Stage Assistant Ingénieur

Conception de robots de surface

Auteur :
Martin Gounabou

Tuteur :
Luc Jaulin
Maître de stage :
Vincent Hugel

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à la réussite de mon stage. Tout d'abord mes remerciements vont à l'endroit de mon professeur M. Luc JAULIN qui m'a permis d'avoir ce stage.

Ensuite je tenais à remercier mon maître de stage M. Vincent Hugel, responsable du laboratoire COSMER au sein de l'université de Toulon, pour ses conseils et la nature des missions qu'il m'a confié. Ces missions m'ont permis d'approfondir mes compétences en électronique et en programmation. Des acquis essentiels que je n'aurais pas intégrer sans son encadrement et sans la confiance qu'il a fait preuve à mon égard.

Je souhaite également remercier, tout le personnel du service notamment Mme. Sabine, Mme. KALLIOPI et Mme. MOUTTE Laïs, qui ont su m'accueillir avec gentillesse et bienveillance.

Table des matières

Introduction

Les robots de surface ou encore USV pour Unmanned Surface Vehicle sont des drones marins de surfaces motorisés. Ils ont pour principales applications l'océanographie et la bathymétrie.
Ce stage vise à réaliser des robots de surface, non pas pour la bathymétrie mais plutôt pour une utilisation pédagogique. Ces robots que nous nommerons DD-BOAT par la suite, embarqueront avec eux plusieurs capteurs à savoir des centrales inertielles, des GPS ainsi que plusieurs composants électroniques tels que des Raspberry et des Arduino. L'idée est de rendre ces robots autonomes d'une part et d'un autre côté de les rendre télécommandes. Ces robots serviront à la formation des étudiants inscrits dans le cadre du programme Erasmus Mundus à l'université de Toulon. Ces étudiants pourront directement mettre en pratique les connaissances vues en classe grâce à ces robots.

Première partie

CONCEPTION D'UN DDBOAT AUTONOME

1 Schéma de conception

1.1 Liste du matériel utilisé

— SparkFun OpenLog Artemis dev 16832
— HobbyWing Quicrun WP-1060, Brushed Sbec W
— Hacker 95000331 Pack de batterie (LiPo)
— Voltcraft Chargeur V-Charge, LiPo eco1000
— Renkforce RF-3425172
— ALIMENTATION 8-26 V BEC
— GPS Micro M8N UBLOX
— Raspberry Pi 4
— Boîtier d'Inclusion ABS Noir, avec couvercle,
— Câble USB type C
— Câble USB micro-B
— Récepteur X8R
— Taranis qx7

1.2 Architecture

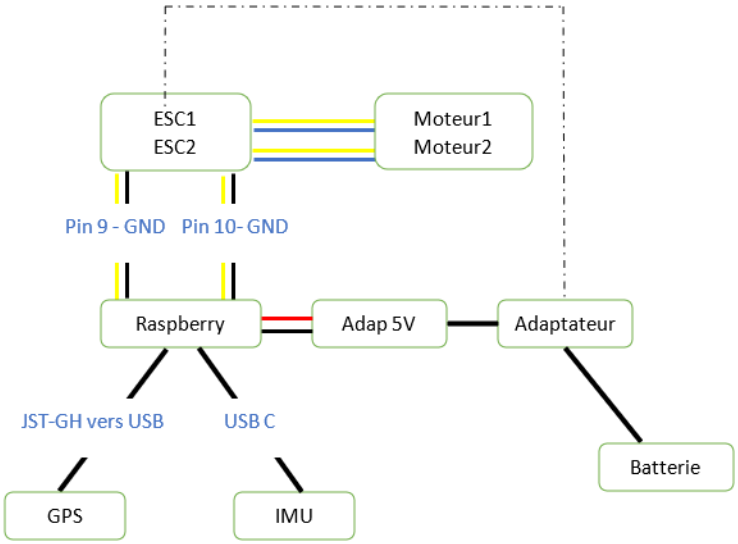


FIGURE 1: Schema de branchement des composants

2 Prise en main du GPS

2.1 Branchement

Pour la conception du DDBOAT, comme GPS nous utiliserons le GPS MICRO M8N U-BLOCK **(Annexe 1)**. Ce GPS est livré avec un connecteur JST-GH **(Annexe 2)** qui permet d'utiliser le GPS sur un PIXHAWK. Dans notre cas, nous utiliserons ce GPS sur une Raspberry Pi 4 et donc il faudra adapter le connecteur. Ainsi deux solutions s'offrent à nous.

1ère solution : utilisation du port mini UART

Cette solution consiste à sectionner un coté du câble JST-GH pour le raccorder avec des fils Dupont **(Annexe 3)**. Ensuite les fils Dupont pourront se fixer sur le port UART de la Raspberry (TX : GPIO14, RX : GPIO15). Ce port sera accessible grâce au périphérique /dev/ttyUSB0. Par ailleurs la commande **dmesg** permet de trouver le nom des périphériques.

