Compte-Rendu Projet Arduino PEIP2 : Jumping-Drone

VOUILLON Benjamin LASSERON Léo FERRERO Fabien encadré par : Pascal MASSON et

0. Description du projet

I. Motivations :

Nous avons voulu réaliser le drone d'espionnage de Rainbow 6, car c'est un jeu vidéo auquel nous jouons et cela nous a motivé de le ramener dans la vie réelle. De plus lors de nos recherches nous sommes tombés sur le Jumping sumo, un drone de parrot qui réalisais toutes les fonctionnalités du drone de Rainbow 6. Ainsi nous avions un modèle existant pour nous inspirer ce qui nous a également motivé à choisir ce projet.

Ce robot se déplace sur 2 roues, et il a une queue à l'arrière pour se stabiliser. C'est cette queue qui lui sert d'appui pour ses sauts.

Nous avons dressé le cahier des charges suivant :

II. Cahier des charges Principal :

-Robot à 2 roues

-Bras stabilisateur amovible

-Pilotage Bluetooth

-LED RGB

Secondaire :

-Commande LED RGB (

-Info batterie

-Esthétique (respiration, clignotant, fixe)

-Réglable par BT

-Saut du drone

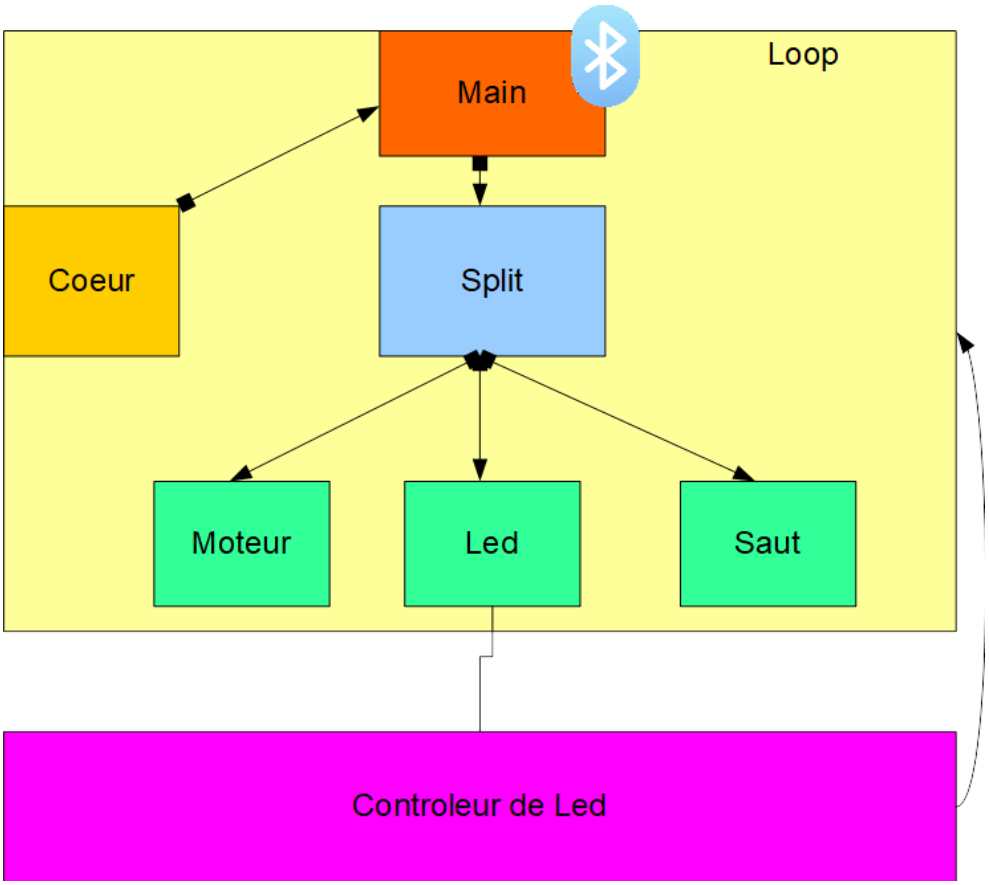
III. Cahier des charges Facultatif:

-Module vidéo

1. Evolution du Projet

I. Le code et son fonctionnement

Pendant 1 mois il a fallu se documenter pour connaitre le langage C++ et ses syntaxes pour arriver à l'architecture suivante :



Main : récupère le message Bluetooth de l'utilisateur grâce à un objet Cœur, puis transmet le message au Split

Cœur : C'est une classe d'objets qui sont composés d'une String (le mot envoyé) et d'un Boolean (détermine si le mot peut-être lu ou non) .

