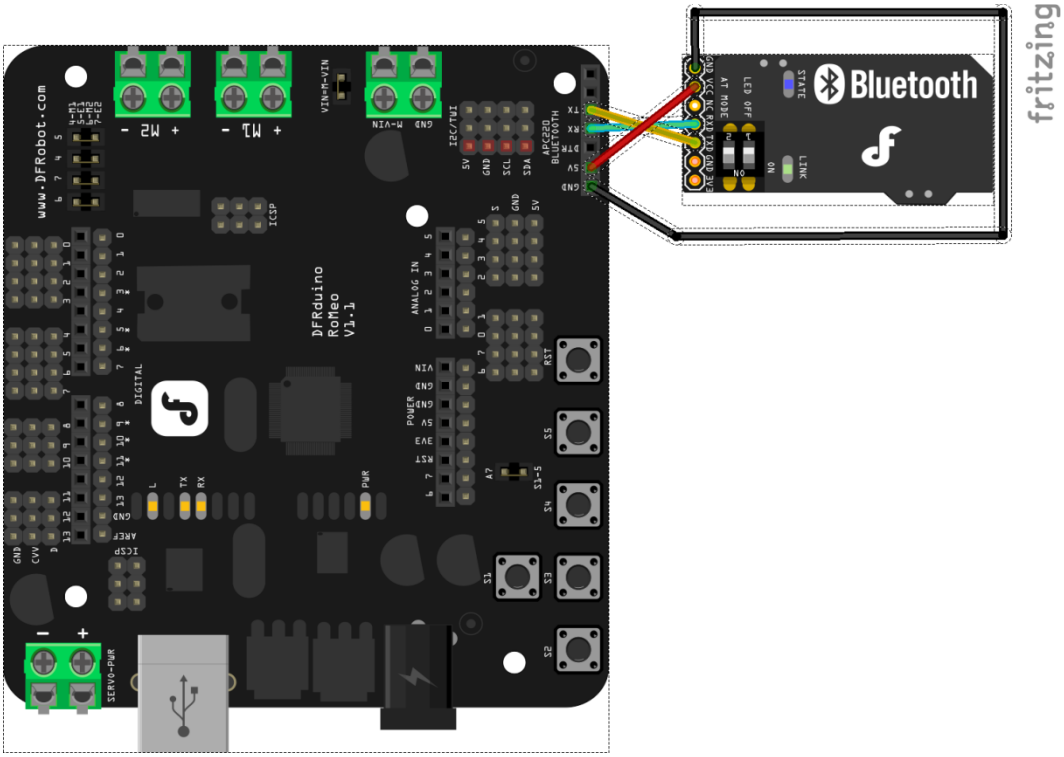
Documentation du robot

Sommaire:

2014- Fernandez Cédric – Freppaz Antoine

Bluetooth :

