ArduiCar

SOMMAIRE

Contenu

Arduicar

Description : ayant eu l'opportunité de récupérer une voiture radiocommandée NIKKO Phoenix, j'ai trouvé des infos sur ses caractéristiques sur le site http://nikkoevolution.fr/voiture-modelisme/839/phoenix-dragon.html.

Fiche Descriptive Nikko Phoenix

Modèle	Phoenix / Dragon - Réf : 120141A
Epoque	Fin 2004
Prix	99€ en france, environ 275€ au Royaume-Uni ! (69 € en promo, exceptionnellement)
Echelle	1/12
Dimensions, poids	50/25/18 cm. 2000 gr en ordre de marche dont batterie 300 gr
Couleurs	Jaune et blanche (Phoenix) ou bleue et blanche (Dragon)
Vitesse	28 km/h (selon constructeur)
Moteur	Mabushi 540
Batterie	7,2 volts Nicd 1600 mAh à prise Tamiya
Télécommande	27 mhz, 2 voies proportionnelles. Emetteur: 1 pile 9v. Quartz amovible.
Transmission	2 roues motrices arrières, 1 différentiel.
Roues	Roues amovibles, fixation par écrou, plus larges à l'arrière. Dimension de pneus à l'avant : 93/45/wh59. Dimensions de pneus à l'arrière : 110/60/wh70
Suspension	4 amortisseurs à ressort, inamovibles. Roues indépendantes à l'avant, pont rigide à l'arrière.

L'exemplaire récupéré (dans une déchetterie…) est sans télécommande. La carrosserie est absente et un des supports est cassé. La batterie est absente également. La jante arrière droite et l'aileron sont « explosés ». La voiture a dû être conduite par un « fou du volant »… Heureusement le site nikkoevolution.fr dispose d'un espace « pièces d'occasion » où j'ai trouvé les pièces cassées (jante et aileron).

D'après les forums la récupération d'une voiture RC est souvent le point de départ de ce genre d'aventure… Ce qui se vérifie à nouveau ici.

La télécommande

Le circuit principal

Pour la télécommande j'avais déjà une platine Arduino Duemillanove et je me suis dit que le plus simple était de générer les infos de commande avec un Arduino et de les transmettre à l'autre Arduino, chargé de gérer la voiture.

Les boutons de commande

Le site DFrobot propose un joystick (13€) qui se connecte sur la carte Arduino :
(http://stores.ebay.fr/dfrobotopensourcehardwarerobot)

Le joystick est équipé de deux potentiomètres et d'un contact. La carte offre deux poussoirs supplémentaires qui pourront servir pour des accessoires (feux, sirène ?). De plus ce joystick est prévu pour accueillir une carte UHF APC220 d'une portée en extérieur de 1000 mètres.

La liaison radio

La liaison radio entre deux Arduino peut se réaliser à partir de nombreuses solutions (Xbee, ZigBee, Walkthrough, APC220…) pour des raisons de coût (il faut un shield Xbee + un module XBee de chaque côté !) et de facilité de mise en œuvre (le port APC220 est prévu sur le joystick) j'ai choisi les modules APC220 (36€). (http://stores.ebay.fr/bestchoose702)

