Club Robotique
Cycle 3 - Voiture RC

- Phase 1 - Conception 2D et 3D - ? x 1h30
- Phase 2 - Électronique et programmation
- Moteur CC - Principe de fonctionnement
- Moteur CC - Commande de moteur à courant continu
- Moteur CC - Contrôler la vitesse avec une roue encodeuse
- Le module Bluetooth HC-05
- RC Car - Voiture modélisme radiocommandée
- Code voiture RC

Phase 1 - Conception 2D et 3D - ? x 1h30

Conception 2D et 3D avec TinkerCad et OnShape

Prérequis participant :

- Travail sur ordinateur
- Travail sur feuille
- Manipulation avec la souris (clic droit, clic gauche, sélection,...)
- Manipulation de fourniture (crayon, gomme...)

Compétences Animateur :

Compétences techniques et soft skills :

- Pratique du logiciel TinkerCAD et du logiciel OnShape
- Comprendre et savoir utiliser une mise en plan
- Animation

Matériels nécessaires

logiciel, électronique, mécanique, outils, matériaux, code...

- 12 PCs (1 par participant)
  - Connexion et navigateur internet (Firefox)
  - Session invité ou nominative
- 12 moteurs à courant continue / dimensions des moteurs
- 24 Feuilles blanches
- 12 crayons de papier
- 12 gommes
- 5 tailles crayons
- 12 règles

Préparation :

Matériel par participant sur un poste PC en début de séance :

- PC allumé
- Codes d'accès session Windows/Linux
- Création d'une activité avec le tutoriel animateur TinkerCAD et OnShape
- Lien d'invitation à l'activité
- Numérotation des PCs pour que les élèves sachent quel compte utiliser
- 1 feuille + 1 crayon + 1 gommes + 1 règle par élèves
- 1 moteur à courant continu
- Temps de préparation : 5min

Documentation / Tutoriels :

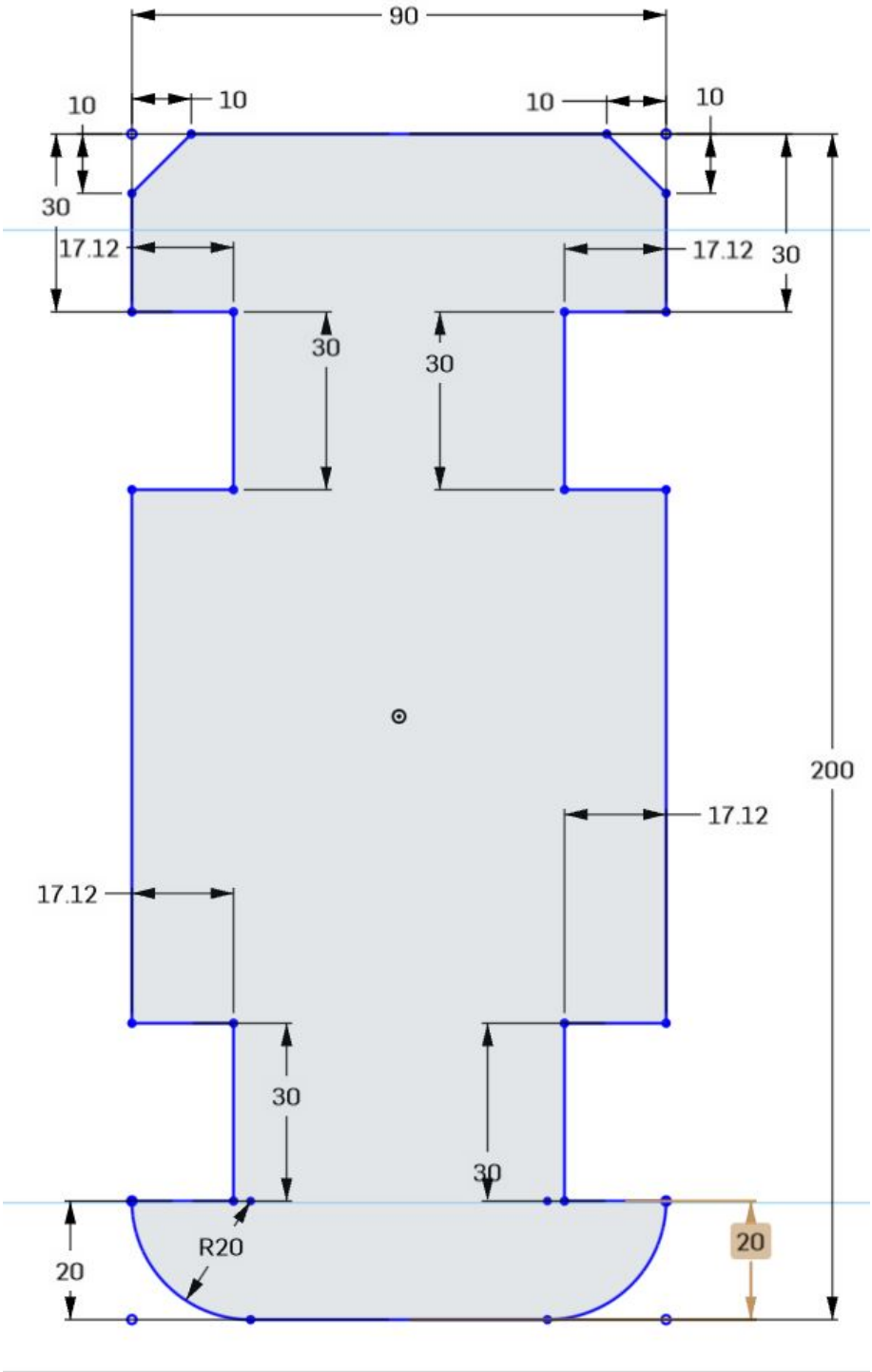
- Tutoriel animateur TinkerCAD - création d'une activité
- Tutoriel élève TinkerCAD - modélisation d'une pièce
- Tutoriel animateur OnShape - création d'une activité
- Tutoriel élève OnShape - modélisation d'une pièce