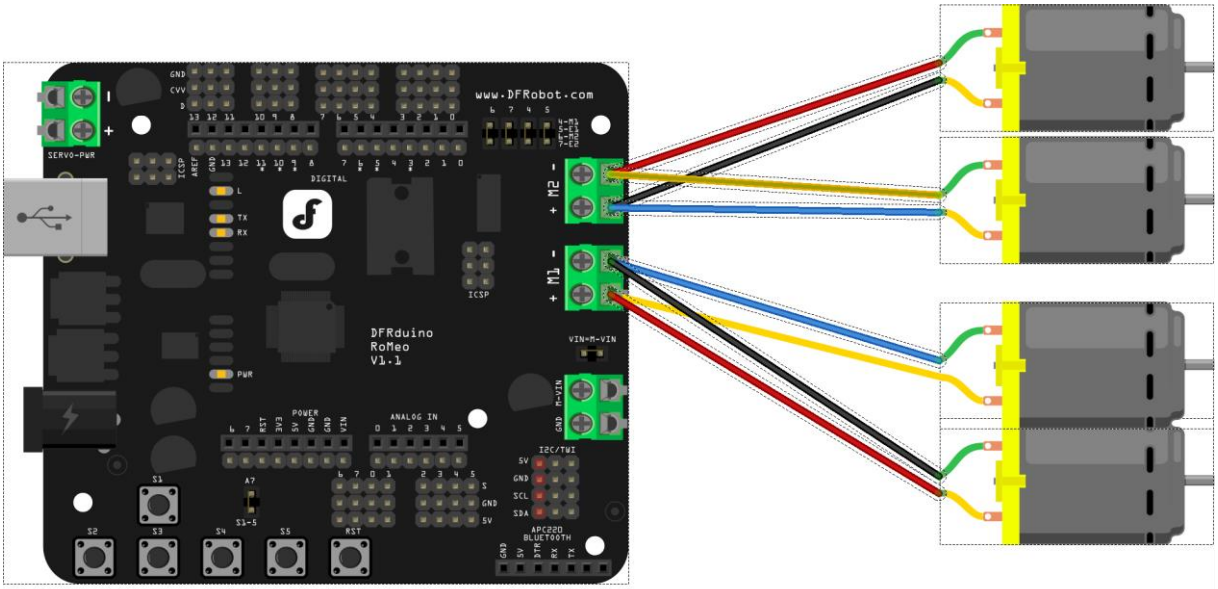
Le module se branche sur l'espace qui lui est dédié, en faisant bien correspondre les pattes Rx et Tx entre le module Bluetooth et la carte. Il faut utiliser le Serial1 afin de communiquer via Bluetooth, voir l'exemple ci-dessous :




```
//Exemple d'envoie et de réception de donnée via bluetooth

void setup(){
      Serial1.begin(9600);
}
void loop(){
      char strTrame[10] ;
      int iBclReception=0 ;
      //Envoie de donnée via bluetooth
      Serial1.write("GEII");

      //Reception de donnée via bluetooth, les caractères sont envoyée un par un,
il faut donc connaitre la taille maximum de la trame envoyé afin de pouvoir
réserver suffisamment de mémoire pour la réception*/
      if(Serial1.available() > 0)
      {
            strTrame[iBclReception] = Serial1.read();
            Serial.print(strTrame[iBclReception]);
            iBclReception++;
      }
}
```

Moteur :

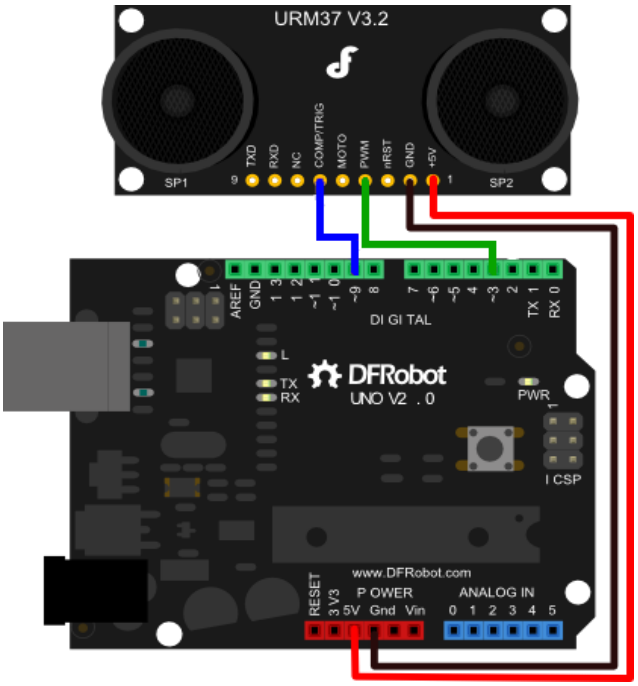
La commande des moteurs se fait côté par côté, on contrôle les deux moteurs gauches ou droit ensemble, à cause du câblage, ils ne sont pas indépendants. Pour plus d'information sur les modes de contrôles : DfRbot. Voir Annexe à propos du problème d'alimentation.




```
//Exemple de contrôle en mode PWN

#define PORT_VITESSE_M1 5
#define PORT_VITESSE_M2 6
#define PORT_DIRECTION_M1 4
#define PORT_DIRECTION_M2 7

#define AVANCER 0
#define RECULER 1

void setup(){
      pinMode(PORT_VITESSE_M1, OUTPUT);
      pinMode(PORT_VITESSE_M2, OUTPUT);
      pinMode(PORT_DIRECTION_M1, OUTPUT);
      pinMode(PORT_DIRECTION_M2, OUTPUT);
}
void loop(){

      //La vitesse des moteurs est comprise entre  0 et 255
      //Le sens de rotation des roues est ensuite géré indépendamment de
      leur vitesse
      int iPuissanceMoteur1= 200;
      int iPuissanceMoteur2= 200;

      analogWrite(PORT_VITESSE_M1, iPuissanceMoteur1);
      digitalWrite(PORT_DIRECTION_M1, AVANCER);


      analogWrite(PORT_VITESSE_M2, iPuissanceMoteur2);
      digitalWrite(PORT_DIRECTION_M2, iSensMoteur2);
}

//Ici le robot avancera en ligne droite
```

