

Rapport de projet

L'EXPLOBOT – MARS 2019

Anthony Barna, Michele Bona
ENCADRANT : PASCAL MASSON



Présentation du projet

Voici le rapport de l'ExploBot de Michele Bona et Anthony Barna. Notre projet est un véhicule tout-terrain étudié pour explorer des grottes, des couloirs, des décombres et autres terrains accidentés difficiles d'accès pour l'Homme...

L'ExploBot est contrôlé avec une manette de PlayStation 2 pour un contrôle précis des déplacements et est alimenté par batterie, ce qui lui garantit une certaine autonomie. En plus de la batterie notre explorateur embarque avec lui une caméra IP (qui retranscrit les images en temps réel sur un téléphone par Wifi) et une LED assez puissante pour rendre les images de la caméra perceptibles, même dans l'obscurité.

Le robot a une bonne mobilité et est assez agile. Il est compact (23.5cm de long, 18cm de large et 18cm de haut) et pèse moins de 2kg qui sont répartis vers l'avant pour faciliter le franchissement d'obstacles.

Le projet a été réalisé sur 8 séances, entre le 10 décembre et le 4 mars. Nous allons ici vous présenter notre travail en détail selon le plan qui suit...

Vous pourrez trouver ici une [vidéo de présentation](#).



SOMMAIRE

I Objectifs et cahier des charges	3
II Agencement du projet	4
II.1. La partie mécanique	4
II.2. La partie électronique	5
II.3 L'algorithme	7
III Contraintes rencontrées et solutions apportées.....	8
IV Conclusion	9
IV.1 Comparasion objectifs/réalité	10
IV.2 Améliorations possibles	10
Bibliographie	11
Annexe A - Le code.....	11
Annexe B - Caractéristiques des pièces utilisées	15

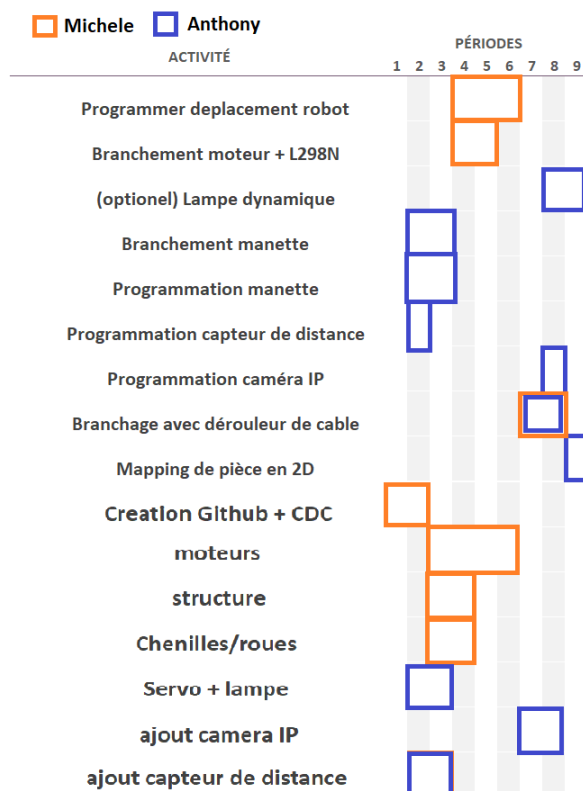
I Objectifs et cahier des charges

Nous avons imaginé l'ExploBot à la base comme un char « tout-terrain » qui avait pour objectif l'exploration de pyramides (avec caméra, lampe, détection d'obstacles, mapping de pièce et d'autres ajouts qui auraient été à préciser plus tard). Il devait être contrôlé par câble car les connections par ondes ne passent pas dans ces environnements.

Ce but primitif s'est ensuite généralisé à l'exploration de toute zone inaccessible comme des décombres ou des conduits. Nous avons donc jugé que l'accès à ces zones, la collecte et la récupération d'images et la mobilité du robot étaient nos priorités.

Nous voulions aussi un dérouleur de câble pour pouvoir transmettre des informations au robot même à grande distance.