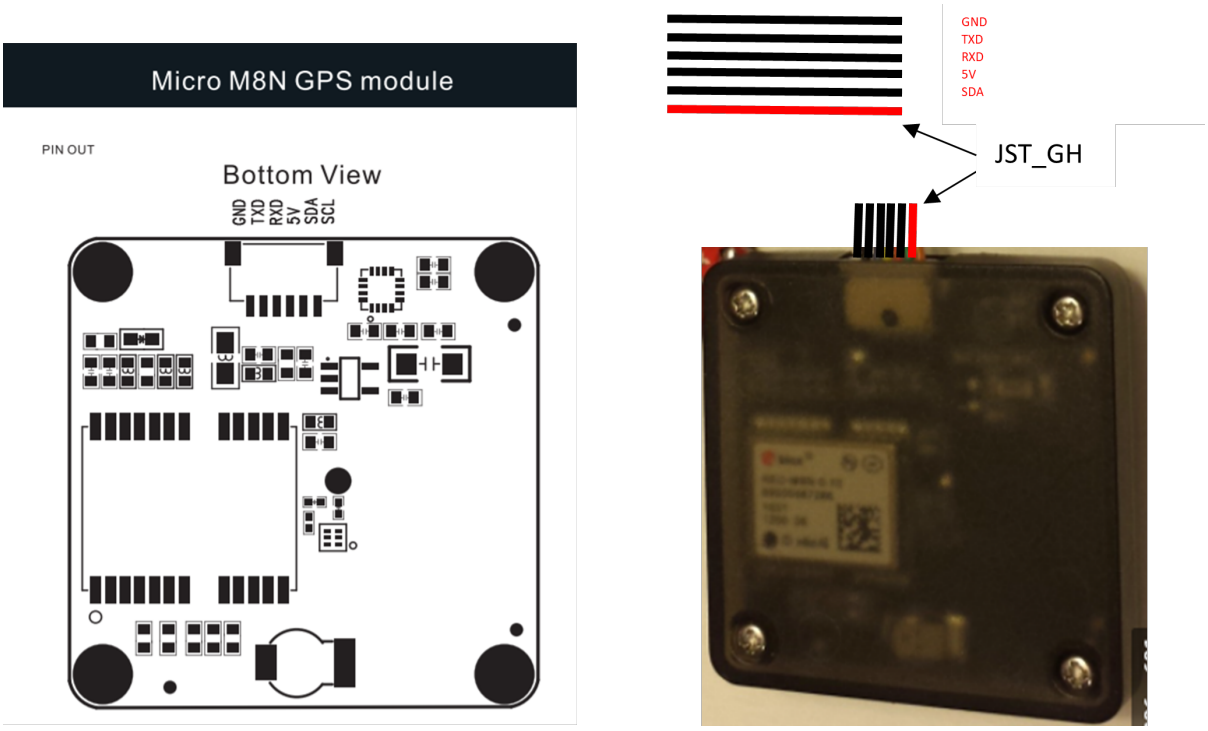


FIGURE 2: GPS Module Pinout Mapping

Puisque nous voulons utiliser le port mini UART de la Raspberry, les pins du GPS concernés sont GND, TXD, RXD, et 5V. Les pins SDA et SCL étant utilisés pour la communication via le bus i2c. Après soudure des fils Dupont sur les pins concernés, on branche le fil Dupont 5V sur l'un des pins 5V de la Raspberry. On branche ensuite le TXD et le RXD respectivement sur le RX (GPIO15) et le TX (GPIO 14) de la Raspberry. Et enfin on branche le GND sur l'un des GND de la Raspberry.

2e solution : utilisation des ports USB de la Raspberry

Cette solution n'a pas été testée. Elle consiste à souder les mêmes pins à savoir GND, TXD, RXD et 5V du câble JST-GH sur un port USB.

2.2 Récupération des données GPS par la Raspberry

Récupération avec cutecom ou putty

Les étapes à suivre pour récupérer les données via cutecom ou putty sont : lancer l'application (cutecom, putty, ...), sélectionner le périphérique puis renseigner le baudrate. Et enfin cliquer sur OK.

Récupération des valeurs avec un script python

Pour récupérer les données du GPS, il faudra utiliser le driver gps_drivers_py3.py. Tout d'abord, il faut l'importer dans le script puis utiliser les différentes fonctions. Dans le script gps_drivers_py3.py, les deux fonctions importantes sont init_line et read_glll. La fonction init_line permet l'ouverture du port /dev/ttyUSB0 avec un débit de 9600. De l'autre côté la fonction read_gll permet de récupérer les trames NMEA. A partir des trames NMEA du GPS, on extrait la latitude et la longitude. Chaque trame (cf. ref. [2]) débute par le caractère $, ce caractère est suivi par un groupe de 2 lettres pour l'identifiant du récepteur :

- **II** Integrated Instrumentation (eg. AutoHelm Seatalk system).
- **LC** Loran-C receiver.
- **G**P pour Global Positioning System.
- **OM** Omega Navigation receiver.