Capteur ultrason :

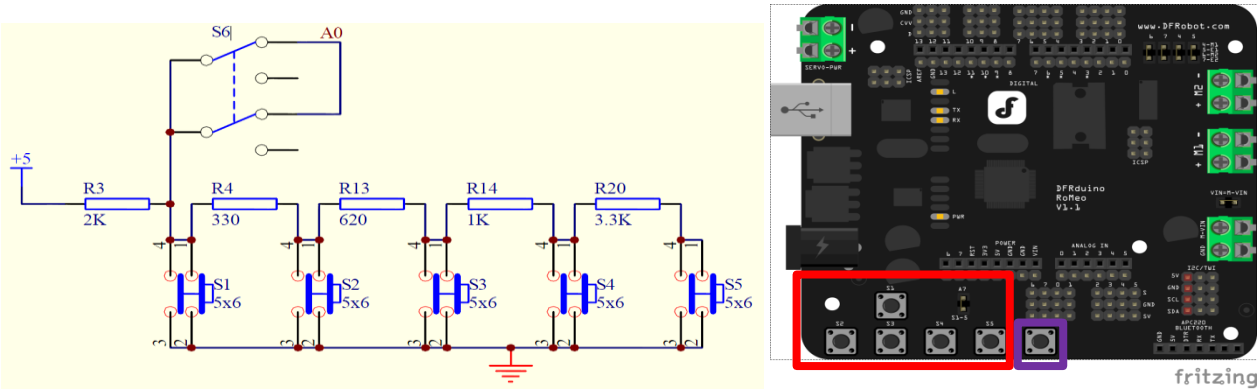
Ici le capteur est utilisé en mode passif, il attend qu'on le sollicite pour récupérer la distance.
Pour les autres modes, voir la documentation [DfRobot.](#)
Exemple de câblage pour le mode passif, les pins reliés à COMP/TRIG et PWN doivent être des PWN :




```
#define PWN_PIN 3 //Numéro de la patte arduino où est branché PWN de la carte ultrason
#define COMPTRIG_PIN 9 //Numéro de la patte arduino  où est branché COMP/TRIG de la carte ultrason

unsigned char Distance=0;
unsigned long lStart=0;
uint8_t EnPwmCmd[4]={0x44,0x22,0xbb,0x01};

void UltraSons_Setup(){
  pinMode(COMPTRIG_PIN,OUTPUT);
  digitalWrite(COMPTRIG_PIN,HIGH);

  pinMode(PWN_PIN, INPUT);

  for(int i=0;i<4;i++){
    Serial.write(EnPwmCmd[i]);
  }
  lStart=millis();
}

char UltraSons_Distance(){
    unsigned char DistanceObstacle=0;
      digitalWrite(COMPTRIG_PIN, LOW);
      digitalWrite(COMPTRIG_PIN, HIGH);

      unsigned long DistanceMeasured=pulseIn(PWN_PIN,LOW);

      if(DistanceMeasured!=50000){
                  Distance=DistanceMeasured/50;
      }
      if(Distance<256){
        DistanceObstacle=Distance;
      }
      else{
        DistanceObstacle=255;
      }
    return(DistanceObstacle);
}
```

Boutons poussoir :

La carte dispose de 5 boutons (rouge) poussoirs câblé de la manière suivante, ils activées ou non grâce ou switch noté « S1-S5 switch » (switch blanc sur la carte) sur la figure 1 :



Le bouton violet est le bouton de reset, il redémarre la carte.