Nous avons commencé par organiser notre travail en créant un GitHub et à réfléchir au montage de notre robot en s'inspirant de projets similaires. De cette réflexion est né le cahier des charges suivant :

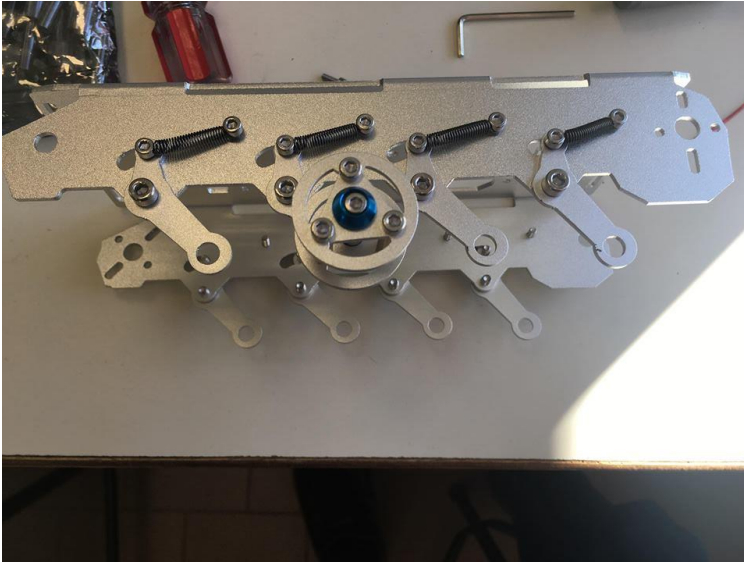


Nous nous sommes donnés pour objectif dans les premières semaines de créer la « base » du robot, c'est-à-dire d'avoir un châssis, des chenilles fonctionnelles et manipulables avec une manette.

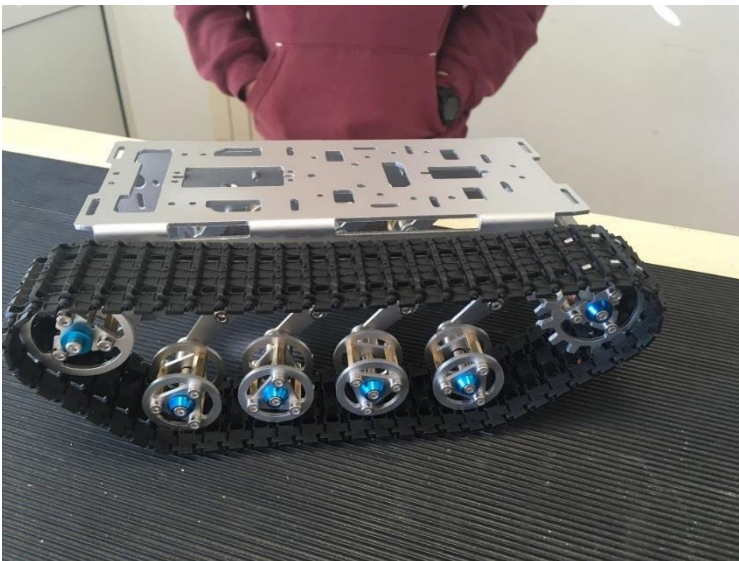
L'idée étant qu'une fois la base terminée nous puissions nous concentrer sur l'ajout de la caméra et de la lampe, mais aussi des fonctionnalités secondaires (comme le capteur de distance ou encore le mapping 2D...).