Puis un groupe de 3 lettres pour l'identifiant de la trame.

- **VTG** : pour Direction (cap) et vitesse de déplacement (en nœuds et Km/h).
- **GGA** : pour GPS Fix et Date.
- **RMC** : pour données minimales exploitables spécifiques.
- **GLL** : pour Positionnement Géographique Longitude - Latitude.
- **GSA** : pour DOP et satellites actifs.
- **GSV** : pour Satellites visibles.

Suivent ensuite un certain nombre de **champs** séparés par une **"virgule"**. La virgule permet la dé-concaténation des données dans le programme de traitement des données, elle sert donc comme séparateur
Comme trame nous avons :

La trame : RMC

Données minimales recommandées de spécification GPS
$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68 225446 = Heure du Fix 22 :54 :46 UTC A = Alerte du logiciel de navigation (A = OK, V = warning (alerte) 4916.45,N = Latitude 49 deg. 16.45 min North 12311.12,W = Longitude 123 deg. 11.12 min West Ces données "minimales", sont le plus souvent utilisées dans les programmes de navigation-GPS simples.

La trame : GLL

Position Géographique - Longitude / Latitude – GPS
$GPGLL,4916.45,N,12311.12,W,225444,A 4916.46,N = Latitude 49 deg. 16.45 min. Nord. 12311.12,W = Longitude 123 deg. 11.12 min. West (ouest) 225444 = Acquisition du Fix à 22 :54 :44 UTC

La trame : GGA

Données d'acquisition du FIX – GPS.
$GPGGA,123519,4807.038,N,01131.324,E,1,08,0.9,545.4,M,46.9,M, , *42 123519 = Acquisition du FIX à 12 :35 :19 UTC 4807.038,N = Latitude 48 deg 07.038' N 01131.324,E = Longitude 11 deg 31.324' E

Par exemple si on utilise la trame GPRMC, juste avant la lettre N et W, on retrouve respectivement la latitude et la longitude.

```
1
2  def read_gll(ser, nmax=20):
3      val=[0.,'N',0.,'W',0.]
4      for i in range(nmax):
5          v=ser.readline().decode("utf-8")
6          if str(v[0:6]) == "$GPRMC":
7              vv = v.split(",")
8              val[0] = float(vv[3])
9              val[1] = vv[4]
10             val[2] = float(vv[5])
11             val[3] = vv[6]
12             val[4] = float(vv[1])
13             break
14     return val
```

Cette fonction permet d'extraire les valeurs de la longitude et de la latitude.

3 Prise en main IMU

3.1 Installation du firmware

Nous utiliserons l'OPENLOG ARTEMIS DEV 16832 **(Annexe 3)**. Tout d'abord il faut installer le firmware, et ensuite on pourra récupérer les données de l'IMU. Pour commencer, il faut brancher l'IMU sur le PC à l'aide d'un câble USB, type A vers Type C, et avoir l'IDE Arduino installé sur le système d'exploitation. Nous utiliserons le système Windows, cependant l'installation du firmware peut se faire aussi grâce au système GNU/Linux. Pour l'installation du firmware, les étapes à suivre sont les suivantes :

— Téléchargement des librairies : il s'agit de deux dossiers, le premier à télécharger sur le site : `https://learn.sparkfun.com/tutorials/9dof-razor-imu-m` (se rendre au $\frac{3}{4}$ de la page, à la section **Librairie and example firmware** et cliquer sur **Dowload the flashstorage arduino librairie**) et le second est un clonage du repository `https://github.com/lebarsfa/SparkFun_`

`MPU-9250-DMP_Arduino_Library`. Ces dossiers devront être copiés et collés dans le dossier : Accès rapide -> Documents -> Arduino -> Librairie.

— Installation du Sparkfun Apollo3 Boards dans l'IDE arduino : Afin de pouvoir installer cette carte, il faudra ajouter le suivant `https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_sparkfun_index.json` dans : File -> Preferences -> Additionnal Boards Manager URLs. Ensuite, il faut s'assuer de choisir la carte SparkFun Apollo3 → SparkFun RedBoard Artemis ATP.

— Installation de la libsubrairie SparkFun ICM 20948 IMU : Sketch ->Include Library -> Manage Libraries. Dans la barre de recherche entrer « SparkFun ICM » puis cliquer sur "installer".