Split : C'est une fonction qui agit comme « un facteur ». On lui donne un message, elle se charge de le donner au bon bloc et le découpe en 2 String (Module, Argument)

Moteur : récupère un module et un argument puis fait fonctionner les moteurs.

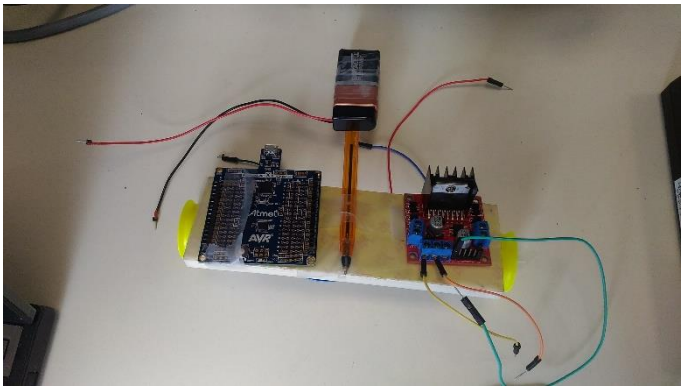
LED : récupère un module et un argument puis modifie les variables volatiles qui gèrent les commandes du ruban de LED. (Le bloc ne gère pas directement le ruban de LED, il modifie juste les variables Volatiles de commande des LED).

Contrôleur de LED : C'est une interruption qui contrôle les LED toutes les X millisecondes (cette interruption permet de contrôler les LEDS et en même temps de laisser l'utilisateur contrôler son drone.

Saut : Ce bloc ne prend rien en entrée, il permet simplement de lancer le servo-moteur pour faire un Saut en lui faisant faire un aller-retour 0°-180°.

II. Le montage et son fonctionnement

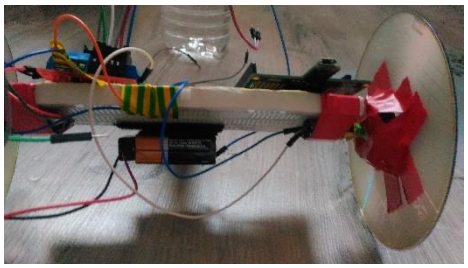
Phase 1



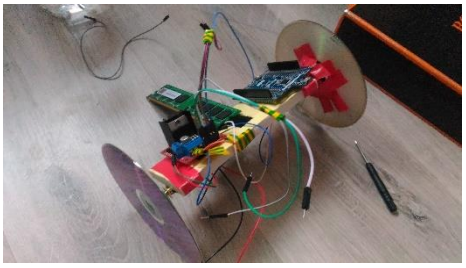
Voilà le concept de départ, simplement pour avoir une idée du positionnement des pièces et de la répartition de la masse sur le drone. Nous avions réalisé ce drone avec ce qu'on avait sous la main.

Le problème de cette maquette hormis son poids, c'est l'encombrement qu'elle représentait.

Phase 2



Pour éviter l'encombrement du prototype, nous avons remplacé les attaches en scotch de tous les éléments (sauf les moteurs car il fallait une fixation stable) par du velcro. Ainsi, on pouvait facilement démonter le projet pour le transporter.



Nous avons aussi lié les câbles de l'Arduino ensemble pour augmenter la rapidité de montage démontage du Drone.

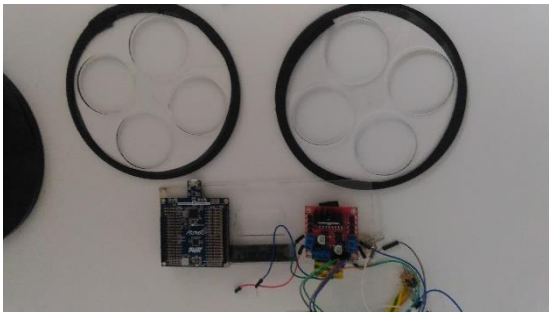
Pour tester les déplacements du drone à la maison nous avons récupéré un vieux chargeur 8.2V et dénudés les câbles pour alimenter les moteurs.

Nous avons aussi réalisé des roues avec des vieux DVD, du scotch d'électricien et des bouchons de bouteilles plastiques. Malgré différents essaies, ces roues ne permettaient pas d'avoir un drone stable.