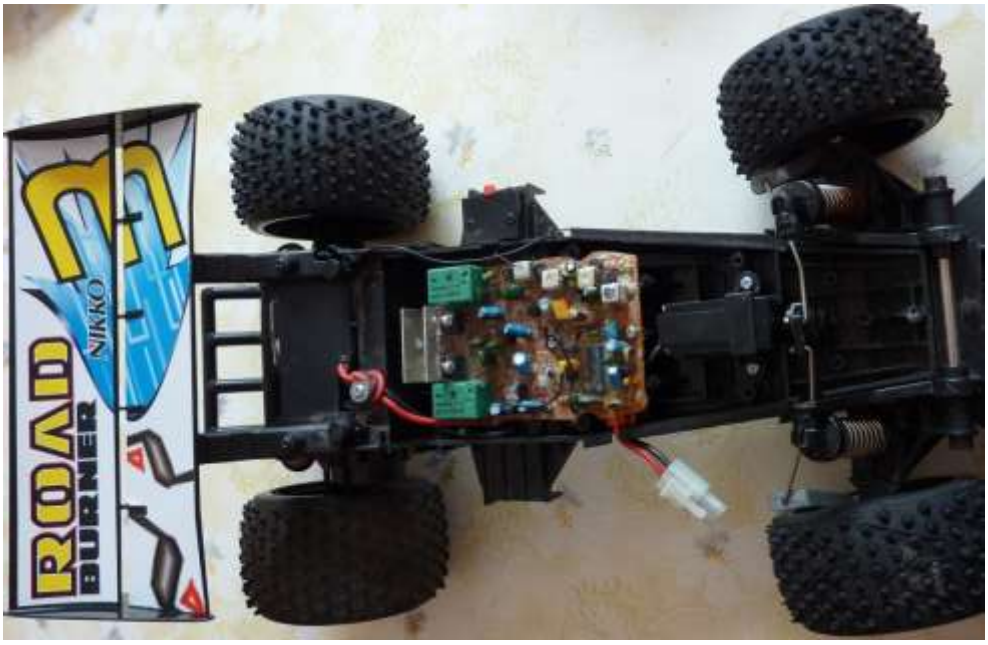
Les APC220 sont livrés par paire, avec les antennes et un adaptateur USB pour les essais avec un PC et le paramétrage des modules.

C'est tout pour la télécommande dans un premier temps. J'ai également un afficheur LCD 2 lignes de 16 caractères, équipé de poussoirs, qui pourra servir plus tard à agrémenter la façade de la télécommande et afficher des infos en retour de la voiture (tension batterie, direction relevée par un capteur magnétique…). Un pack de piles ou une batterie fournira l'alimentation à l'ensemble.

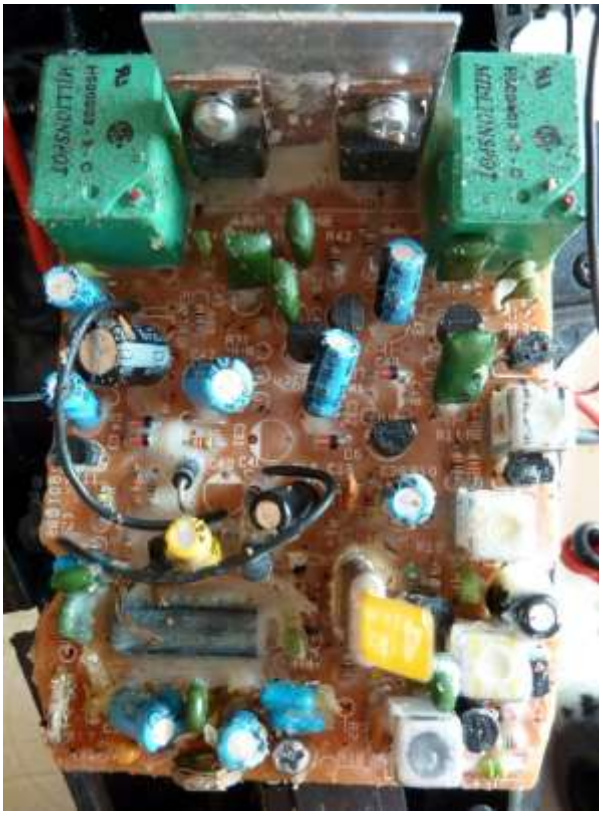
La voiture

Le circuit principal

Ce sera un Arduino Uno qui pilotera la voiture. Tout simplement pour des raisons de disponibilité… J'avais en stock une carte Duemilanove et une Uno. Chacune trouvera donc son utilisation



La carte d'origine Nikko Phoenix



Sur la carte principale on reconnait en haut les deux relais d'inversion du sens de marche (MILLIONSPOT H500S03-2-C)

Et les deux transistors qui commandent la vitesse du moteur.
A droite le récepteur superhétérodyne sur lequel arrive l'antenne (les bobines sont « noyées » dans la cire). Le quartz interchangeable est accessible par le dessus, avec sa tirette jaune. Enfin un circuit Nikko est chargé de décoder les informations envoyées par la télécommande et de commander relais, étage de puissance et servo de direction.

La liaison radio

Puisque le choix s'est porté sur les APC220, c'est également une carte APC220 qui équipera la voiture et recevra les flux de données pour les retransmettre à la carte Arduino.

