Design DDBOAT

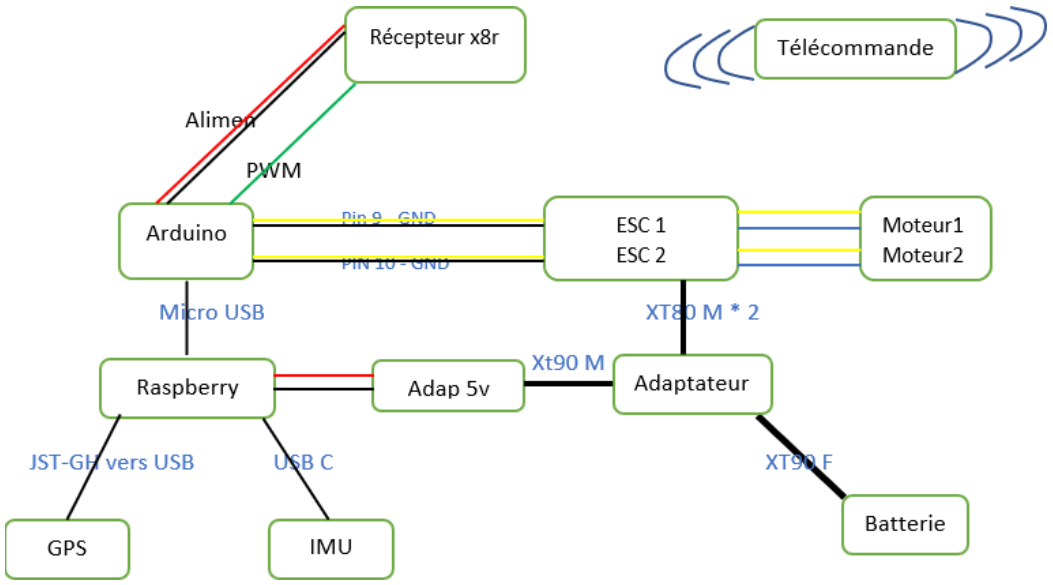
I. PART 1: DESIGN OF AN AUTONOMOUS DDBOAT

1. Design scheme

List of materials used

- SparkFun OpenLog Artemis dev 16832
- HobbyWing Quicrun WP-1060, Brushed Sbec W
- Hacker 95000331 Battery pack (LiPo)
- Voltcraft V-Charge, LiPo eco1000
- Renkforce RF-3425172, Connection cable
- POWER SUPPLY 8-26 V NOZZLE
- GPS micro m8n UBLOX
- Raspberry pi 4
- Black ABS enclosure with cover,
- USB type C cable
- USB micro-B cable
- X8R receiver
- Taranis qx7