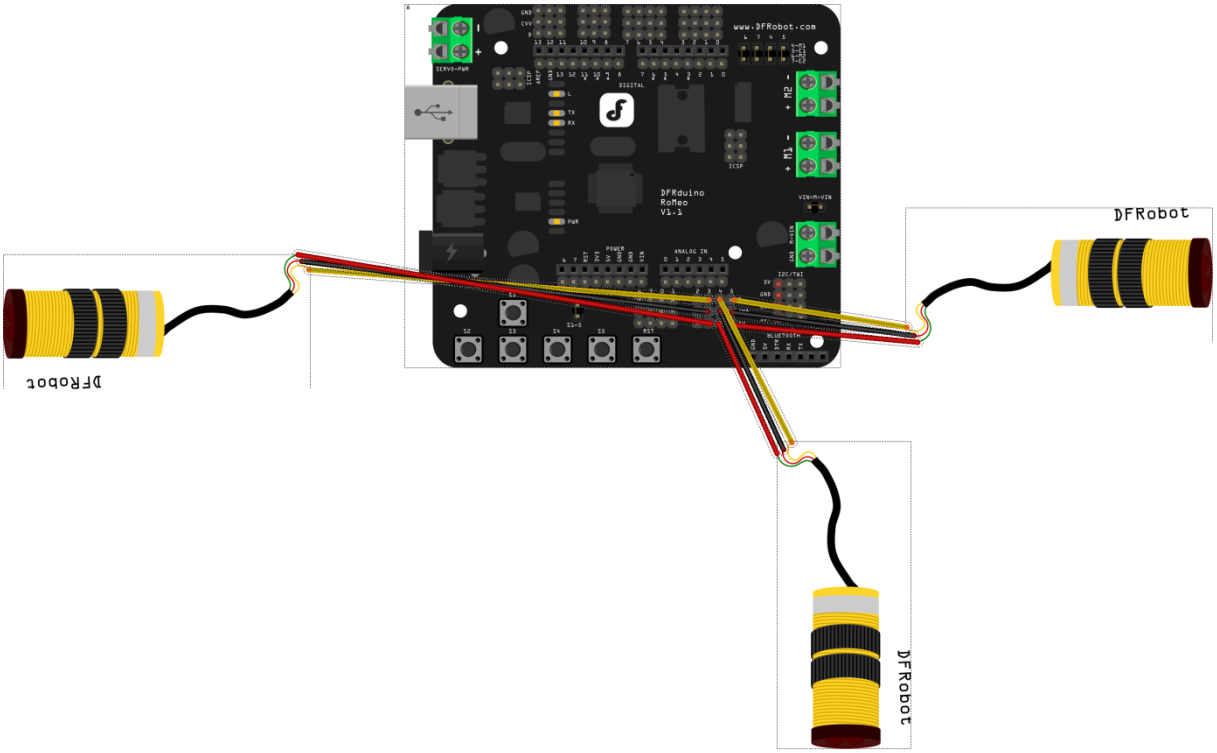
La patte A0 permet donc de savoir quel bouton est appuyé suivant la tension.

```
//Code permettant d'utiliser les boutons
int  adc_key_val[5] = {
    30, 150, 360, 535, 760
};
int NUM_KEYS = 5;

void loop() {
  static int iValeurAnalogiqueBouton = 0;
  static int iBouton = 0;
  iValeurAnalogiqueBouton = analogRead(0);// read the value from the sensor

  iBouton = get_key(iValeurAnalogiqueBouton); // convert into iBouton press
}

int get_key(unsigned int input) {
    int k;
    for (k = 0; k < NUM_KEYS; k++) {
        if (input < adc_key_val[k]) {
            return k;
        }
    }
    if (k >= NUM_KEYS)
        k = -1;
    return k;
}La valeur de iBouton dépend du bouton appuyé : 0 pour S1, 1 pour S2, 2 pour S3, 3 pour S4 et 4 pour S5
```

Capteur Infra rouge :

Le capteur infrarouge délivre suivant s'il détecte un obstacle à une certaine distance. Le seuil de détection se règle grâce à la vis qui se situe derrière le capteur. Lorsque le capteur détecte un objet, une lumière s'allume dans celui-ci, ceci permet de régler la distance de détection.