La commande moteur

Ne connaissant pas l'état de la carte électronique récupérée avec la voiture, je préfère partir sur du neuf. Le moteur Mabushi 540 qui équipe la voiture (alimentée par une batterie 7,2v d'origine) consomme environ 2A à vide, et une dizaine en charge.

RS-540RH/SH

MABUCHI MOTOR
Carbon-brush motors

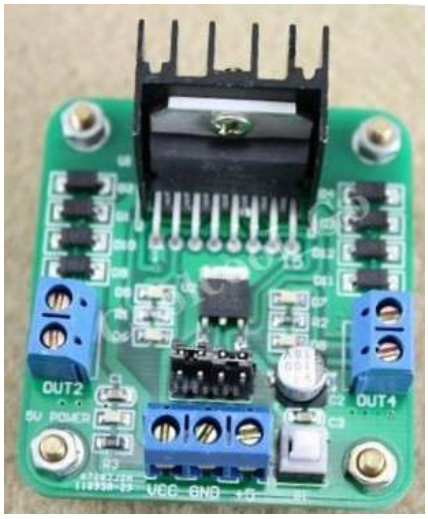
OUTPUT : 5.0W~90W (APPROX)

WEIGHT : 160g (APPROX)

Typical Applications **Home Appliances** : Vacuum Cleaner
Cordless Power Tools : Drill / Screwdriver / Cordless Garden Tool / Air Compressor
Toys and Models : Radio Control Model

MODEL		VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
		OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
				r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RS-540RH-6530	(*1)	3.6~6.0	6V CONSTANT	19200	1.30	15980	6.45	15.1	154	25.3	90.3	920	32.0
RS-540RH-7522	(*1)	3.6~4.8	4.8V CONSTANT	20200	1.90	16510	8.50	14.3	146	24.8	78.5	800	38.0
RS-540SH-7520	(*1)	4.8~7.2	7.2V CONSTANT	23400	2.40	19740	13.0	30.6	312	63.2	196	1998	70.0
RS-540SH-6527	(*1)	4.8~9.6	9.6V CONSTANT	23400	1.60	20040	9.55	31.0	316	64.9	216	2202	57.0

Il fallait donc un circuit qui tienne la puissance. Le L298 devrait permettre de faire les essais. (http://stores.ebay.fr/coolcoolgo) (8€)



Servo de direction

En fonction de l'état du servo de direction intégré et de la réutilisation possible de l'électronique, j'utiliserai le servo d'origine ou je le remplacerai par un autre servo plus moderne. Sur le site nikkoevolution (http://nikkoevolution.fr/article/article_fiche.php?articleID=278) on comprend que le servo possède 5 ou 6 fils : 2 pour le moteur, 3 pour le potentiomètre et éventuellement un pour l'antiparasitage (soudé à la carcasse du moteur). Ici il y a 6 fils qui sortent du servo.

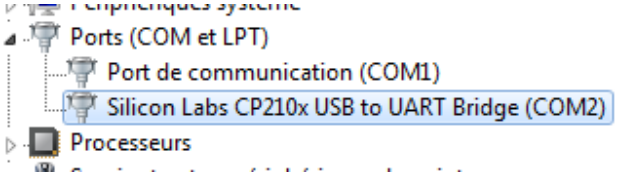
Les étapes à réaliser

Sur la télécommande

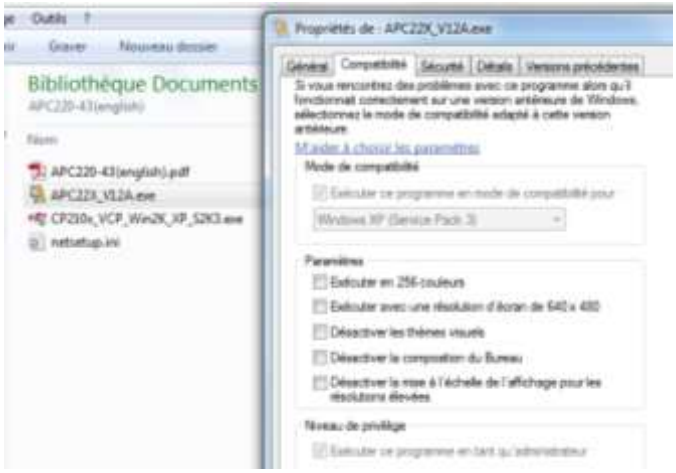
Définir le codage des infos envoyées par la télécommande
Connecter le joystick sur Arduino, sans liaison radio
Tests de fonctionnement du joystick et des poussoirs, sortie sur la RS232
Configuration des cartes APC 220, tests entre un PC et un Arduino
Le 29 juin 2012 les cartes APC220 arrivent.
Le vendeur m'envoie les drivers + logiciels par mail :

```
Hi,
please confirm from attachment file.
thanks
kevin
APC220-43(english).rar
```