Par ailleurs ces roues nous ont permis d'avoir une idée de l'envergure du drone final.

Les moteurs définitifs sont arrivés pendant cette phase et nous ont montrés des résultats plus intéressant que les premiers moteurs qui ne tournaient pas, car ces nouveaux moteurs possédaient un coefficient de couple de 70.

Phase 3



On a découpé les roues et le châssis dans du plexiglas pour avoir une structure plus solide.

Le problème 1 était que le moyeu des roues était trop large pour l'axe.

Le problème 2 était que les roues n'étaient pas équilibrées donc il y avait un risque d'abimer les axes des moteurs (axe pas centré).

Phase 4

Enfin des roues parfaites découpées dans du contreplaqué, elles sont plus légères, avec un moyeu qui correspond parfaitement avec l'axe et sont bien équilibrés. On monte les moteurs sur le châssis avec deux arceaux métalliques

Le bloc moteur/roues est enfin fini.

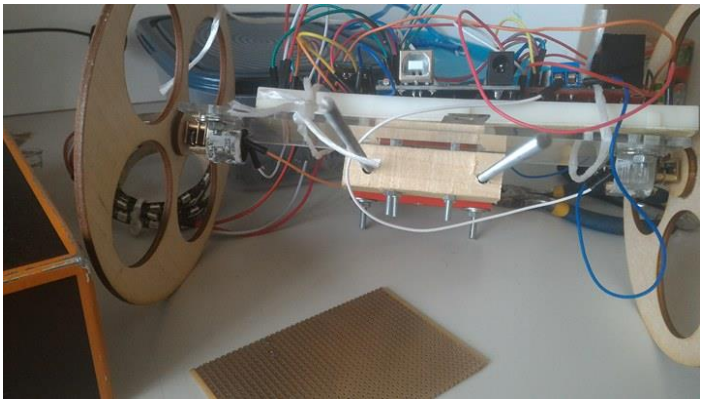
Phase 5

On fait imprimer la pièce 3d qui se visse sur le Servo moteur et permet la compression des ressorts.

Phase 6

On monte la queue sur le châssis ainsi que le Servo moteur.

Phase 7

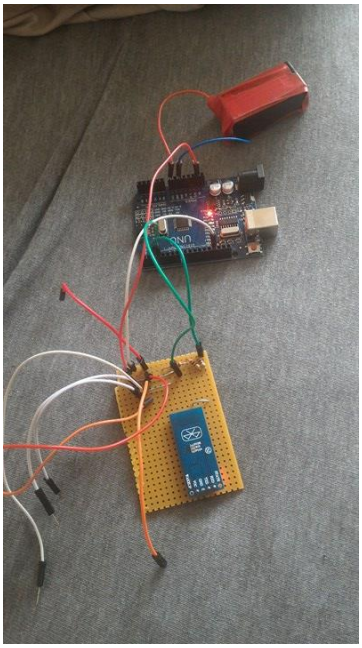


On réalise les branchements de tous nos modules sur la breadboard et on pose la breadboard sur le châssis en la maintenant attaché avec des ficelles.

On change de carte Arduino car l'alimentation externe ne fonctionne pas sur la nôtre.

On procède ensuite a différents tests pour vérifier nos branchements et notre programme. On comprend alors qu'il nous faut ajouter des batteries (2 en plus) car l'alimentation est trop peu puissante pour tous les moteurs et le ruban de LED.

Phase 8



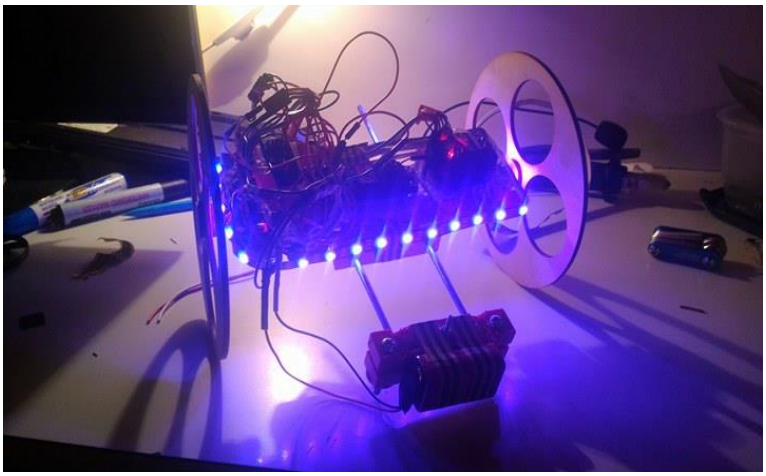
Les tests étant concluants on décide de retirer la breadboard et de faire tous nos branchements sur une plaquette, plus légère et moins encombrante.