— La dernière étape consiste à cloner le repository suivant `https://github.com/Razor-AHRS/razor-9dof-ahrs`. Dans le fichier razor\razor_imu_9dof\rsrc\Razor_AHRS\Razor_AHRS.ino, il faut décommenter la ligne qui concerne notre modèle de razor c'est-à-dire la ligne 230 (#define HW__VERSION_CODE 16832 // SparkFun "OpenLog Artemis" version "DEV-16832").

Après toutes ces étapes, il ne reste plus qu'à compiler et à téléverser.

3.2 Lecture et récupération des données

Pour un premier test, on peut ouvrir le moniteur série de l'IDE Arduino, normalement on doit voir défiler des données. Si les données sont illisibles, cela est normal, il faut alors modifier le bauderate et mettre 57600, les données seront un peu plus compréhensibles. Pour avoir les données de l'IMU directement grâce à un script, il faudra utiliser le driver imu_drivers_py3_v2.py. Tout d'abord, il faut l'importer dans le script puis utiliser les différentes fonctions. Dans le script imu_drivers_py3_v2.py, les deux fonctions importantes sont init_line et read_gll. La fonction init_line permet l'ouverture du port /dev/ttyUSB0 avec un débit de 57600. De l'autre côté la fonction read_gll() permet de récupérer les valeurs de l'accéléromètre, du gyroscope et du magnétomètre. Le format de la chaine de caractère retournée est <timeMS>, <accelX>, <accelY>, <accelZ>, <gyroX>, <gyroY>, <gyroZ>, <magX>, <magY>, <magZ>.

4 Prise en main de la Raspberry

Installer la dernière version de Raspbian, ensuite activer le protocole SSH et le port UART. Pour installer la dernière version de Raspbian, on peut télécharger l'image sur le site de Raspberry et ensuite la graver avec par exemple Balena etcher. On peut aussi utiliser l'application Raspberry pi imager, qui se charge de télécharger la dernière version de Raspbian tout seul et de la graver sur une carte micro SD. Ensuite pour activer le protocole SSH, il suffit de créer un fichier nommé "ssh" dans le dossier \ boot. Enfin, pour activer le port UART, on peut utiliser l'interface graphique afin de modifier les paramètres par défaut ou utiliser l'utilitaire "raspi-config".

5 L'adaptateur 5v

L'adaptateur 5V, BEC Modelcraft sbec-26V **(Annexe 5)**, nous permettra de fournir un pur 5V à la Raspberry. Cet adaptateur est capable de fournir une tension de sortie de 6V ou de 5V. Etant donné que la tension fournie par défaut est 6V, il faudra faire des réglages, afin d'obtenir une tension de sortie de 5V. Quand le cavalier est connecté au contact extérieur médian et supérieur, la tension de sortie du récepteur et des servos raccordés est de 6 V **(voir figure 2, illustration A)**. Quand le cavalier est connecté au contact extérieur médian et inférieur, la tension de sortie du récepteur et des servos raccordés est de 5 V **(voir figure 2, illustration B)**.

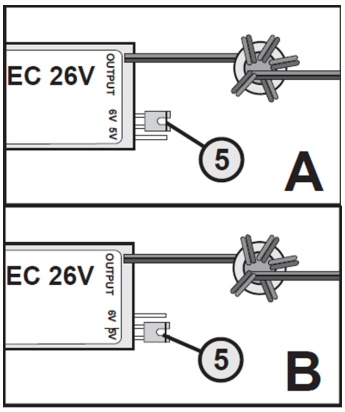


FIGURE 3: Adaptateur 5V ou 6V

6 Prise en main des ESC

Le modèle utilisé est l'HOBBYWING QUICRUN WP-1060, BRUSHED SBEC W **(Annexe 6)**. Une étape importante à chaque démarrage de l'ESC est la calibration de la plage des gaz. Cette étape permet de définir la valeur minimale des valeurs (gaz). Tout d'abord à l'allumage de l'ESC, on entend un nombre de bip qui dépend du voltage et du type de batterie. Pour une LIPO 3S, en entend 3 bips courts. Ces bips permettent de vérifier le bon fonctionnement de la connexion entre la batterie et l'esc.

Lorsque la calibration est bien effectuée, on entend un long bip après. Pour calibrer la plage des gaz, il faut envoyer via la commande des gaz, une première commande pendant 3s qui sera le point neutre puis une deuxième commande à la suite pendant 3s également qui définira la valeur minimale des gaz.