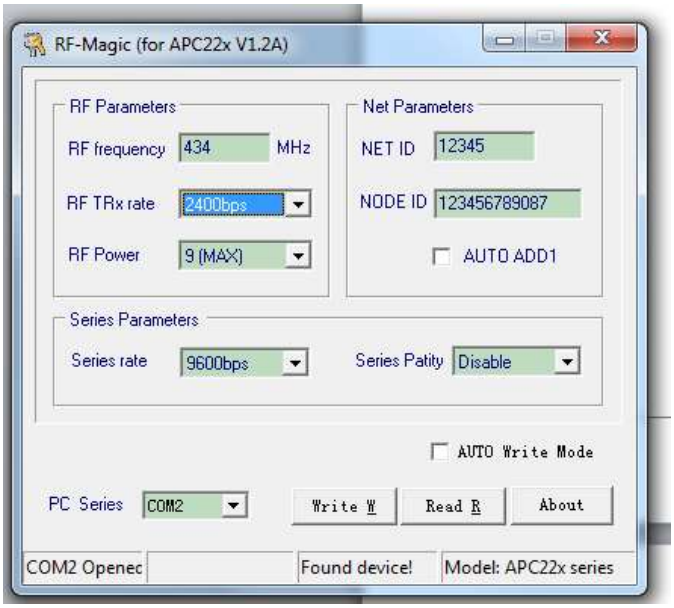
Le fichier .rar contient la notice de l'APC220, le driver USB CP210x… et le programme RF-ANET pour paramétrer les modules. L'installation du driver se passe sans encombre. Le port est vu dans le gestionnaire de périphérique. Je l'ai mis en COM2 dans les propriétés du pilote.



Le programme APC22X_V12A.exe sera configuré pour être exécuté en compatibilité WIN XP SP3 et comme administrateur.



Après lancement du programme, l'APC 220 connecté sur la carte adaptateur USB apparait :



On va pouvoir passer aux essais….

Voici la configuration testée : la télécommande, Arduino Duemillanove + shield joystick DFROBOTS + APC220-V3.0 et de l'autre côté APC220-V3.0 avec adaptateur USB.