La plaquette est bien moins pratique pour des tests que la breadboard en effet on s'y reprend plusieurs fois pour terminer le montage. Mais une fois finie elle est bien plus pratique pour faire les tests car elle se fixe facilement au châssis.

Phase 9

On monte la plaquette et tous les composants sur le châssis et on test le drone. Tous les moteurs fonctionnent et la commande Bluetooth aussi

Phase 10



On finalise le montage en rajoutant le ruban de LED sur le tour du châssis. On le test, tout fonctionne les modes lumineux sont modifiables via le Bluetooth

Enfin on visse la pièce 3D sur le Servo moteur. Le robot roule, change de couleur et le Servo moteur tourne : On prend donc une vidéo pour sauvegarder l'événement.

2. Quels sont les problèmes rencontrés ?

Le premier problème que nous avons rencontré c'est le manque d'organisation de l'architecture du code au début qui nous a fait perdre 1 mois de travail. En effet le travail séparé du code fonctionne bien au début mais lorsque le code se complexifie et que la partie codée par l'autre devient nécessaire cela fonctionne moins bien car il faut réadapter certaines parties voir réécrire une bonne partie du code car les parties ne pouvaient pas communiquer entre elles (différents moyens de coder).

Nous avons aussi perdu du temps sur le code du fait de ne pas connaitre la logique des interruptions avant le début du programme. Ce qui à fait perdre beaucoup de temps durant le projet car une fois comprise nous avons dû chacun l'intégrer dans notre code.

La non prise en compte du délai entre l'envoi des différents caractères via le Bluetooth à faire perdre du temps car nous ne savions pas que nos problèmes de code venaient de là, nous pensions à un autre problème ou à un souci électronique (module Bluetooth cassé). Nonobstant en testant le module Bluetooth avec nos codes réalisés lors du cours nous avons compris que cela ne venait pas de là et avons finalement réalisé d'où venait réellement le problème et avons pu le solutionner.

Le second problème c'est l'organisation de nos recherches de produits pour le projet, on a perdu du temps au début à chercher une pile pour notre projet car on n'arrivait pas à définir la consommation totale du drone. Nous avons aussi testé plusieurs moteurs qui ne marchaient pas pour rouler et avons dû les commander, ce qui as pris du temps car nous ne trouvions pas de moteurs avec une datasheet complète sur Aliexpress or nous voulions être sûr de commander directement les moteurs adaptés à notre projet.

Quand notre code été terminé et fonctionnel nous avons rencontrés des problèmes au montage, cela prenait beaucoup de temps car nous ne pouvions pas le faire chez nous et nous ne pouvions pas aller souvent au FabLab qui n'était pas toujours ouvert aux heures ou nous étions libres. L'Arduino ne fonctionnait pas au niveau de l'alimentation externe nous avons donc dû changer pour une autre carte. Nous avons aussi rencontré un problème de puissance car au début nous n'avions qu'une pile, lorsque l'on branchait tous

les moteurs la carte redémarrait car les moteurs prenaient tout le courant. Nous avons donc rajouté 2 piles.

Enfin nous n'avons pas eu le temps de terminer le système de saut, le Servo moteur compressant les ressorts est bien fonctionnel, la pièce 3D nécessaire est bien monté dessus et la queue est construite mais nous n'avons pas pu relier la queue à ce système Servo moteur pièce 3D. En effet nous avons mis beaucoup de temps à trouver les ressorts adaptés, de plus nous avions des difficultés à calculer leur constante de raideur car nous ne savions pas le poids que ferait le drone une fois monté. De surcroit la queue n'est pas parfaite, nous avons utilisés les matériaux disponibles mais ils ne sont pas idéaux.

3. Quelles sont les améliorations futures ?

Nous pensons que le drone mérite un contrôle de la trajectoire plus intelligent pour éviter que le drone dérive en ligne droite et que le pilotage soit plus facile. Pour cela nous aurions besoin d'un accéléromètre pour pouvoir corriger la trajectoire, et remplacer les moteurs continues par des moteurs pas à pas pour une correction de trajectoire plus précise.

Le saut n'étant pas fini il serait bien de le terminer afin de répondre pleinement à nos objectifs initiaux, notamment de reproduire le drone du jeux vidéo Rainbow Six Siege.