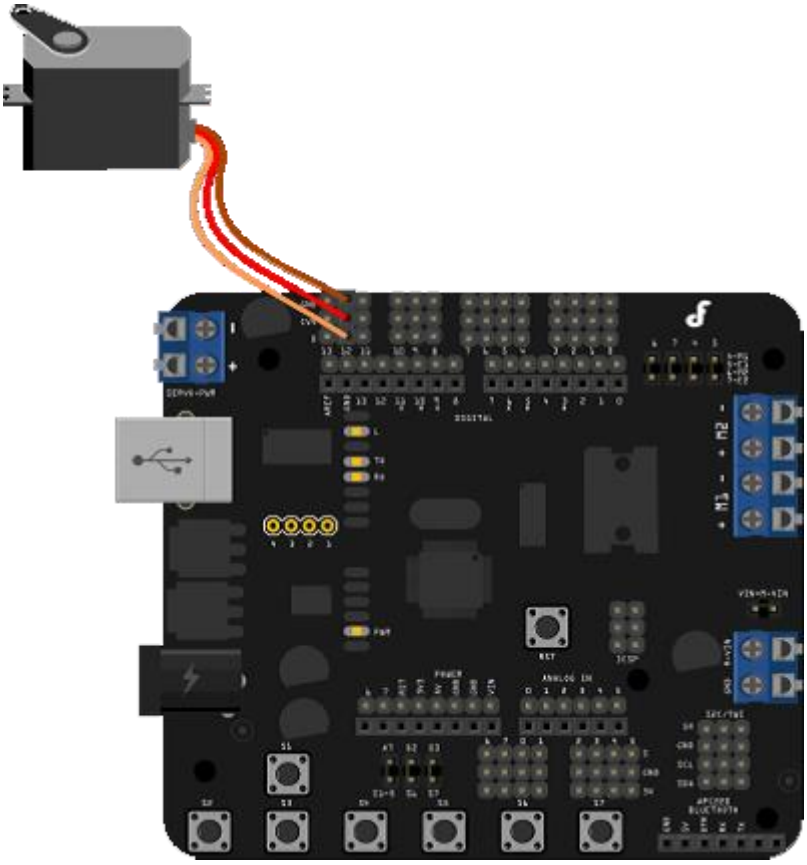
Le fil vert du capteur correspond à l'alimentation, le rouge la masse et le jaune le signal.




```
//Exemple pour un capteur
#define CAPTEUR_IR 4

void setup( void){
      pinMode(A4, INPUT);
}

void loop (void){
  iValeurAnalogiqueIRArriere = analogRead(CAPTEUR_IR);

  iIRGauche = (iValeurAnalogiqueIRGauche < 30); //Il s'agit d'un capteur
TOR branché sur un port analogique – il faut donc traiter l'information.
S'il renvoie une faible valeur (proche de 15), c'est qu'il y a un obstacle.
Sinon, il renvoie une valeur proche de 800. La distance à laquelle le
capteur détecte ou non un obstacle est réglable grâce à une petite vis
proche du fil.

}
```

Servomoteur :

Exemple de branchement issu de la documentation de la carte Romeo.