II Agencement du projet

II.1 La partie mécanique



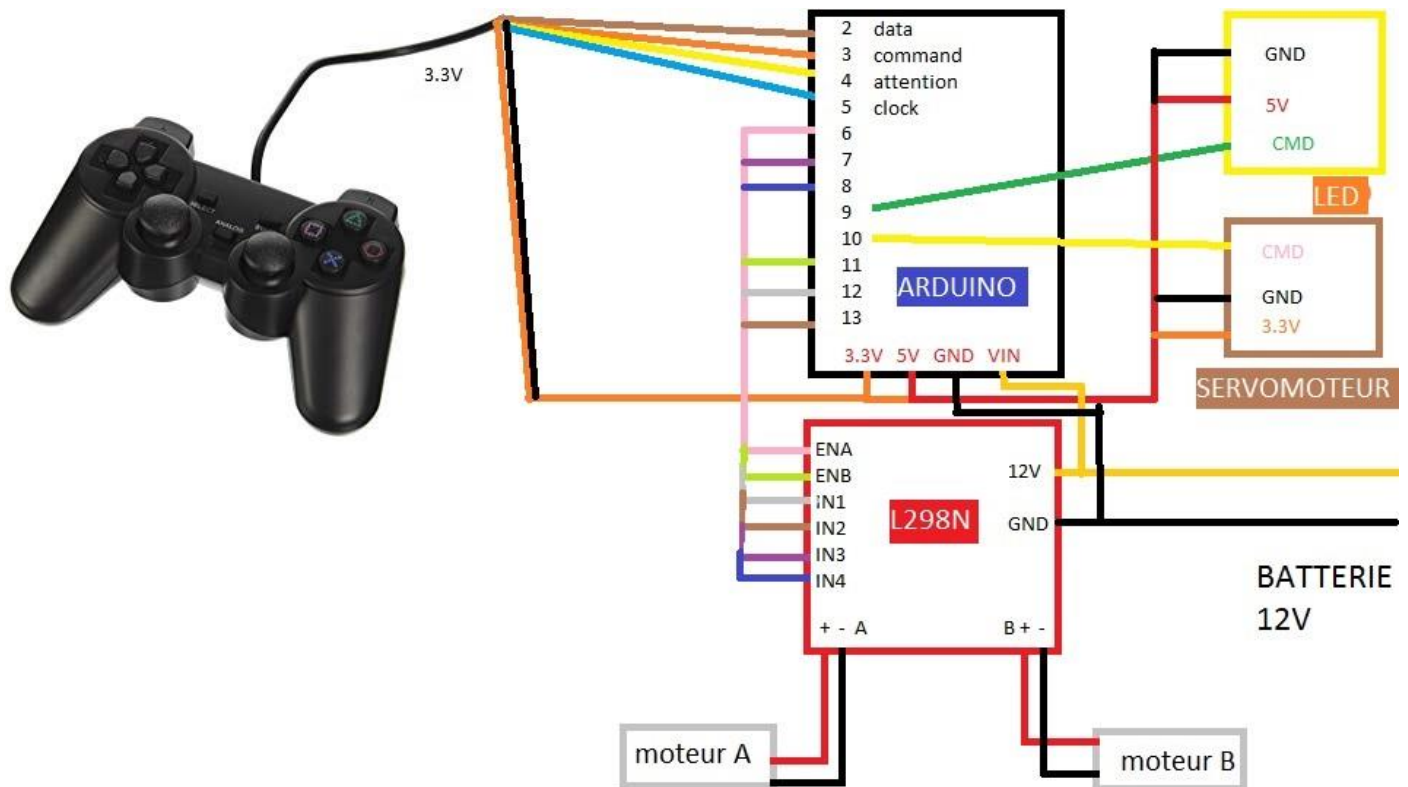
En ce qui concerne la partie mécanique de l'ExploBot, il est composé d'un châssis en alliage d'Aluminium où sont attachées 12 roues (6 par côté) et de deux moteurs 12V. Il y a de chaque côté la même structure : une roue motrice dentée connectée à un moteur pour entraîner la chenille, cinq roues dont quatre plus basses, attachées à des suspensions en Aluminium.




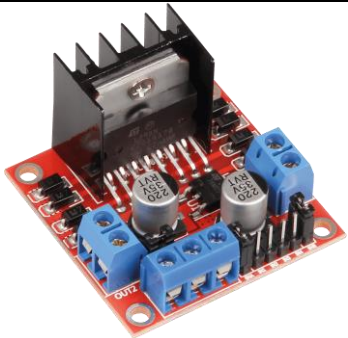




La chenille, elle, est en plastique. Elle a deux rangées de « dents » pour y caler les roues non-motrices au milieu et ainsi les bloquer. Mais aussi des trous pour les dents des roues motrices. On a par la suite fixé une planche de bois sur le châssis pour y installer la carte Arduino et les autres composants électroniques...

II.2 La partie électronique

Voici un schéma qui représente le montage électronique de l'ExploBot :



Voici les composants présents sur le schéma :

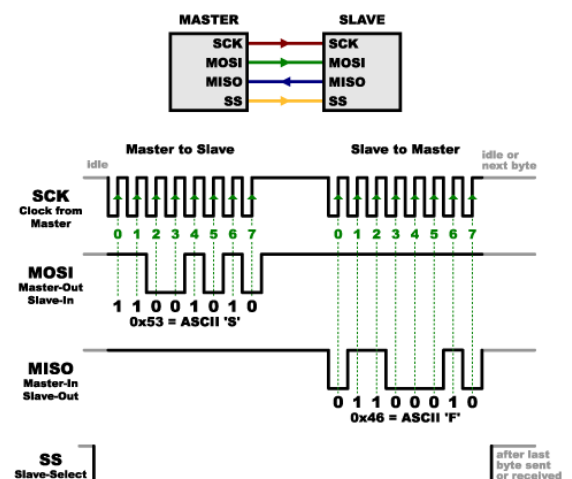
 <p>Arduino UNO R3 x1</p>	 <p>L298N x1</p>	 <p>Module LED haute puissance x1</p>
 <p>Servomoteur x1</p>	 <p>Manette PS2 x1</p>	 <p>Moteur JGA 25 370 x2</p>

Alimentation :

- Une batterie de 12V (ou un câble) alimente le module L298N (c'est un module qui va envoyer le courant désiré dans les deux moteurs à courant continu) qui va à son tour alimenter en 12V la carte Arduino qui possède une entrée appropriée.
- Le UNO R3 possède deux alimentations : une en 5V et une en 3.3V. La manette et la LED haute puissance sont alimentées sur le 3.3V tandis que le servomoteur est alimenté sur le 5V.