Architecture



DDBOAT CONNECTION DIAGRAM - REMOTE CONTROL

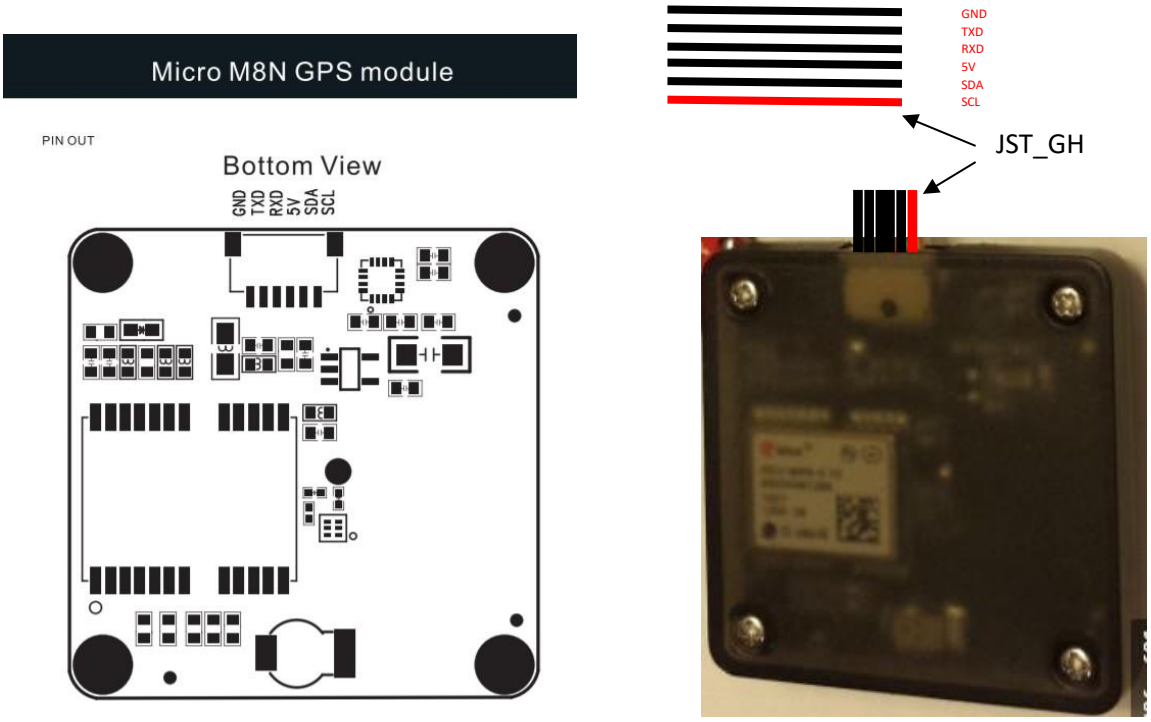
2. Getting started with GPS

2.1.Connection

For the design of the DDBOAT, we will use the M8N U-BLOCK micro GPS (Appendix 1). This GPS is delivered with a JST-GH connector (Appendix 2) which allows the GPS to be used on a PIXHAWK. In our case we will use this GPS on a Raspberry Pi 4 and therefore we will have to adapt the connector. So we have two solutions.

1st solution: use of the mini UART port

This solution consists in cutting one side of the JST-GH cable to connect it with Dupont wires (Appendix 3). Then the Dupont wires can be attached to the UART ports of the Raspberry (TX: GPIO14, RX: GPIO15). This port will be accessible through the /dev/ttyUSB0 device. Moreover, the dmesg command allows to find the name of the devices.



GPS Module Pinout Mapping

Since we want to use the mini UART port on the Raspberry, the GPS pins involved are GND, TXD, RXD, and 5V. The SDA and SCL pins are used for communication via the i2c bus. After soldering the Dupont wires to the relevant pins, the 5v Dupont wire is connected to one of the 5v pins on the Raspberry. Then we connect the TXD and the RXD respectively to the RX (GPIO15) and the TX (GPIO 14) of the Raspberry. Finally, we connect the GND to one of the GNDs on the Raspberry.

2nd solution: use the USB ports of the Raspberry

This solution has not been tested. It consists in soldering the same pins, namely GND, TXD, RXD and 5V of the JST-GH cable to a USB port.

2.2- GPS data retrieval by the Raspberry

Recovery with cutecom or putty

The steps to follow to retrieve data via cutecom or putty are: launch the application (cutecom, putty, ...), select the device and then enter the baudrate. And finally click on OK.

Retrieving values with a python script

To retrieve the GPS data, the gps_drivers_py3.py driver must be used. First of all, you have to import it into the script and then use the different functions.

In the gps_drivers_py3.py script, the two important functions are init_line and read_glll. The init_line function allows the opening of the /dev/ttyUSB0 port with a data rate of 9600. On the other hand the read_gll function allows to retrieve NMEA frames. From the NMEA frames of the GPS, we extract the latitude and longitude.

Each frame begins with the **$** character followed by a group of 2 letters for the receiver identifier. (Non-limiting):
· **GP** stands for Global Positioning System.
· **LC** Loran-C receiver.
· **OM** Omega Navigation receiver.
· **II** Integrated Instrumentation (eg. AutoHelm Seatalk system).

Then a group of 3 letters for the frame identifier.

· **GGA**: for GPS Fix and Date.
· **GLL**: for Geographical Positioning Longitude - Latitude.
· **RMC**: for specific minimum usable data.
· **GSA**: for DOP and active satellites.
· **GSV**: for visible satellites.
· **VTG**: for Direction (heading) and travel speed (in knots and Km/h).