On pourrait également améliorer l'aspect esthétique en rajoutant une coque plastique tout autour du châssis pour masquer les composants et nous pourrions recouvrir les LED d'une gaine translucide.

Enfin nous pourrions coder une application Android propre à notre projet plus ergonomique et plus adapté que l'application généraliste « Bluetooth Android Controller ».

4. Ce que l'on retient de ce projet :

On a beaucoup appris sur plusieurs plans :

- Nous avons appris à ne plus perdre de temps lorsque l'on recherche de l'électronique sur un site. Nous avons perdu un temps fou à comparer les Batteries et les moteurs. Au début Benjamin allait plusieurs fois par jour sur internet pour trouver un nouveau produit, plus léger, plus performant, moins chère, bien qu'il ait déjà choisis le meilleur produit qui était disponible. Mais après quelques semaines nous appris à être plus méthodique et prendre les problèmes les uns après les autres, pour avancer efficacement. Ce gain d'efficacité c'est fait ressentir lorsque j'ai nous avons dû reprendre le code à 0 pour créer une architecture de code plus propre et ensuite traiter les requêtes Bluetooth. Nous avons d'abord cherché des pages en lien avec les interruptions, puis repensé le programme du début à la fin en conséquence. Ce qui a permis en quelques heures de travails réparties sur plusieurs semaines d'avoir un code qui fonctionne sans pour autant l'avoir testé sur le drone final.
- Au niveau du travail d'équipe, Nous avons appris à nous faire comprendre et à comprendre notre binôme. Par exemple durant le projet nous avons été confrontés à des problèmes, ou à des désaccords souvent dus à des soucis de communication plutôt que de réels désaccords sur le fond. On a donc pris du recul pour mieux expliquer, fais des dessins, utilisé des images pour que le message passe et pris du temps pour bien s'expliquer nos codes.
- Benjamin a obtenu de bonnes notions sur la programmation en C++. Et découvert appliqué la méthode « habile » pour savoir chaque semaine où j'en suis dans mon projet et ce qui reste à faire.
- Enfin nous avons compris qu'il valait mieux prévoir un projet moins ambitieux mais le terminer et pourquoi pas l'améliorer ensuite.