Fonctionnement :

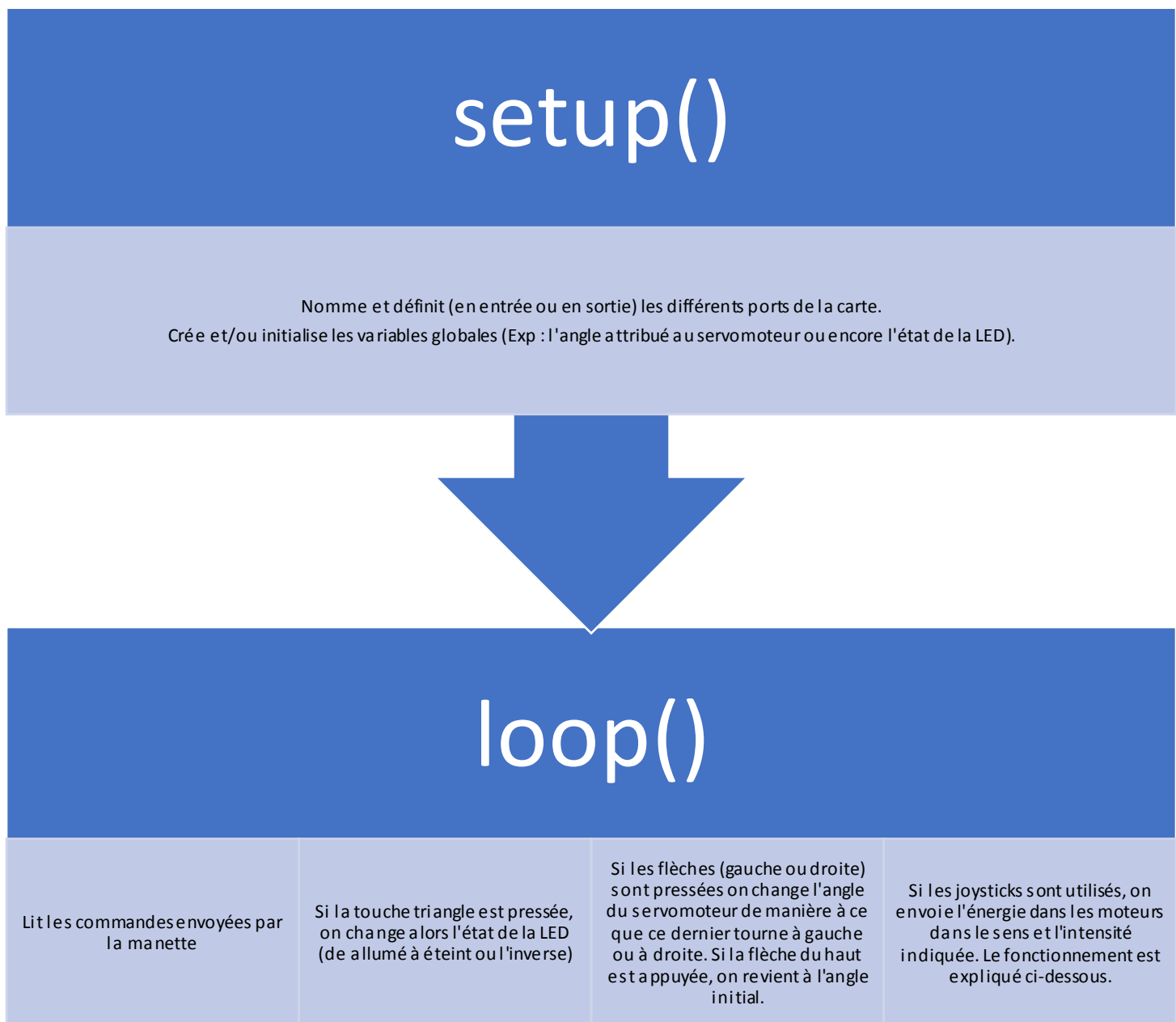
- La carte Arduino UNO R3 est programmable et contient un l'algorithme (cf II.3) qui lui permet d'interagir avec le reste des composants avec ses entrées et sorties numériques et analogiques. C'est « l'intelligence » du robot.
- Le module L298N permet de faire passer un courant désiré dans un sens désiré dans un moteur, et peut gérer deux moteurs simultanément (il utilise le principe du double pont en H). Par exemple pour le moteur A, les entrées numériques IN1 et IN2 vont définir un sens de rotation (si IN1 vaut la valeur logique HAUT et que IN2 vaut la valeur logique BAS, le sens de rotation sera le sens inverse de celui imposé par les valeurs IN1 = BAS et IN2 = HAUT). Le module enverra un courant proportionnel à celui envoyé dans le port ENA (le courant maximal pour ENA = 5V et un courant nul pour ENA = 0V). Le principe est exactement le même pour le moteur B avec IN3, IN4 et ENB.
- Le servomoteur est fait pour recevoir une information (un signal plus ou moins long en fonction de l'angle désiré) et en fonction de cette dernière, se positionner sur un angle compris entre 0 et 180 degrés.
- La LED haute puissance reçoit une valeur logique haute ou basse, en fonction de cela elle va s'allumer ou s'éteindre.
- En ce qui concerne la manette, elle fonctionne en liaison Serial Peripheral Interface. C'est un mode de communication où la carte Arduino se positionne en maître et la manette en esclave. Au démarrage de l'ExploBot, une fréquence d'horloge est imposée par le maître (cf câble SCK). Ensuite des paquets d'octets vont être échangés entre le maître et l'esclave (par les câbles MOSI et MISO). En fonction de ces octets, l'algorithme va reconnaître les