This is followed by a number of **fields** separated by a "**comma**". The role of the comma is to be the field separator, which allows the de-concatenation of data in the data processing program, calculator, browser.
As a frame we have :

The frame:

GGA Acquisition data from the FIX - GPS.
$GPGGA,123519,4807.038,N,01131.324,E,1,08,0.9,545.4,M,46.9,M, , *42
123519 = FIX acquisition at 12:35:19 UTC
4807.038,N = Latitude 48 deg 07.038' N
01131.324,E = Longitude 11 deg 31.324' E

The frame: GLL

Geographical position - Longitude / Latitude - GPS
$GPGLL,4916.45,N,12311.12,W,225444,A
4916.46,N = Latitude 49 deg. 16.45 min. N = North.
12311.12,W = Longitude 123 deg. 11.12 min. West
225444 = Acquisition of the Fix at 22:54:44 UTC

The frame: RMC

Recommended minimum GPS specification data
225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68 225446 = Time of Fix 22:54:46 UTC
A = Navigation software warning (A = OK, V = warning)
4916.45,N = Latitude 49 deg. 16.45 min North
12311.12,W = Longitude 123 deg. 11.12 min West
This "minimal" data is most often used in simple GPS navigation programs.

For example, if we use the GPRMC frame, just before the letter N and W, we find the latitude and longitude respectively.

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

This function extracts the longitude and latitude values.

3. Getting started IMU

3.1. Firmware installation

We will use the OPENLOG ARTEMIS DEV 16832 (Appendix 3). First we need to install the firmware, and then we can retrieve the data from the IMU.

To start, you need to connect the IMU to the PC using a USB cable, type A to type C, and have the Arduino IDE installed on the operating system. We will be using the Windows system, however the firmware installation can also be done using the GNU/Linux system.

To install the firmware, the steps to follow are as follows:

- *Downloading the libraries*: there are two folders, the first one to download from https://learn.sparkfun.com/tutorials/9dof-razor- imu-m0-hookup-guide (go to ¾ of the page, to the Library and example firmware section and click on Dowload the flashstorage arduino library) and the second one is a cloning of the repository https://github.com/lebarsfa/SparkFun_MPU-9250-DMP_Arduino_Library . This information is for information only, otherwise these libraries will be downloaded and thus directly accessible in the folder I will send you. In this case, the files will have to be copied and pasted in the folder: Quick access -> Documents -> Arduino -> Library.

- *Installation of the Sparkfun Apollo3 Boards in the arduino IDE*: In order to install this board,the following must be added https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_sparkfun_index.jsin : File-> Preferences -> Additionnal Boards Manager URLs. Next, make sure to choose SparkFun Apollo3 → SparkFun RedBoard Artemis ATP.
- *Install the SparkFun ICM 20948 IMU library: Sketch* ->Include Library -> Manage Libraries. In the search bar enter "SparkFun ICM "and click on "install". For more details on installing libraries, the following link explains the steps: https://learn.sparkfun.com/tutorials/installing-an-arduino- library
- The last step is to clone the following repository https://github.com/Razor-AHRS/razor-9dof-ahrs, (which I will also put in the final folder). In the file razor\razor_imu_9dof\src\Razor_AHRS\Razor_AHRS.ino, you have to uncomment the line that concerns our razor model, i.e. line 230 *(#define HW VERSION_CODE 16832 // SparkFun "OpenLog Artemis" version "DEV-16832").*

After all these steps, all that remains is to compile and upload.

3.2. Reading and retrieving data

For a first test, you can open the serial monitor of the Arduino IDE, normally you should see data scrolling. If the data is unreadable, this is normal, you should change the bauderate to 57600, the data will be a little more understandable.

To get the IMU data directly through a script, you will have to use the imu_drivers_py3_v2.py driver. First of all, you have to import it into the script and then use the different functions.

In the imu_drivers_py3_v2.py script, the two important functions are init_line and read_gll. The init_line function opens the port
On the other hand, the read_gll() function retrieves the accelerometer, gyroscope and magnetometer values. The format of the returned string is <timeMS>,
<accelX>, <accelY>, <accelZ>, <gyroX>, <gyroY>, <gyroZ>, <magX>, <magY>, <magZ>.

4. The 5v adapter

The 5V adapter, BEC Modelcraft sbec-26V (Appendix 5), will allow us to supply pure 5V to the Raspberry.

This adapter is capable of providing an output voltage of 6V or 5V. As the default voltage is 6V, it is possible to use the adjustments will have to be made in order to obtain an output voltage of 5V.

When the jumper is connected to the middle and top external contact, the output voltage of the receiver and the connected servos is 6 V (see figure 1, illustration A).
When the jumper is connected to the middle and bottom external contact, the output voltage of the receiver and the connected servos is 5 V (see figure 1, illustration B).