5. Le code :

```
#include <L298N.h> // Controle moteur
#include <Servo.h> // Commande servo
#include <MsTimer2.h> // Interrupteur logique
#include <WS2812FX.h> // Controle des Leds
#include <SoftwareSerial.h> //Software Serial Port

#define LED_COUNT 33
#define LED_PIN 11
#define SAUT 10
#define EN 4
#define IN1 5
#define IN2 6
#define RxD 3 //Pin 10 pour RX, PB2 sur votre board, a brancher sur le TX du HC-06
#define TxD 2 //Pin 11 pour TX, PB3 sur votre board, a brancher sur le RX du HC-06
#define EN1 7
#define IN3 8
#define IN4 9
Servo myservo;
SoftwareSerial BTSerie(RxD,TxD);


WS2812FX ws2812fx = WS2812FX(LED_COUNT, LED_PIN, NEO_GRB +
NEO_KHZ800);// Led controleur
volatile char mode='B'; //mode de fct led
volatile int vite=200; // vite de clignotement
volatile String couleur="0x660099"; //couleur led
bool Bordre;

/*Début de requête des blocs*/
const String moteur = (String) 'm';
const String led = (String) 'l';
const String saut = (String) 's';

/// caractére d'ordre ////
```

```
const String avance = (String) 'f';
const String recule = (String) 'b';
const String gauche = (String) 'l';
const String droite = (String) 'd';
const String vitesse = (String) 'v';
const String stoppe = (String) 's';

/// limite d'utilisation ///
const int vmax = 180; // 180 car sinon risque de bruler les moteurs

/// caractére de comportement Led ///
const String Rainbow= (String) 'R';
const String Cligno= (String) 'C';
const String Static= (String) 'S';
const String Breath= (String) 'B';
const String Larson= (String) 'F';

////Variables ///////

int vit = 0; // vitesse, puissance électrique

//creation moteurs
L298N motorGauche(EN, IN1, IN2);
L298N motorDroite(EN1, IN3, IN4);

////////////////////////Recepteur d'ordre ////////////////////////////////
class Coeur {
private:
String mot;
bool loading; // détermine si le mot est affichable ou non
public:
void Ecrire(); // Ecrit le caractére
void Afiche(); // Affiche le mot
void Refresh(); // Réinisialise le mot
int Test(); // Test si le mot est lisible ou non
```

```
String Retour();// Retourne le mot
};

void Coeur::Ecrire() {
char car;
car = BTSerie.read(); // catch mot
if (car == '$' or Serial.available() < 0) { // caractére de fin de mot
loading = 0;
} else { // je crée mon mot
mot = String(mot + car);
}
};
void Coeur::Afiche() {
Serial.println(mot);
};

int Coeur::Test() {
return loading;
};

void Coeur::Refresh() {
mot = "";
loading = 1;
}
String Coeur::Retour() {
return mot;
}

Coeur bon; // variable de test

//////////////////////Distribue les commandes et les arguments//////////////////

/* Objectif de Splitter : Séparer le mot du coeur
en 2 partie, une lettre arg et un mot qui sera une suite de chiffres
Et distribuer l'ordre aux différents modules du drone*/

void Split(String mot) {
```

```
String bloc = "";
// je catch le premier caractére du mot
bloc = mot.substring(0, 1);
mot.remove(0, 1);
// je catch le module
String arg = mot.substring(0, 1);
mot.remove(0, 1);

// je dispache mon ordre dans le bon bloc
if (bloc == moteur) {
Moteur(arg , mot);
};
if (bloc == led) {
Led(arg , mot);
};
if (bloc == saut) {
Saut();
};
};

////////////////////////// bloc de commandes//////////////////////////////////////
void Moteur(String arg, String module) {
/// je distribue l'argument
if (arg == avance) {
motorGauche.forward();
motorDroite.forward();
} else if (arg == recule) {
motorGauche.backward();
motorDroite.backward();
} else if (arg == droite) {
motorGauche.forward();
motorDroite.backward();
} else if (arg == gauche) {
```

```
motorGauche.backward();
motorDroite.forward();
} else if (arg == vitesse) {
vit = module.toInt(); //// je convertie le module en nombre
if (vit > vmax) { //// je limite la tension sinon les moteurs crame ( 9v ---> 6v)
vit = vmax;
};
motorGauche.setSpeed(vit);
motorDroite.setSpeed(vit);
} else if (arg == stoppe) {
motorGauche.stop();
motorDroite.stop();
};
};
void Led(String arg, String module) {
ordre(arg,module);
};

void Saut() {
//// j'utilise le bloc saut
myservo.write(0);
Serial.println("JEVAISEN0");
delay(1000); // délai à modifier en fonction du temps réel pour le servo a effectuer une demi rotation.
Serial.println("JEVAISEN180");
myservo.write(180);
};

void ordre(String mode,String coul){
if(mode== Rainbow ){
ws2812fx.setMode(FX_MODE_RAINBOW_CYCLE);
//R=MODE ARCENCIEL RAINBOW_CYCLE
}
```

```
else if(mode== Cligno){
ws2812fx.setMode(FX_MODE_BLINK);
//C=MODE CLIGNOTEMENT BLINK
}
else if(mode== Static){
ws2812fx.setMode(FX_MODE_STATIC);
//S=MODE FIXE STATIC
}
else if(mode== Breath){
ws2812fx.setMode(FX_MODE_BREATH);
//B=MODE RESPIRATION BREATH
}
else if(mode== Larson){
ws2812fx.setMode(FX_MODE_LARSON_SCANNER);
//F=MODE BALAYAGE LARSON_SCANNER
}else{
ws2812fx.setMode(FX_MODE_STATIC);
ws2812fx.setColor(0x000000);
// if nothing else matches, do the default
// default is optional
}

ws2812fx.setSpeed(vite);
ws2812fx.setColor(0x660099);
ws2812fx.start();
};
void runLED(){
ws2812fx.service();
};
void setup() {
Serial.begin(9600);
motorDroite.setSpeed(vit); // definie la vitesse des moteurs
motorGauche.setSpeed(vit);
```

```
myservo.attach(SAUT);

// Configuration du bluetooth
pinMode(RxD, INPUT);
pinMode(TxD, OUTPUT);
BTSerie.begin(9600);
delay(100);
// Configuration LED
ws2812fx.init();
ws2812fx.setBrightness(100);
ordre("","");

MsTimer2::set(10, runLED); // 500ms period
MsTimer2::start();
}

void loop() {
if (BTSerie.available() > 0) {
while (BTSerie.available() and bon.Test() == 1) {
bon.Ecrire();
}
Split(bon.Retour());
bon.Refresh();
}
}
```