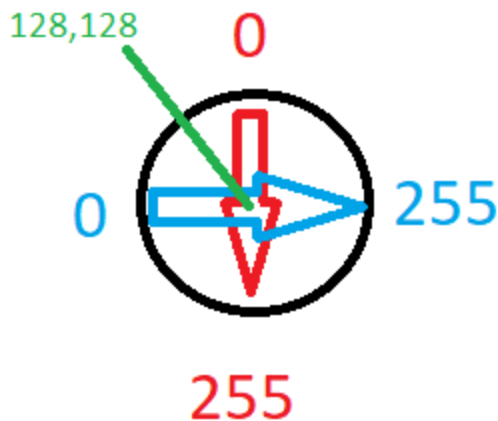


commandes de la manette. Le câble SS reste ici fixe car il n'y a qu'un seul esclave.

II.3 L'algorithme

L'algorithme est programmé en langage C (une surcouches est apportée par Arduino). Il se décompose en deux grosses fonctions : la fonction `setup()` qui est exécutée une fois au démarrage de l'ExploBot et la fonction `loop()` qui est exécutée en boucle, après la fonction `setup()`. Voici un schéma qui explique ce que réalise l'algorithme.





En ce qui concerne les joysticks, la manette en possède deux. Celui de gauche indique la direction du robot, tandis que celui de droite indique l'intensité à fournir (pour avancer plus ou moins vite) et le sens de fonctionnement (marche avant ou arrière). Cette configuration permet une manipulation précise de l'ExploBot. Chaque joystick possède deux axes qui renvoient des valeurs comprises entre 0 et 255. Si on ne touche pas le joystick (au repos), il renverra alors 128 pour l'axe X et Y. Grâce à ces informations on peut alors traiter les données renvoyées

par les joysticks pour indiquer le sens de rotation (donc les valeurs respectives à envoyer dans les ports IN1, IN2, IN3 et IN4) et l'intensité (donc les valeurs respectives à envoyer dans ENA et ENB) des deux moteurs.

Pour plus d'informations sur le code de l'ExploBot, référez-vous à l'Annexe A.

III Contraintes rencontrées et solution apportées

Les principales contraintes rencontrées en ce qui concerne la partie mécanique de l'ExploBot ont été les suivantes :

- La recherche du « châssis parfait ». En effet pour qu'il puisse atteindre nos objectifs, nous avons cherché un châssis assez petit avec un bon rapport poids/puissance, mais aussi capable de déplacement précis et en « tout-terrain ». Initialement nous voulions le monter nous-mêmes de A à Z, mais le choix d'un kit de châssis de char avec chenille nous a permis de satisfaire toutes les contraintes mécaniques.
- Ensuite le montage du kit que nous avons reçu nous posa un problème... En effet il s'agissait d'un kit contenant de nombreuses pièces et il n'y avait aucun manuel ou tutoriel de montage en ligne. Néanmoins la structure étant symétrique et similaire aux autres modèles de ce type nous avons pu à force de temps le monter complètement.

- Enfin la qualité du matériel nous posa de nombreux problèmes tout au long du projet. En effet pour des raisons de budget la qualité de nos pièces n'était pas toujours excellente : les vis se desserraient souvent et le système prévu pour fixer l'axe moteur était très fragile. Il est arrivé deux fois qu'il casse et donc que le moteur tourne dans le vide. Nous avons donc utilisé les rechanges et essayé de consolider le tout avec de la colle forte. Cela a par la suite bien tenu.

En ce qui concerne la partie électronique et algorithmique, il n'y a pas eu de problème particulier, mis à part la gestion de la manette qui a pris beaucoup plus de temps que prévu et la première manette qui était défectueuse (qui a donc engendré la commande d'une deuxième et un retard dans notre emploi du temps).