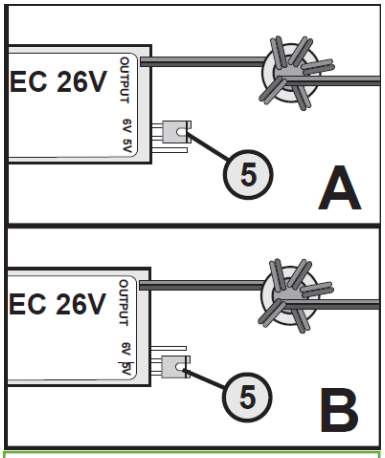


Figure 1

5. Getting to grips with ESCs,

The model used is the *HobbyWing Quicrun WP-1060, Brushed Sbec W* (Appendix 6). An important step at each ESC start-up is the calibration of the throttle range. This step defines the minimum value of the values (throttle).

First of all, when the ESC is switched on, a number of beeps are heard, depending on the voltage and the type of battery. For a LIPO 3S, 3 short beeps are heard. These beeps are used to check that the connection between the two components, the battery and the ESC, is working properly.

When the calibration is successful, a long beep is heard afterwards. To calibrate the throttle range, send a first command for 3s via the throttle control, which will be the neutral point, and then a second command in succession for 3s, which will define the minimum throttle value.

6. Getting started with the Raspberry

Install the latest version of Raspbian, then enable the SSH protocol and the UART port.

To install the latest version of Raspbian, you can download the image from the Raspberry website and then burn it with for example Balena etcher. You can also use the Raspberry pi imager application, which downloads the latest version of Raspbian and burns it to a micro SD card. Then to activate the SSH protocol, you just have to create a file named "ssh" in the \boot folder.
Finally, to activate the UART port, one can use the graphical interface to modify the default settings or use the "raspi-config" utility as shown in the following link https://www.framboise314.fr/utiliser-le-port-serie- du-raspberry-pi-3-et-du-pi-zero/.

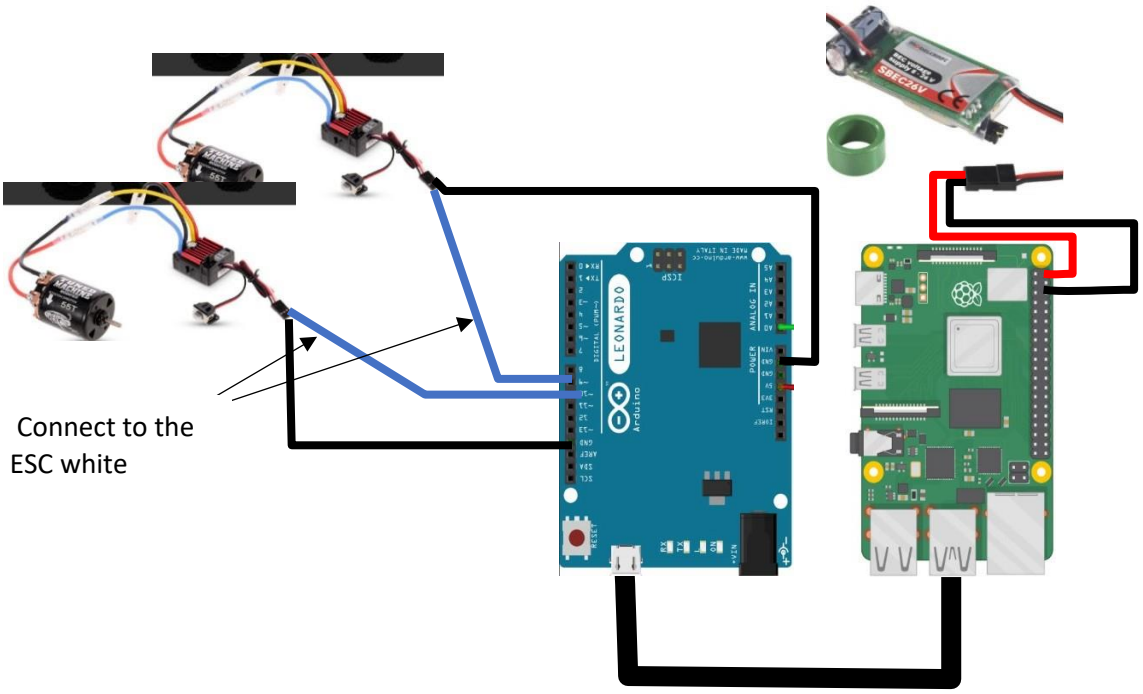
II. Part 2: Making a remote-controlled DDBAT