Déroulement de la séance

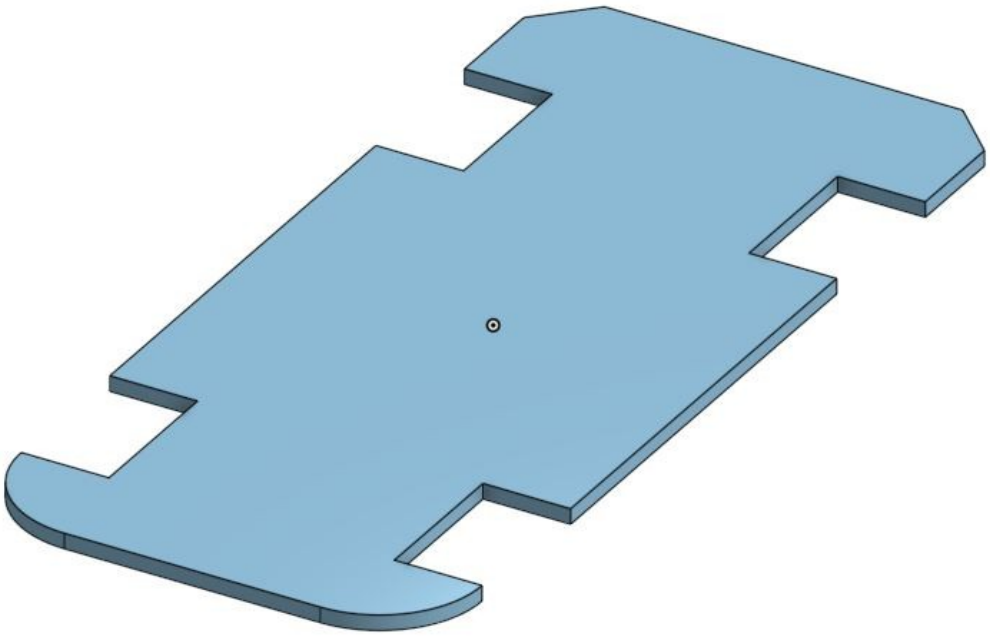
- Consignes : Sécurité, précautions matériel :
  - Travail individuel
  - Travail collaboratif
- Phases et méthodes d’animation
  1. Présentation du cycle 3 (10min)
  2. Présentation du cahier des charges de la voiture RC (10min)
  3. Présentation du fonctionnement d'une direction de voiture RC avec un servomoteur. (10min)
  4. Création de la voiture RC sur papier.
     1. Châssis
        1. Création d'un modèle de châssis pouvant accueillir deux moteurs CC en vue du dessus avec un système d'accroche des moteurs (imaginé par l'élèves)
        2. Création de la vue de côté pour permettre une meilleure visualisation du système d'accroche des moteurs.
        3. Création du système de direction avec un servomoteur
        4. Création du système d'accroche du châssis à la carrosserie.
     2. Châssis
        1. Création de la carrosserie
        2. Création du système d'accroche de la carrosserie au châssis
  5. Tutoriel TinkerCAD / OnShape (15min)
  6. Création des différentes parties en CAO