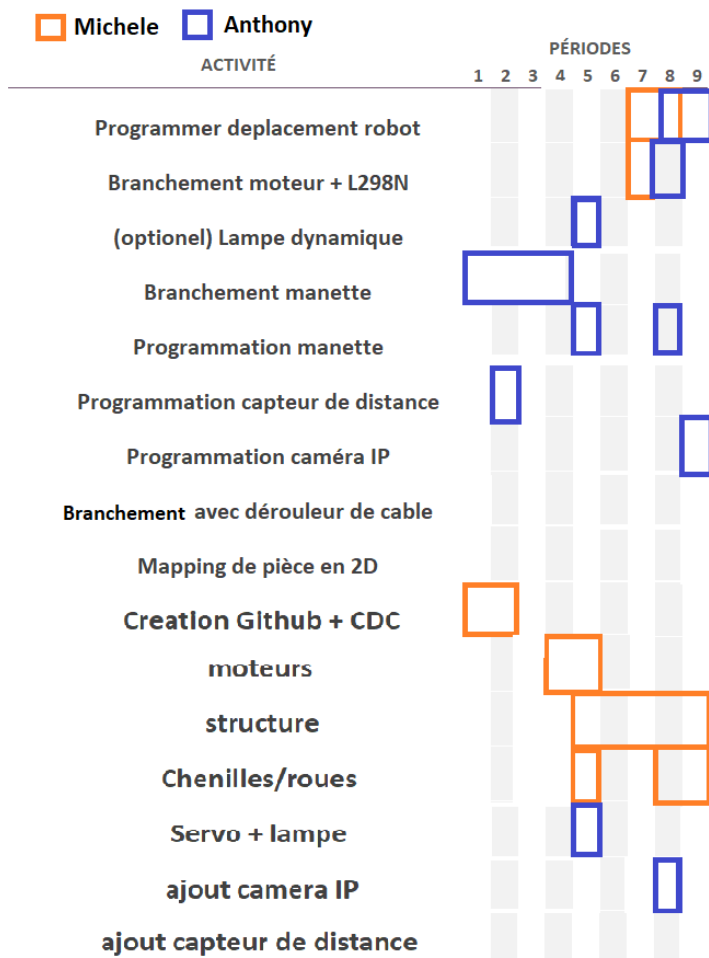
Ainsi, entre le branchement de la manette, son montage (soudures éventuelles) et sa programmation (qui demandait d'utiliser et de comprendre la librairie PS2X, disponible sur le GitHub du projet), beaucoup de temps s'est écoulé.

IV Conclusion

À la fin des délais accordés, l'ExploBot est opérationnel et remplit parfaitement sa fonction principale, qui est d'obtenir des images depuis des endroits difficiles d'accès pour l'Homme (cf [vidéo de présentation](#)). Les objectifs de départ et le résultat obtenu diffèrent pourtant. Nous expliquons ci-dessous pourquoi et comment nous aurions pu atteindre ces objectifs ou encore ce que nous aurions aimé faire de plus si nous avions eu plus de temps.

IV.1 Comparaison objectifs/réalité

Nous allons voir comment nous avons finalement géré notre temps, pour cela nous avons refait notre cahier des charges mais avec cette fois le temps réel que nous a pris chaque partie du projet.



On peut remarquer que nous sommes en général restés dans les temps et que nous avons atteint la plupart de nos objectifs (le char est opérationnel et contrôlé par manette, la caméra embarquée et la LED sont branchées sur un servomoteur aussi contrôlé avec la manette).

Certains éléments comme on l'a vu dans la partie III nous ont pris plus de temps que prévu (en particulier le branchement de la manette et la fixation de la structure, car le châssis nous a posé des soucis jusqu'à la fin...).

Enfin par faute de temps, certains éléments n'ont pas été rajoutés (comme le capteur de distance (donc au passage le mapping de pièce) qui était déjà programmé et installable ou encore le dérouleur de câble car nous n'avions plus le temps d'en commander un).

IV Améliorations possibles

Nous considérons que le cahier des charges initial était réalisable. Avec notre recul actuel, nous pensons qu'en ayant mieux optimisé notre temps (nos commandes, notre ordre de priorité...) nous aurions pu :

- Alimenter l'ExploBot par câble, à l'aide d'un dérouleur de câble qui aurait offert un grand champ de déplacement et plus de légèreté (la batterie est un poids non négligeable ajouté au robot).
- Ajouter un capteur de distance sur le servomoteur, et donc faire une analyse de l'environnement même sans luminosité et à une plus grande distance que par caméra.