1. List of additional equipment

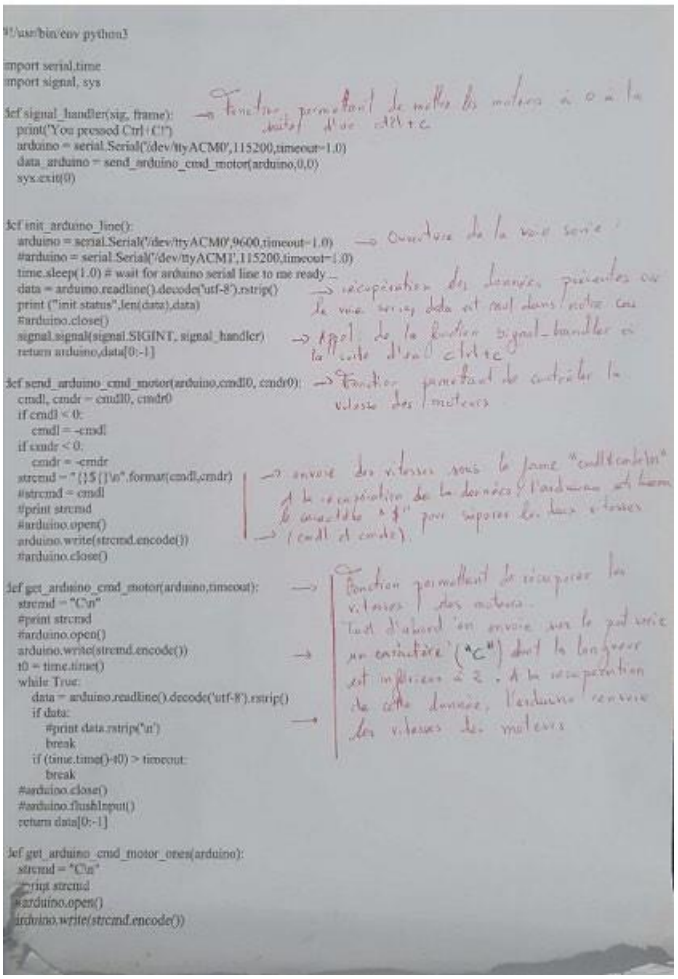
- Taranis qx7
- FrSky model x8r version : EU
- Arduino Leonardo