Deuxième partie

FAIRE UN DDBOAT TELECOMMANDABLE

1 Liste du matériel supplémentaire

— Taranis qx7
— FrSky model x8r version : EU
— Arduino Léonardo

2 Commande des moteurs par la Raspberry via l'Arduino

2.1 Branchement

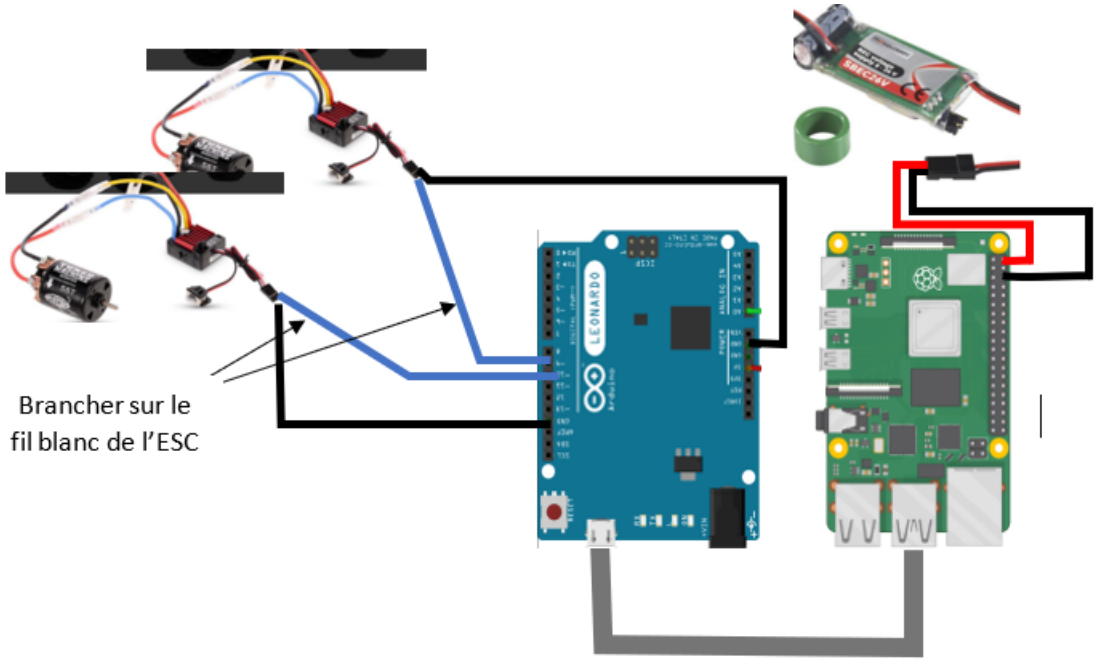


FIGURE 4: Branchement

2.2 Envoie et réception de données

La ligne 1 importe la librairie (cf. ref. [2]) permettant de controler les servo-motteurs. la ligne 6 initialise une variable qui permet de connaître la position du bouton SD de la télécommande. Pour paramétrer la valeur minimale des moteurs à 70, on utilise la commande à la ligne 70.

De la ligne 30 à la ligne 42, les instructions conditionnelles permettent de connaître la position des boutons, la première indique que le bouton est en position basse, la suivante que le bouton est au milieu et la dernière que le bouton est haut.

Ensuite, deux fonctions à savoir map et runMotor sont définies. La première permet de convertr une valeur de l'intervalle

$$a, b$$

dans l'intervalle

$$c, d$$

. Quant à la seconde, elle permet d'envoyer la valeur cmdl au moteur de gauche et la valeur cmdr au moteur de droite.

La suite du code peut être divisée en deux parties. La ligne 63 à la ligne 86, concerne le pilotage des moteurs grâce à la Raspberry par l'intermédiaire de l'Arduino. La suite du code traite du pilotage des moteurs grâce à la télécommande.

A la ligne 63, on fait la lecture des données, si la longueur est supérieur à 2,cela signifie que la donnée a été envoyée par la fonction send_arduino_cmdl_cmdr. Ensuite on extrait les vitesses cmdl et cmdr aux lignes 69 et 70.

L'instruction **else** à la ligne 7 permet la lecture de la vitesse des moteurs par l'utilisateur. Lorsque data.length ≤ 2 alors la donnée est envoyée par get_arduino_cmdl_cm et donc l'utilisateur veut les vitesses cmdl et cmdr.

La deuxième partie qui traite de la commande des moteurs par la télécommande commence par l'initialisation de la position du boutuon SD à la ligne 86. Ce bouton permet de changer les modes de pilotage. Lorsque SD est au mileu on permet le pilotage du bâteau par la télécommande

A la ligne 88, on lit les valeurs envoyés sur les canaux CH4, CH32, et SD.