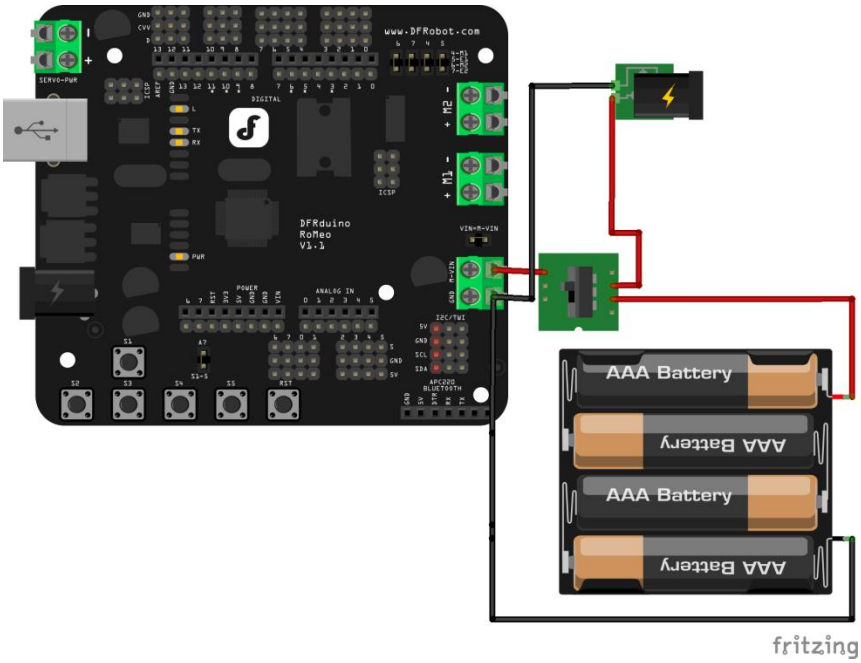
Pour plus d'information concernant l'utilisation du servomoteur voir la documentation Arduino.

Cet exemple permet de faire varier l'angle du servomoteur de 0 à 120°, le servomoteur moteur ne pouvant aller au-delà de 120° (cf documentation).

```
#include <Servo.h>

#define PIN_SERVO_MOTEUR 9 // dépend de la patte (PWN) où est branché le fil du signal PWN

Servo myservo;

void setup()
{
  myservo.attach(PIN_SERVO_MOTEUR);
}
void loop()
{
  static int iBcl=0;
  for(iBcl=0; iBcl < 120; iBcl ++){
     myservo.write(iBcl);
     delay(15);
  }
}
```

Annexe :

Concernant l'alimentation du robot, il peut être alimenté par les piles (interrupteur à gauche) ou par une alimentation externe (interrupteur à droite). L'alimentation externe est recommandée car les piles s'épuisent vite et quand elles ne peuvent plus fournir assez de puissance lorsque les moteurs forcent (puissance maximal ou rotation) la carte redémarre.



À propos du câblage de l'infrarouge, la carte utilisée sur le robot est une Romeo V2 alors que celle utilisé sur le schéma est une V1.1, il n'y a donc pas 8 broches mais bien que 6.