2. Control of the motors by the Raspberry via the Arduino

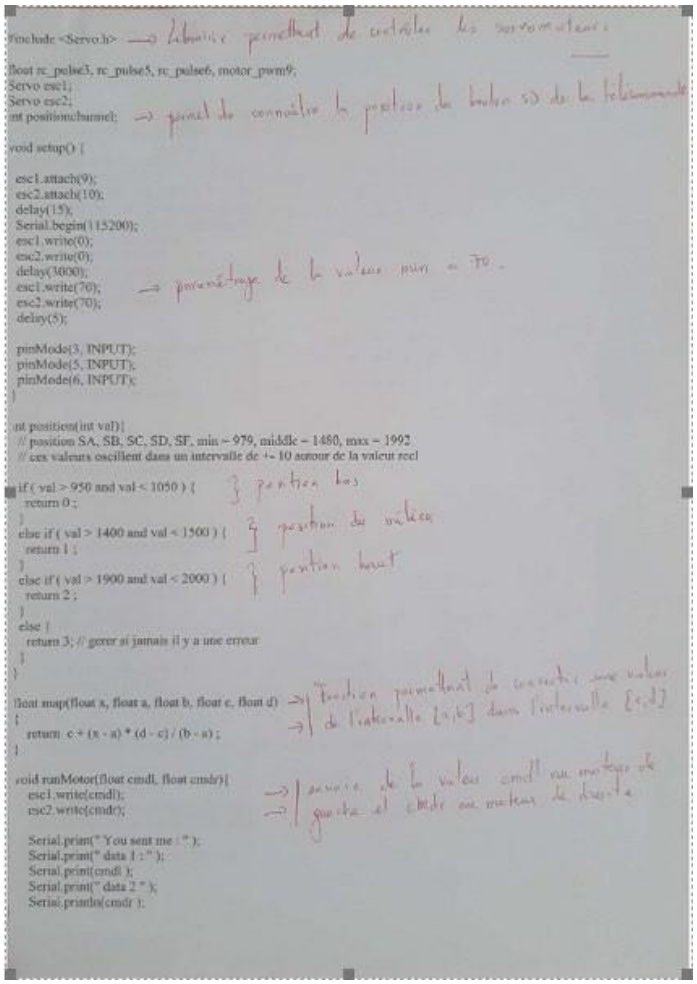
2.1. Connection



2.2. Sending and receiving data



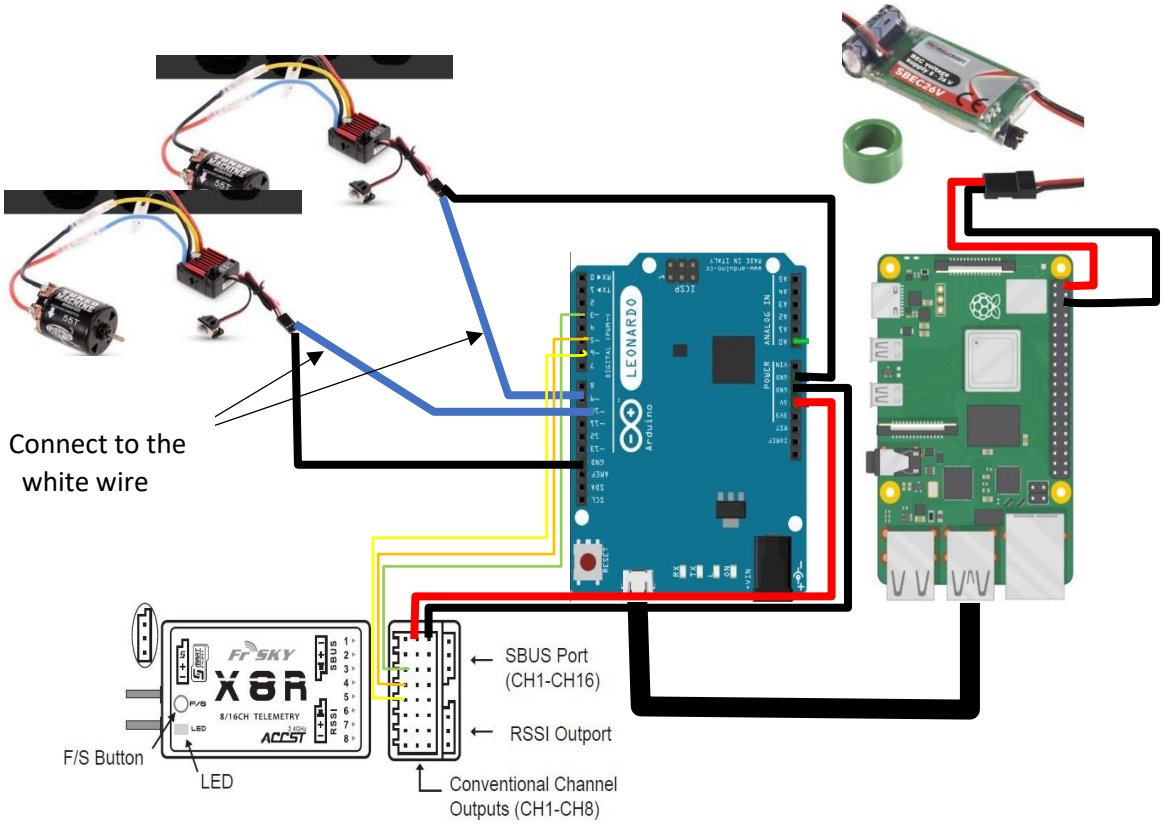
```
#!/usr/bin/env python3

import serial,time
import signal, sys

def signal_handler(sig, frame):
    print('You pressed Ctrl+C!')
    arduino = serial.Serial('/dev/ttyACM0',115200,timeout=1.0)
    data_arduino = send_arduino_cmd_motor(arduino,0,0)
    sys.exit(0)


def init_arduino_line():
    arduino = serial.Serial('/dev/ttyACM0',9600,timeout=1.0)
    #arduino = serial.Serial('/dev/ttyACM1',115200,timeout=1.0)
    time.sleep(1.0) # wait for arduino serial line to me ready
    data = arduino.readline().decode('utf-8').rstrip()
    print ("init status",len(data),data)
    #arduino.close()
    signal.signal(signal.SIGINT, signal_handler)
    return arduino,data[0:-1]

def send_arduino_cmd_motor(arduino,cmdl0, cmdr0):
    cmdl, cmdr = cmdl0, cmdr0
    if cmdl < 0:
        cmdl = -cmdl
    if cmdr < 0:
        cmdr = -cmdr
    strcmd = "{},{}\n".format(cmdl,cmdr)
    #strcmd = cmdl
    #print strcmd
    #arduino.open()
    arduino.write(strcmd.encode())
    #arduino.close()

def get_arduino_cmd_motor(arduino,timeout):
    strcmd = "C\n"
    #print strcmd
    #arduino.open()
    arduino.write(strcmd.encode())
    t0 = time.time()
    while True:
        data = arduino.readline().decode('utf-8').rstrip()
        if data:
            #print data.rstrip('\n')
            break
        if (time.time()-t0) > timeout:
            break
    #arduino.close()
    #arduino.flushInput()
    return data[0:-1]

def get_arduino_cmd_motor_oneo(arduino):
    strcmd = "C\n"
    #print strcmd
    #arduino.open()
    arduino.write(strcmd.encode())
```



```
#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 valeur 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);
    esc2.write(cmdr);

    Serial.print(" You sent me : " );
    Serial.print(" data 1 : " );
    Serial.print(cmdl );
    Serial.print(" data 2 " );
    Serial.println(cmdr );
```