Après la lecture des valeurs, on peut maintenant les envoyer aux moteurs. La ligne 101 fait tourner le bâteau à droite, la ligne 104 fait quant à elle, fait tourner le bâteau à gauche. Enfin la dernière structure **else** permet de savoir si la commande CH3 est actionnée. Et lorsque cette commande CH3 est actionnée le bâteau avance en ligne droite du bâteau.

```
#include <Servo.h>

float rc_pulse3, rc_pulse5, rc_pulse6, motor_pwm9;
Servo esc1;
Servo esc2;
int positionchannel;

void setup() {

  esc1.attach(9);
  esc2.attach(10);
  delay(15);
  Serial.begin(115200);
  esc1.write(0);
  esc2.write(0);
  delay(3000);
  esc1.write(70);
  esc2.write(70);
  delay(5);

  pinMode(3, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
}

int position(int val){
  // position SA, SB, SC, SD, SF, min = 979, middle = 1480, max = 1992
  // ces valeurs oscillent dans un intervalle de +- 10 autour de la valeut
   reel

  if ( val > 950 and val < 1050 ) {
    return 0 ;
  }
  else if ( val > 1400 and val < 1500 ) {
    return 1 ;
  }
  else if ( val > 1900 and val < 2000 ) {
    return 2 ;
  }
  else {
    return 3; // gerer si jamais il y a une erreur
  }
}

float map(float x, float a, float b, float c, float d)
{
    return  c + (x - a) * (d - c) / (b - a) ;
}

void runMotor(float cmdl, float cmdr){
    esc1.write(cmdl);
```

```
51      esc2.write(cmdr);
52
53      Serial.print(" You sent me : " );
54      Serial.print(" data 1 : " );
55      Serial.print(cmdl );
56      Serial.print(" data 2 " );
57      Serial.println(cmdr );
58    }
59
60
61  void loop() {
62
63    if( Serial.available()>0 ){ //if( Serial.available()>0 and position(
       rc_pulse6) != 1
64        String data = Serial.readStringUntil('\n');
65
66        Serial.print(data.length());
67        if ( data.length() > 2 ) {
68          int index = data.indexOf('$');
69          String data1 = data.substring(0,index);
70          String data2 = data.substring(index+1);
71          //float cmdl = data1.toFloat();
72          //float cmdr = data2.toFloat();
73          float cmdl = map ( data1.toFloat() , 0 , 100, 70, 80 );
74          float cmdr = map ( data1.toFloat() , 0 , 100, 70, 80 );
75          runMotor(cmdl, cmdr);
76        }
77        else {
78          Serial.print(" les vitesses sont : es1 ") ;
79          Serial.print(esc1.read());
80          Serial.print(" esc 2 : " );
81          Serial.println(esc2.read());
82        }
83    }
84
85    rc_pulse6 = pulseIn(6, HIGH);
86    positionchannel = position(rc_pulse6) ;
87
88    while( positionchannel == 1 ) {
89      rc_pulse3 = pulseIn(3, HIGH);
90      rc_pulse5 = pulseIn(5, HIGH);
91      rc_pulse6 = pulseIn(6, HIGH); // SD
92
93      Serial.print(rc_pulse3);
94      Serial.print("   rc_pulse5 ");
95      Serial.print(rc_pulse5);
96      Serial.print(" rcpulse6 ");
97      Serial.println(rc_pulse6);
98
99      if (position(rc_pulse3) == 1 ){
100       if ( rc_pulse5 > 1480 ){
```

```
        esc1.write(map ( rc_pulse5 , 1500 , 2000, 70, 80 ));
      }
      else{
        esc2.write(map ( -rc_pulse5 , -1500 , -980, 70, 80 ));
      }
    }
    else {
      esc1.write(map ( rc_pulse3 , 1500 , 2000, 70, 80 ));
      esc2.write(map ( rc_pulse3 , 1500 , 2000, 70, 80 ));
    }
    positionchannel = position(rc_pulse6) ;
}
}
```

La fonction signal_handler permet de mettre les moteurs à 0 à la suite d'un **CTR+C**.

La ligne 14 permet l'ouverture de la voie série. A la ligne 17, on récupére les données présentes sur la voie série, dans notre cas *data* est nulle.

La fonction send_arduino_cmd_motor permet de contrôler la vitess des moteurs. Elle Envoie des vitesses sous la forme "cmdl$cmdr$\n", et à la récuperation de la donnée, l'arduino utilisera le caractèe "$" pour séparer les deux vitesses cmdl et cmdr.