Envoi du codage au PC via les APC220

Le sketch suivant, provenant de *mon-club-elec.fr* a servi de base pour tester la liaison, d'abord entre la télécommande et le PC avec le câble USB, puis au travers de la liaison radio fournie par les APC220.

```
// --- Programme Arduino ---
// Trame de code générée par le générateur de code Arduino
// du site www.mon-club-elec.fr

// Auteur du Programme : X. HINAULT - Tous droits réservés
// Programme écrit le : 18/2/2011.

// ------- Licence du code de ce programme -----
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License,
//  or any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

// ////////////////////  PRESENTATION DU PROGRAMME ////////////////////

// -------- Que fait ce programme ? ---------
 /* Ce programme test le Shield Arduino Joystick + BPx2
 et affiche l'état des BP et du Joystick dans la fenetre Terminal série.  */

// --- Fonctionnalités utilisées ---

// Utilise la connexion série vers le PC
// Utilise la conversion analogique numérique 10 bits
// Utilise la carte d'extension Arduino (shield) Joystick + BP x 2

// -------- Circuit à réaliser ---------

// La connexion série vers le PC utilise les broches 0 et 1 (via le câble USB)
// Enficher la carte d'extension Joystick + BPx2 broche à broche sur la carte
Arduino


// ////////////////////////////// 1. Entête déclarative //////////////////////
// A ce niveau sont déclarées les librairies incluses, les constantes, les
variables, les objets utiles...

// --- Déclaration des constantes ---

// --- Inclusion des librairies ---

// --- Déclaration des constantes utiles ---
const int APPUI=LOW; // constante pour tester état BP

// --- Déclaration des constantes des broches E/S numériques ---

const int bpRouge=3; // Constante pour la broche 3
const int bpBleu=4; // Constante pour la broche 4
const int bpJoystick=5; // Constante pour la broche 5
```

```
// --- Déclaration des constantes des broches analogiques ---

const int axe1Joystick=0; // Constante pour la broche analogique 0
const int axe2Joystick=1; // Constante pour la broche analogique 1

// --- Déclaration des variables globales ---

int mesure_brute=0;// Variable pour acquisition résultat brut de conversion
analogique numérique
float mesuref=0.0;// Variable pour calcul résultat décimal de conversion analogique
numérique

int positionAxe1=0; // Variable pour acquisition résultat brut de conversion
analogique numérique axe 1 Joystick
int positionAxe2=0; // Variable pour acquisition résultat brut de conversion
analogique numérique axe 2 Joystick


// --- Déclaration des objets utiles pour les fonctionnalités utilisées ---


// ////////////////////////// 2. FONCTION SETUP = Code d'initialisation
//////////////////////////
// La fonction setup() est exécutée en premier et 1 seule fois, au démarrage du
programme

void setup()   { // debut de la fonction setup()

// --- ici instructions à exécuter 1 seule fois au démarrage du programme ---

// ------- Initialisation fonctionnalités utilisées -------

Serial.begin(9600); // initialise connexion série à 115200 bauds
// IMPORTANT : régler le terminal côté PC avec la même valeur de transmission


// ------- Broches en sorties numériques -------
 pinMode(13, OUTPUT);

// ------- Broches en entrées numériques -------
 pinMode (bpRouge,INPUT); // Broche bpRouge configurée en entrée
 pinMode (bpBleu,INPUT); // Broche bpBleu configurée en entrée
 pinMode (bpJoystick,INPUT); // Broche bpJoystick configurée en entrée

// ------- Activation si besoin du rappel au + (pullup) des broches en entrées
numériques -------
 //digitalWrite (bpRouge,HIGH); // Rappel au + activé sur la broche bpRouge
configurée en entrée
 //digitalWrite (bpBleu,HIGH); // Rappel au + activé sur la broche bpBleu
configurée en entrée
 //digitalWrite (bpJoystick,HIGH); // Rappel au + activé sur la broche bpJoystick
configurée en entrée
// cette activation n'est pas indispensable pour le shield Joystick + BPx2

// ------- Initialisation des variables utilisées -------

// ------- Ecriture du message d'accueil et allumage de la LED --------
Serial.println("  Test du module APC220");
```

```
  Serial.println("Taper 1 pour allumer la LED,et 2 pour l'eteindre");
  Serial.println("Tapez une touche: ");
  digitalWrite(13, HIGH);

} // fin de la fonction setup()
// ******************************************************************************

////////////////////////////////// 3. FONCTION LOOP = Boucle sans fin = coeur du
programme ///////////////////
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino
est sous tension

void loop(){ // debut de la fonction loop()

// --- ici instructions à exécuter par le programme principal ---

//mesure_brute= analogRead(broche_analogique) ; // acquisition conversion
analogique numérique (100µs env.) sur broche analogique indiquée

//----- lecture position Joytstick
positionAxe1=analogRead(axe1Joystick); // acquisition conversion analogique
numérique sur broche analogique axe 1
positionAxe2=analogRead(axe2Joystick); // acquisition conversion analogique
numérique sur broche analogique axe 2
// Serial.print(compteur++, HEX);
// Serial.print(","); // Séparateur
Serial.print(positionAxe1); // affiche état Axe 1
Serial.print(","); // Séparateur
Serial.print(positionAxe2); // affiche état Axe 2
Serial.print(","); // Séparateur
//---- lecture état des BP du shield Joystick
if (digitalRead(bpJoystick)==APPUI)
  {
    Serial.print("1"); // Etat du bouton du joystick
  }
 else
  {
    Serial.print("0"); // Etat du bouton du joystick
  }

Serial.print(","); // Séparateur

if (digitalRead(bpRouge)==APPUI)
{
  Serial.print("1"); // Etat du bouton rouge
}
else
{
  Serial.print("0"); // Etat du bouton rouge
}
Serial.print(","); // Séparateur
if (digitalRead(bpBleu)==APPUI)
{
  Serial.print("1"); // Etat du bouton bleu
}
else
{
  Serial.print("0"); // Etat du bouton bleu
}
```

```
Serial.println(); // saut de ligne

// exploitation des données depusi le PC
  if (Serial.available())
  {
    char input = Serial.read();
    switch (input)
      {
        case '1': //turn led on
        digitalWrite(13, HIGH);   // set the LED on
        delay(100);              // wait for a second
        Serial.println("LED allumee!!");
        break;
        case '2':
        digitalWrite(13, LOW);    // set the LED off
        delay(100);              // wait for a second
        Serial.println("LED eteinte!!");
        break;
      }
  }

delay(150); // entre 2 lectures

} // fin de la fonction loop() - le programme recommence au début de la fonction
loop sans fin
// ********************************************************************************

// ////////////////////////// Fin du programme ////////////////////
```