Si nous avons plus de temps pour développer notre projet, en plus des améliorations citées ci-dessus, nous mettrions en place :

- Un nouvel axe de rotation pour la caméra qui nous permettrait de voir le sol et le plafond.
- Une meilleure protection du robot car il peut être amené à évoluer dans des milieux dangereux (chutes de gravats..).
- Une possibilité de passer facilement d'un mode filaire à un mode sans fil en fonction de l'environnement.

Bibliographie

Voici une liste des sites qui nous ont aidé à développer notre projet :

- <https://guides.github.com/activities/hello-world/>
- <https://github.com/madsci1016/Arduino-PS2X>
- <http://electrotuto.com/2014/04/24/manette-ps2-et-arduino-ps2-controler/>
- <https://www.youtube.com/watch?v=AgYTPRtgiOc&t=13s&index=243&list=WL>
- <http://store.curiousinventor.com/guides/PS2/>
- https://www.robotshop.com/community/blog/show/dimensionnement-dun-moteur-dentrainement?fbclid=IwAR0VdNh-2mzuaciSKdauuZ91zApe_bjoAggo7xqJ2IzH82BfVPI8dG-eU0
- <https://www.youtube.com/watch?reload=9&v=-9WLhvXwGhw&fbclid=IwAR0rRtJXA89wvFhWXzcnA9K-GSughgLdlz31veiSMiEf1Z12Aa2vgWzvQ0Y>

Annexe A – Le code

```
1 #include <PS2X_lib.h> //for v1.6
2 #include<Servo.h>
3
4 //manette
5 #define PS2_DAT      2 // data brown résistance de 1K soudée
6 #define PS2_CMD      3 // command orange
7 #define PS2_SEL      4 // attention yellow
8 #define PS2_CLK      5 // clock blue
9
10 #define pressures    false
11 #define rumble        false
12
13 PS2X ps2x;
14
```

```

15  int error = 0;
16  byte type = 0;
17  byte vibrate = 0;
18
19  //servomoteur
20  #define cmdServo 9
21  Servo servo;
22  int angle=90; //angle donné en paramètre au servomoteur
23
24  //lampe
25  #define lampe 10
26  bool light=false;
27
28  //moteurs
29  #define ENA 6
30  #define ENB 11
31  #define IN1 12
32  #define IN2 13
33  #define IN3 7
34  #define IN4 8
35
36  int vitesseA; //vitesse des deux moteurs
37  int vitesseB;
38
39  void setup(){
40
41      //moteurs
42      pinMode(ENA,OUTPUT);
43      pinMode(ENB,OUTPUT);
44
45      pinMode(IN1,OUTPUT);
46      pinMode(IN2,OUTPUT);
47      pinMode(IN3,OUTPUT);
48      pinMode(IN4,OUTPUT);
49
50      digitalWrite(ENA,LOW);
51      digitalWrite(ENB,LOW);
52
53      //servo
54      servo.attach(cmdServo);
55
56      //lampe
57      pinMode(lampe,OUTPUT);
58      digitalWrite(lampe,LOW);
59
60
61      //manette
62
63      Serial.begin(57600);
64      delay(300); //pour laisser la manette se connecter
65      error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT, pressures, rumble);
66
67      if(error == 0){
68          Serial.print("Found Controller, configured successful ");
69          Serial.print("pressures = ");
70          if (pressures)
71              Serial.println("true ");
72          else
73              Serial.println("false");
74          Serial.print("rumble = ");
75          if (rumble)
76              Serial.println("true");
77          else
78              Serial.println("false");

```

```

79     }
80     else if(error == 1)
81         Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshootin
82
83     else if(error == 2)
84         Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for trou
85
86     else if(error == 3)
87         Serial.println("Controller refusing to enter Pressures mode, may not support it. ");
88
89
90     type = ps2x.readType();
91     switch(type) {
92         case 0:
93             Serial.print("Unknown Controller type found ");
94             break;
95         case 1:
96             Serial.print("DualShock Controller found ");
97             break;
98         case 2:
99             Serial.print("GuitarHero Controller found ");
100             break;
101         case 3:
102             Serial.print("Wireless Sony DualShock Controller found ");
103             break;
104     }
105 }
106
107 void loop() {
108     //commandes servo
109     servo.write(angle); //on passe l'angle en argument à chaque passage de boucle, on changera la valeur de angle pour faire bouger l
110
111     //moteurs
112     if(ps2x.Analog(PSS_RY)<=128){ //si on est en marche avant (on regarde le joystick droit sur l'axe Y)
113
114         //A avant
115         digitalWrite(IN1,HIGH);
116         digitalWrite(IN2,LOW);
117
118         //B avant
119         digitalWrite(IN3,LOW);
120         digitalWrite(IN4,HIGH);
121
122         //Marche avant le joystick va de 128 vers 0
123         vitesseA=map(ps2x.Analog(PSS_RY),128,0,0,255);
124         vitesseB=map(ps2x.Analog(PSS_RY),128,0,0,255);
125     }
126     else{ //si on est en marche arrière
127         //moteur A arrière
128         digitalWrite(IN1,LOW);
129         digitalWrite(IN2,HIGH);
130
131         //moteur B arrière
132         digitalWrite(IN3,HIGH);
133         digitalWrite(IN4,LOW);
134
135         //marche arrière le joystick va de 129 vers 255
136         vitesseA=map(ps2x.Analog(PSS_RY),129,255,0,255);
137         vitesseB=map(ps2x.Analog(PSS_RY),129,255,0,255);
138     }
139
140     if(ps2x.Analog(PSS_LX)<=128){ //tourner à gauche càd réduire la vitesse du moteur gauche
141
142         vitesseA-=map(ps2x.Analog(PSS_LX),128,0,0,255);

