peux pas avancer.

Pour la faire avancer il faut avoir respectivement LOW et HIGH sur input1 et input2, ainsi qu'une certaine vitesse donnée par la constante AVANCE.

```
Void avancer() {  
    digitalWrite (b2, LOW);  
    digitalWrite (b7, HIGH);  
    digitalWrite (b10, HIGH);  
    digitalWrite (b15, LOW);  
  
    analogWrite(mt_g,MARCHE);  
    analogWrite(mt_d,MARCHE);  
}
```

Pour reculer il faut juste inverser les HIGH et LOW.

Votre voiture peut maintenant avancer et reculer c'est génial ! Et si on la faisait tourner ?

Pour cela il faut comme dit précédemment diminuer la vitesse du coté ou l'on veut tourner et augmenter celle de l'autre. Mais il y a une contrainte supplémentaire, on ne sait pas si la voiture avance, recule ou est à l'arrêt, du coup nous n'allons que modifier les différentes vitesses, on supposera que l'utilisateur sera déjà entrain de faire une action.

Comme sur une vrai voiture si l'on veut, on peut tourner le volant mais il ne se passera rien tant que l'on aura pas passé une vitesse et appuyé sur l'accélérateur.

```
Void gauche() {  
    analogWrite(mt_g,MARCHE+80);  
    analogWrite(mt_d,TURN-10);  
}
```

et de la même façon pour tourner à droite :

```
void droite() {  
    analogWrite(mt_d,MARCHE+80);  
    analogWrite(mt_g,TURN-10);  
}
```

Voilà vous pouvez maintenant contrôler votre voiture à votre guise !

Mais vous avez envie de vous faire remarquer en roulant dans votre belle auto, pour interpeller les gens et titiller leur jalousie.

La partie la plus importante de cette voiture est sur le point d'être réalisée, le klaxon !

Nous allons pour cela réutiliser le code vu pour utiliser le speaker dans la partie qui lui est consacrée, mais un problème se pose avec le code écrit précédemment, si on utilise le `delay()` pendant temps passé en argument, on ne peut plus rien faire, c'est à dire changer de direction, arrêter la voiture ou autre, ce qui n'est pas très pratique à vrai dire.

Nous allons donc utiliser des « stamps », qui vont nous permettre de laisser le programme se dérouler sans interruptions, et d'arrêter le klaxon à une durée fixée.

Pour ce faire, nous avons besoin de deux variables capable de stocker un temps, qui sera récupéré plus tard grâce à la fonction `millis()`.

```
unsigned long time1;  
unsigned long time2;
```

Quand un 'K' est entré, il va falloir allumer le klaxon de cette façon :

```
analogWrite(9,KLX);
```

puis récupérer le temps auquel cette action a débutée

```
time1 = millis();
```

Maintenant que nous avons un temps de départ, il va falloir lancer un test à chaque tour de boucle `loop()` qui vérifiera si un certain temps s'est écoulé, et arrêtera le klaxon.

```
if (((time2 = millis()) - time1) > 1000) {  
    analogWrite(9,0);  
}
```

Nous arrivons presque à la fin du code de cette voiture, il faut maintenant lui permettre de détecter si un obstacle est présent devant elle et la stopper.

Voilà de quelle façon nous allons procéder pour arrêter notre voiture :

-> regarder la distance de l'objet le plus proche devant la voiture, et la stocker

-> tester si la distance est plus petite ou non que la distance minimale à laquelle on peut s'approcher.

-> Si oui,

couper tout les moteurs

-> Sinon,

faire ce que l'on fait d'habitude

Pour regarder la distance, nous allons reprendre le code sur le module ultrason.

```
digitalWrite(trig, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(trig, LOW);
```

```
duration = pulseIn(echo, HIGH);  
distance = (duration/2)/29;
```

Puis un if pour tester la condition et savoir quoi faire.

```
if (distance <= 30) {  
  arreter la voiture  
}  
else {  
  le code qui etait la au debut  
}
```

Voilà votre voiture est maintenant fonctionnelle !