Les premiers essais avec le câble USB sont concluants.
Les essais avec les APC220 le sont un peu moins. Après configuration des cartes en 9600 b/s pour la liaison série côté PC et Arduino, et configuration de la liaison radio en 2400 b/s, les données provenant de la carte Arduino : position des 2 potentiomètres et état des 3 boutons (joystick, rouge et bleu) arrivent correctement sur le PC. Cependant, la commande de LED depuis le PC (1=allumée, 2=éteinte) ne parvient pas à l'Arduino…
Après essais, il faut passer en 19200 b/s et tout fonctionne. (saturation de la liaison si elle est établie à une vitesse trop faible ?, ou charge de la liaison par les infos qui seraient échangées entre les cartes : NET ID et NODE ID… en plus des données utiles)

La configuration des cartes APC220 est la suivante :

Il restera à tester la portée avec un débit de 19200 b/s, mais pour une voiture ça devrait suffire.
Extrait de la sortie du Moniteur série Arduino connecté à la COM2, sur l'APC220

```
  Test du module APC220
Taper 1 pour allumer la LED,et 2 pour l'eteindre
Tapez une touche:
504,497,0,0,0
504,497,0,0,0 .....................joystick au repos, aucun bouton appuyé
504,498,1,0,0 .....................bouton du joystick appuyé
504,497,0,0,0
504,497,0,1,0 .....................bouton rouge appuyé
504,498,0,0,0
504,497,0,0,1 .....................bouton bleu appuyé
504,497,0,0,0
503,0,0,0,0 .......................déplacement négatif joystick axe2
504,0,0,0,0
504,498,0,0,0
505,1023,0,0,0 ....................déplacement positif joystick axe2
504,497,0,0,0
1023,50,0,0,0 .....................déplacement positif joystick axe1
1023,234,0,0,0
504,498,0,0,0
314,498,0,0,0
0,497,0,0,0 .......................déplacement négatif joystick axe1
504,497,0,0,0
504,498,0,0,0
LED eteinte!! .....................Envoi de 2 depuis le clavier du PC
504,497,0,0,0
504,497,0,0,0
LED allumee!! .....................Envoi de 1 depuis le clavier du PC
504,497,0,0,0
…
```

Envoyer le codage au deuxième Arduino, tester la bonne réception (renvoyer vers la RS232)

Sur la voiture

Tester la faisabilité du pilotage du servo Nikko 6 fils
Après démontage (2 juillet 2012) je trouve un servo « maison », avec 6 fils en sortie, sans doute 2 pour le moteur, un pour la carcasse et trois pour le potentiomètre. Ils sont soudés directement sur la carte, à proximité d'un circuit marqué Nikko qui doit piloter le servo de direction et la commande moteur.

En cas d'impossibilité : montage d'un servo classique et tests de pilotage droite/gauche
Tester la commande moteur traction par le L298, accélération avant/arrière
Tester le freinage (qui sera fait par appui sur le poussoir du joystick)
Commander la voiture depuis le PC à travers les APC220

Sur l'ensemble

Test de la liaison et du codage
Portée des APC 220
Fiabilité de la liaison
Blocage de la voiture en cas de perte de réception des données
Ajouter les commandes auxiliaires (feux, sirène ?)