```



```

143
144     if(vitesseA<0){ //la vitesse doit rester à minimum 0
145         vitesseA=0;
146     }
147 }
148 else{ //tourner à droite càd réduire la vitesse du moteur droit
149
150     vitesseB-=map(ps2x.Analog(PSS_LX),128,255,0,255);
151
152     if(vitesseB<0){ //la vitesse doit rester au minimum à 0
153         vitesseB=0;
154     }
155 }
156
157 analogWrite(ENA, vitesseA);
158 analogWrite(ENB, vitesseB);
159
160 if(error == 1) //on passe la boucle si aucune manetyte n'est trouvée
161     return;
162 else { //DualShock Controller
163     ps2x.read_gamepad(false, vibrate); //read controller and set large motor to spin at 'vibrate' speed --> on n'utilise pas le vib
164
165     if(ps2x.Button(PSB_PAD_UP)) {
166
167         angle=90; //on remet le servo à sa position initiale
168     }
169     if(ps2x.Button(PSB_PAD_RIGHT)){ //augmenter l'angle du servo
170
171         angle-=10;
172     }
173     if(ps2x.Button(PSB_PAD_LEFT)){//baisser l'angle du servo
174
175         angle+=10;
176     }
177
178     //pour rester avec un angle compris entre 0 et 180
179     if(angle<0){
180         angle=0;
181     }
182     if(angle>180){
183         angle=180;
184     }
185
186     vibrate = ps2x.Analog(PSAB_CROSS); //this will set the large motor vibrate speed based on how hard you press the blue (X) butt
187     if (ps2x.NewButtonState()) { //will be TRUE if any button changes state (on to off, or off to on)
188
189         if(ps2x.Button(PSB_TRIANGLE))
190
191             if(light==false){
192                 digitalWrite(lampe,HIGH);
193                 light=true;
194             }
195             else{
196                 digitalWrite(lampe,LOW);
197                 light=false;
198             }
199         }
200     }
201     delay(50);
202 }

```

Annexe B – Caractéristiques des pièces utilisées

Arduino UNO R3

Features of the Arduino UNO:

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

Commande d'entraînement de moteur à courant continu L298N

- Alimentation intégrée : 5V
- Tension de pilotage : 7-35V
- Courant logique: 0mA-36mA
- Tension logique: 5V
- Courant d'entraînement: 2A
- Tension d'entraînement: 5V-35V
- Température de stockage: -20 à +135
- Puissance maximum: 25W

Moteurs JGA 25 370

Nom de la marque: ASLONG

Courant Continu (A): 0.35A

Usage: Bateau,Appareil domestique,Voiture,Ventilateur...

Type: Moteur à engrenage

Efficacité: IE 2

Commutation: Brosse

keyword: Mini gear motor

application : automatic equipment

OEM service : yes

Certification: CE,VDE,CCC,ROHS,UL

Construction: Aimant permanent

Couple: 9.5KG.CM

Puissance de sortie: 28W

Numéro du modèle: JGA 25-370

Caractéristique de protection: Fermé totalement

factory property : professional manufacturer

quality policy : life long warranty

Poids	55g
Dimensions	40.7 x 19.7 x 42.9 mm
Couple (kg.cm)	10
Couple (N.m)	0.98
Vitesse de rotation (s/°)	0.20 sec/60°
Tension d'alimentation	4.8 – 7.2V
Tension maximale	7.2V
Température de fonctionnement	0 – +55 °C

3W LED Module haute puissance Module Pour Arduino

Caractéristique:

Température de couleur: blanc chaud 3000-3200k / 6000-6500k blanc

Courant direct: 700mA

Flux lumineux: White 180-200LM (lumens)

Tension directe: 3.3 à 3.6 v

Tension inverse: 5v

Puissance: 3w

Angle de vision: 180 °

Température de travail: -20 ° à 60 °

Température de stockage: 20 ° à 60 °