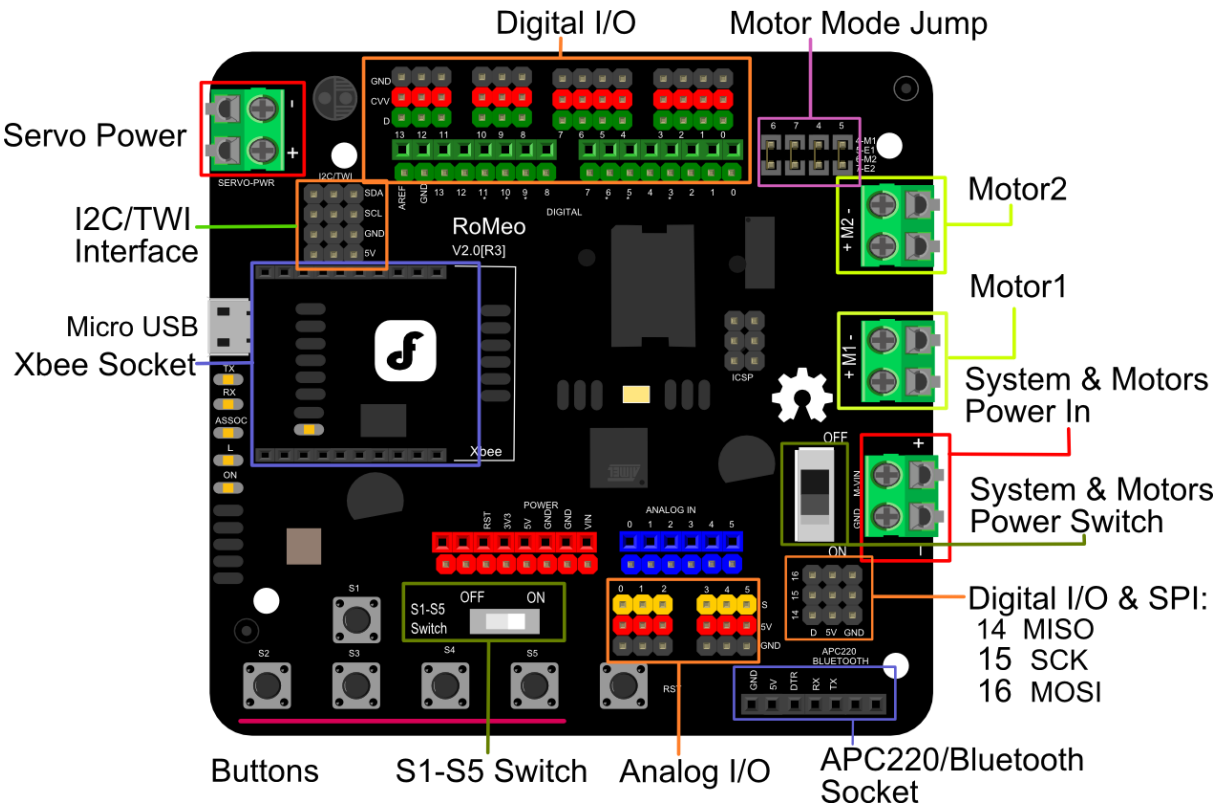
Les moteurs utilises déjà certaines sorties mais celles-ci restent accessible, on peut donc toujours brancher quelque chose dessus mais cela implique de ne pas se servir des moteurs. Voici les sorites réservées pas les moteurs suivant le mode de contrôle :

Mode PWN	
Sortie	Fonction
Digitale 4	Contrôle de la direction du moteur 1
Digitale 5	Contrôle de la vitesse du moteur 1
Digitale 6	Contrôle de la direction du moteur 2
Digitale 7	Contrôle de la vitesse du moteur 2
Mode PLL	
Sortie	Fonction
Digitale 4	Active le contrôle du moteur 1
Digitale 5	Contrôle de la direction du moteur 1
Digitale 6	Contrôle de la direction du moteur 2
Digitale 7	Active le contrôle du moteur 2

Récapitulatif des références :

Moteur	http://www.dfrobot.com/wiki/index.php/Romeo_V2-All_in_one_Controller_%28R3%29_%28SKU:DFR0225%29#PWM_Control_Mode
Bouton	http://www.dfrobot.com/wiki/index.php/Romeo_V2-All_in_one_Controller_%28R3%29_%28SKU:DFR0225%29#Example_use_of_Button_S1-S5
Capteur ultra son	http://www.dfrobot.com/wiki/index.php/URM37_V3.2_Ultrasonic_Sensor_%28SKU:SEN0001%29#Arduino_Sketch
Capteur infra rouge	http://www.dfrobot.com/wiki/index.php?title=Adjustable_Infrared_Sensor_Switch_%28SKU:SEN0019%29
Bluetooth	http://www.dfrobot.com/wiki/index.php?title=DF-BluetoothV3_Bluetooth_module_%28SKU:TEL0026%29#How_to_usea
Servomoteur	http://www.dfrobot.com/index.php?route=product/product&product_id=236

Figure 1 :



Si vous avez des questions concernant le code ou le robot en général, n'hésitez pas à envoyer un mail à Valentin.Schneider@etu.univ-savoie.fr

La dernière version du code Arduino incluant réception et calculs GPS, et une version épurée avec mode manuel fonctionnel et mode auto vide sont fournies avec ce pdf.