Je vais vous remettre le code complet ainsi que son lien plus bas, si vous avez des améliorations à proposer ou des questions, vous pouvez les poser ici :

maximouth@gmail.com

```

#include <SoftwareSerial.h>

#define STOP 0
#define MARCHE 127
#define TURN 50
#define KLX 110

SoftwareSerial bluetooth_card (8,7);

// ultrason
const int trig = 4;
const int echo = 6;

//klaxon
const int klaxon = 9;

// pwm moteurs gauches
const int mt_g = 3;
//pwm moteurs droits
const int mt_d = 5;

//sens des moteurs gauches
const int b2 = 12;
const int b7 = 13;

//sens des moteurs droits
const int b10 = 10;
const int b15 = 11;

/*
bluetooth_card.available(); // retourne le nombre de caractères à
lire
bluetooth_card.read();      // retourne le prochain caractère
reçu
bluetooth_card.write(val);  // envoie le char "val" sur la voie
série
*/

void setup() {
    bluetooth_card.begin(9600); // démarre la voie série à la
vitesse speed
    // Serial.begin(9600);

    pinMode(klaxon, OUTPUT);

```



```

pinMode(trig, OUTPUT);
pinMode(echo, INPUT);

pinMode(mt_g, OUTPUT);
pinMode(mt_d, OUTPUT);

pinMode(b2, OUTPUT);
pinMode(b7, OUTPUT);
pinMode(b10, OUTPUT);
pinMode(b15, OUTPUT);

digitalWrite (b2, LOW);
digitalWrite (b7, LOW);
digitalWrite (b10, LOW);
digitalWrite (b15, LOW);

}

unsigned long time1;
unsigned long time2;

int distance;
int duration;
int flag = 0;
char dir;

void loop() {

    if (((time2 = millis()) - time1) > 1000) {
        analogWrite(9,0);
    }

    //regarder la distance de l'objet devant la voiture à chaque
    tour
        digitalWrite(trig, HIGH);
        delayMicroseconds(10);
        digitalWrite(trig, LOW);

        duration = pulseIn(echo, HIGH);
        distance = (duration/2)/29;
        // pulseIn(echo, LOW);

        // si l'objet est à moins de 10 cm flag = 1
        if (distance <= 30) {
            // Serial.println ("arret");

            analogWrite(mt_g,STOP);

```

```

    analogWrite(mt_d,STOP);

    digitalWrite (b2, LOW);
    digitalWrite (b7, LOW);
    digitalWrite (b10, LOW);
    digitalWrite (b15, LOW);

}
else
{
    // if (Serial.available()) {
    if (bluetooth_card.available()) {
    //     dir = Serial.read();

    dir = bluetooth_card.read();

    switch (dir) {

        //arreter la voiture
    case 'b' :
        // Serial.println ("arret");

        analogWrite(mt_g,STOP);
        analogWrite(mt_d,STOP);

        digitalWrite (b2, LOW);
        digitalWrite (b7, LOW);
        digitalWrite (b10, LOW);
        digitalWrite (b15, LOW);

        break;

        //faire avancer la voiture
    case 'u' :
        //Serial.println ("avance");

        digitalWrite (b2, LOW);
        digitalWrite (b7, HIGH);
        digitalWrite (b10, HIGH);
        digitalWrite (b15, LOW);

        analogWrite(mt_g,MARCHE);
        analogWrite(mt_d,MARCHE);

        break;

        //tourner Å gauche
    case 'l' :

```

```

        //Serial.println ("droite");

        analogWrite(mt_g,MARCHE+80);
        analogWrite(mt_d,TURN-10);

        break;

        //tourner Ã droite
    case 'r' :
        //Serial.println ("gauche");

        analogWrite(mt_d,MARCHE+80);
        analogWrite(mt_g,TURN-10);

        break;

        //reculer
    case 'd' :
        //Serial.println ("reculer");

        digitalWrite (b2, HIGH);
        digitalWrite (b7, LOW);
        digitalWrite (b10, LOW);
        digitalWrite (b15, HIGH);

        analogWrite(mt_d,MARCHE);
        analogWrite(mt_g,MARCHE);

        break;

        //klaxon
    case 'k' :
        //Serial.println ("klaxon");
        analogWrite(9,KLX);
        time1 = millis();
        break;

    default :
        //Serial.println ("mauvais");
        break;

    }

    }

}

```

téléchargeable ici :

https://github.com/maximouth/voiture_arduino/tree/master/voiture

VII code, video, améliorations possibles, sources

Voici une petite video de la voiture en pleine action :



visible ici :

Pour recuperer une application pour communiquer entre le telephone et le module bluetooth :

allez sur le store de votre téléphone et cherchez « bluetooth spp » vous devrez tomber sur quelques application gratuite qui permettent cette comunication.

Ou si vous êtes assez courageux, pourquoi pas essayer d'en refaire une vous même.