3. Motor control via remote control

3.1. Connection



3.2. Binding the remote control and the receiver

Binder is the process of uniquely associating a particular receiver with a transmitter module. The transmitter module can be linked to several receivers. However, a receiver can only be bound to one transmitter module.

Before bind the transmitter and receiver, a model for the receiver must first be created.
The steps for creating a model are as follows:

- First press the "Menu" button, then move to a free line.
- Press and hold "Enter", and the option to create a template or restore a template will appear
- Then "Enter" on "create a model".
- Press "Enter" on "Multi" (for multi copter)

- Press "Enter" once and then press and hold
  Press "Enter" to confirm, then press "Exit".

After creating the model, the linking procedure can now be completed. First of all, go to the Menu via the "Menu" button. Select the model for which you want to bind the X8R. (Model selected with the asterisk "*"). Then go to the "CONFIGURATION" page (pages 2/13) by pressing the "PAGE" button (tip: pressing and holding the "PAGE" button will take you back one page).

Once in the "CONFIGURATION" page, you must scroll down using the "SCROLL" to the "*MODE*" line of the "Internal HF" (the external HF will be used to configure an external module).
There are several modes available, but to bind the X8R you need to be in *« D16 »*.

Then go to the RxNum line. There are three possibilities on the line to the right, the first is a number that corresponds to the receiver number we want to bind for this model. The second "Bind" is the one we are interested in. This is the option that will allow us to start the bind procedure between the X8R and the Taranis.

Press "Bind", then press the "CH1-8 Telem ON" option (which starts the pairing procedure). The remote control will emit a small periodic beep to indicate that it is waiting to be paired.

While the receiveris not powered, press and hold the F/S button (Figure 2) with a small screwdriver and,while holding, power the receiver. Wait for about 6 seconds, the LED will light up red and flash rapidly. Unplug the receiver and press "Exit" on the transmitter.
Finally connect the receiver, the LED should be green.

Figure 2: FrSky receiver

The following video https://www.youtube.com/watch?v=xL1QglHdlYc summarises the steps to follow to bind the FrSky x8r receiver and a Taranis.

Important remote control options

As an option, we looked at page 4 and page 5.

Page 4 (inputs) allows the assignment of physical buttons to a function name. Go to a new line, then press and hold "Input", a new menu will open, then choose the name of the input (for example "mod"). Then at the source level you have to choose the button that executes the mode (SA, SB, SC, SD, SH, SF, ...) directly by pressing the button. Then press "Exit".

As for Page 5 (the mixer), it allows you to assign channels ch1, 2, 3 to a function. Go to a new line, then press and hold "Enter", at the source level choose the function you want, in our case it is "mod". On the first line you can also choose the name of the channel.

The following video https://www.youtube.com/watch?v=aDZjEpZ-ut0 gives you an introduction to the remote control and summarises everything you need to know to operate the remote control.

3.3. Sending control to motors

See Part III-2-b

Annexes

Appendix 1

Annex2

Annex3

Annex4

Annex5

Annex6