la derniere fonction get_arduino_cmd_motor permet de récuperer les vitesses des moteurs. Tout d'abord on envoie sur le port série le caractère ("C") qui à une longueur inférieur à 2. A la récupération de cette donnée, l'arduino renvoie les vitesses des moteurs.

```
1  #!/usr/bin/env python3
2
3  import serial,time
4  import signal, sys
5
6  def signal_handler(sig, frame):
7      print('You pressed Ctrl+C!')
8      arduino = serial.Serial('/dev/ttyACM0',115200,timeout=1.0)
9      data_arduino = send_arduino_cmd_motor(arduino,0,0)
10     sys.exit(0)
11
12
13 def init_arduino_line():
14     arduino = serial.Serial('/dev/ttyACM0',9600,timeout=1.0)
15     #arduino = serial.Serial('/dev/ttyACM1',115200,timeout=1.0)
16     time.sleep(1.0) # wait for arduino serial line to me ready ...
17     data = arduino.readline().decode('utf-8').rstrip()
18     print ("init status",len(data),data)
19     #arduino.close()
20     signal.signal(signal.SIGINT, signal_handler)
21     return arduino,data[0:-1]
22
23 def send_arduino_cmd_motor(arduino,cmdl0, cmdr0):
24     cmdl, cmdr = cmdl0, cmdr0
25     if cmdl < 0:
26         cmdl = -cmdl
27     if cmdr < 0:
28         cmdr = -cmdr
29     strcmd = "{}${}\n".format(cmdl,cmdr)
30     #strcmd = cmdl
31     #print strcmd
32     #arduino.open()
33     arduino.write(strcmd.encode())
34     #arduino.close()
```

```
35
36 def get_arduino_cmd_motor(arduino,timeout):
37     strcmd = "C\n"
38     #print strcmd
39     #arduino.open()
40     arduino.write(strcmd.encode())
41     t0 = time.time()
42     while True:
43         data = arduino.readline().decode('utf-8').rstrip()
44         if data:
45             #print data.rstrip('\n')
46             break
47         if (time.time()-t0) > timeout:
48             break
49     #arduino.close()
50     #arduino.flushInput()
51     return data[0:-1]
52
53 def get_arduino_cmd_motor_ones(arduino):
54     strcmd = "C\n"
55     #print strcmd
56     #arduino.open()
57     arduino.write(strcmd.encode())
58     data = arduino.readline().decode('utf-8').rstrip()
59     #arduino.close()
60     #arduino.flushInput()
61     return data
62
63 if __name__ == '__main__':
64
65     arduino, data = init_arduino_line()
66
67     while True :
68
69       #send_arduino_cmd_motor(arduino,cmd,cmd)
70       send_arduino_cmd_motor(arduino, 30, 30 )
71       data = get_arduino_cmd_motor_ones(arduino)
72       print(data)
73
74
```

3 Commande des moteurs via la télécommande

3.1 Branchement

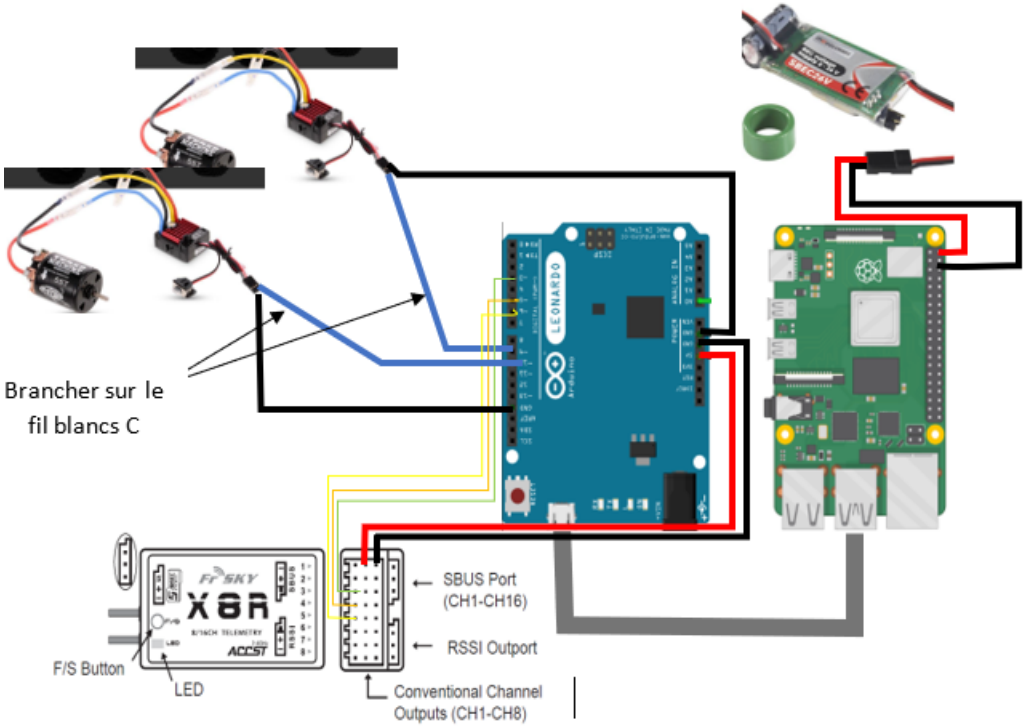


FIGURE 5: Branchement

3.2 Binder la télécommande et le récepteur

Binder (appareiller) est le processus d'association unique d'un récepteur particulier à un module émetteur. Le récepteur peut être lié qu'à une seule télécommande tandis que la télécommande peut être liée à plusieurs récepteurs. Avant de binder l'émetteur et le récepteur, il faut d'abord créer un modèle pour le récepteur. Les étapes pour la création d'un modèle sont les suivantes :

- Tout d'abord faire un appuie court sur le bouton « Menu », ensuite passer sur une ligne libre.
- Faire un appuie long sur « Entrer », et la possibilité de créer un modèle ou de restaurer un modèle apparaitra.
- Puis « Entrer » sur « créer un modèle ».
- Faire « Entrer » sur « Multi » (pour multi copter).
- Appuyer une première fois sur « Entrer » et enfin faire un appuie long sur « Entrer » pour confirmer, puis faire « Exit ».

Après avoir créé le modèle, on peut maintenant terminer la procédure de liaison. Tout d'abord il faut aller dans le menu via le bouton « Menu ». Sélectionnez le modèle pour lequel on veut binder le X8R. (Modèle sélectionné à l'aide de l'astérisque « * »). Puis allez dans la page « CONFIGURATION » (pages 2/13) en appuyant sur le bouton « PAGE » (astuce : restez appuyé sur le bouton « PAGE » vous fera revenir d'une page en arrière).

Une fois dans la page « CONFIGURATION » il faut descendre dans celle-ci en utilisant le « SCROLL » jusqu'à la ligne « MODE » du « HF interne » (le HF externe servira dans la configuration d'un module externe).

Il y a plusieurs modes disponibles, mais pour binder le X8R il faut se mettre en « D16 ». Après se rendre à la ligne RxNum. Il y a trois possibilités sur la ligne à droite, le premier est un nombre qui correspond au numéro de récepteur qu'on veut binder pour ce modèle. Le deuxième « Bind » est celui qui nous s'intéresse. C'est cette option qui nous permettra de lancer la procédure de bind entre le X8R et la Taranis . Faire entrer sur « Bind », puis entrer sur l'option « CH1-8 Télem ON » (ce qui enclenche la procédure d'appareillage ». La télécommande

emmétrera un petit bip périodique pour dire qu'il est en attente d'association.

Pendant que le récepteur n'est pas alimenté, maintenir appuyer le bouton F/S (Figure 2) avec un petit tournevis puis, toujours en maintenant, alimenter le récepteur. Attendre 6s environ, la LED s'allumera en rouge et se mettra à clignoter rapidement. Débrancher le récepteur et faire « Exit » sur l'émetteur. Enfin brancher le récepteur, la LED doit être verte.

FIGURE 6: Récepteur FrSky

Options importantes de la télécommande

Comme option, on s'intéressa à la page 4 et à la page 5. La page 4 (les entrées) permet l'assignation des boutons physiques à un nom de fonction. Aller sur une nouvelle ligne, ensuite faire un appuie long sur « Entrée », un nouveau menu s'ouvrira, puis choisir le nom de l'entrée (par exemple "mod"). Après au niveau de source il faut choisir le bouton qui exécute le mode (SA, SB, SC, SD, SH, SF, ...) directement en actionner le bouton. Puis faire « Exit ». Quant à la Page 5 (le mixeur), elle permet d'affecter les canaux ch1, 2, 3 à une fonction. Aller sur une nouvelle ligne, ensuite faire un appuie long sur « Entrée », au niveau de source choisir la fonction que l'on veut dans notre cas dans notre cas il s'agit de "mod". A la première ligne on peut choisir aussi le nom du canal.

3.3 Envoie de commandes aux moteurs

Voir la partie III-2-b

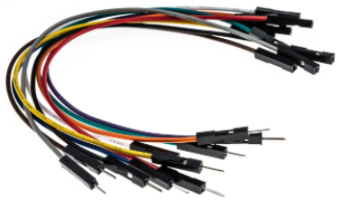
Conclusion

En définitive, ce stage fut très enrichissant pour moi. Il m'a permis de toucher à plusieurs domaines : la fabrication de robots, le developpement d'algorithmes, la théorie de localisation. Lors de ce stage, j'ai pu mettre en pratique les connaissances acquises au cours de l'année.
J'ai été aussi confronté plusieurs fois à des imprévus tels que les retards de livraison. Ces difficultés m'ont permis de grandir, et de mieux cerner le métier d'ingénieur en robotique. En effet dans ce métier, l'on peu être confronté à des imprévus qu'il faut à surpasser.

Annexes

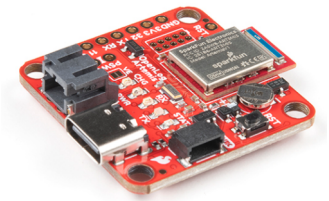
Annexe1

Annexe2



Annexe3

Annexe4



Annexe5

Annexe6

Bibliographie

[1] http://www.cedricaoun.net/eie/trames%20NMEA183.pdf/.

[2] John M. HUGHES. Arduino le guide complet une référence pour ingénieurs techniciens et bricoleurs, 2014.

[3] Eben Upton. Raspberry pi - le guide de l'utilisateur. Dunod, 2017.