1. Création du châssis en CAO avec tout les systèmes.
2. Création de la carrosserie avec tout les systèmes.
3. Assemblage des deux parties.

Nous allons donc créer notre châssis, pour commencer voici la forme ainsi que les cotations de notre pièce.



Puis une extrusion sur 3mm.



Conclusion / Rangement / Démontage :

- Rangement en fin de séance
  - Débrancher et ranger les composants et les fournitures
  - Chaque participant vérifie la boite du voisin (check-list)
  - Remettre l'ordinateur dans l'état initial
- Programme de la prochaine séance

Phase 2 - Électronique et programmation

Prérequis participant :

- Travail sur ordinateur
- Manipulation avec la souris (clic droit, clic gauche, sélection,...)
- Manipulation de composant électronique (composant, câble...)

Compétences Animateur :

Compétences techniques et soft skills :

- Pratique du logiciel Arduino IDE
- Comprendre et savoir un langage de programmation
- Animation

Matériels nécessaires

logiciel, électronique, mécanique, outils, matériaux, code...

- 12 PCs (1 par participant)
  - Connexion et navigateur internet (Firefox)
  - Session invité ou nominative
- 12 moteurs à courant continue
- 12 module de contrôle L9110S
- 12 module Bluetooth HC-05
- Câble Dupont

Préparation :

Matériel par participant sur un poste PC en début de séance :

- PC allumé
- Codes d'accès session Windows/Linux
- Temps de préparation : 5min

Documentation / Tutoriels :

- Tutoriel Module Bluetooth + Moteur CC

Déroulement de la séance

- Consignes : Sécurité, précautions matériel :
  - Travail individuel
  - Travail collaboratif
- Phases et méthodes d’animation
  1. Présentation du cycle 3 (10min)
  2. Présentation du cahier des charges de la voiture RC (10min)
  3. Présentation du fonctionnement d'un moteur CC. (10min)
  4. Tutoriel Module Bluetooth + Moteur CC
  5. Câblage des composant
  6. Programmation des modules

Conclusion / Rangement / Démontage :

- Rangement en fin de séance
  - Débrancher et ranger les composants
  - Chaque participant vérifie la boite du voisin (check-list)
  - Remettre ordinateur dans l'état initial
- Programme de la prochaine séance

Moteur CC - Principe de fonctionnement

Explication du moteur à courant continue :

https://www.youtube.com/watch?v=JV50zqHvqAM&list=PLKLRexlzHZepqmJXVM3Kq52xCRRqlOZ0H

Câblage et programmation avec Module L9110S :

https://www.robot-maker.com/shop/blog/32_Utilisation-des-encodeurs.html

Câblage et programmation sans Module L9110D :

https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-SJ01_SKU__FIT0450

Moteur CC - Commande de moteur à courant continu

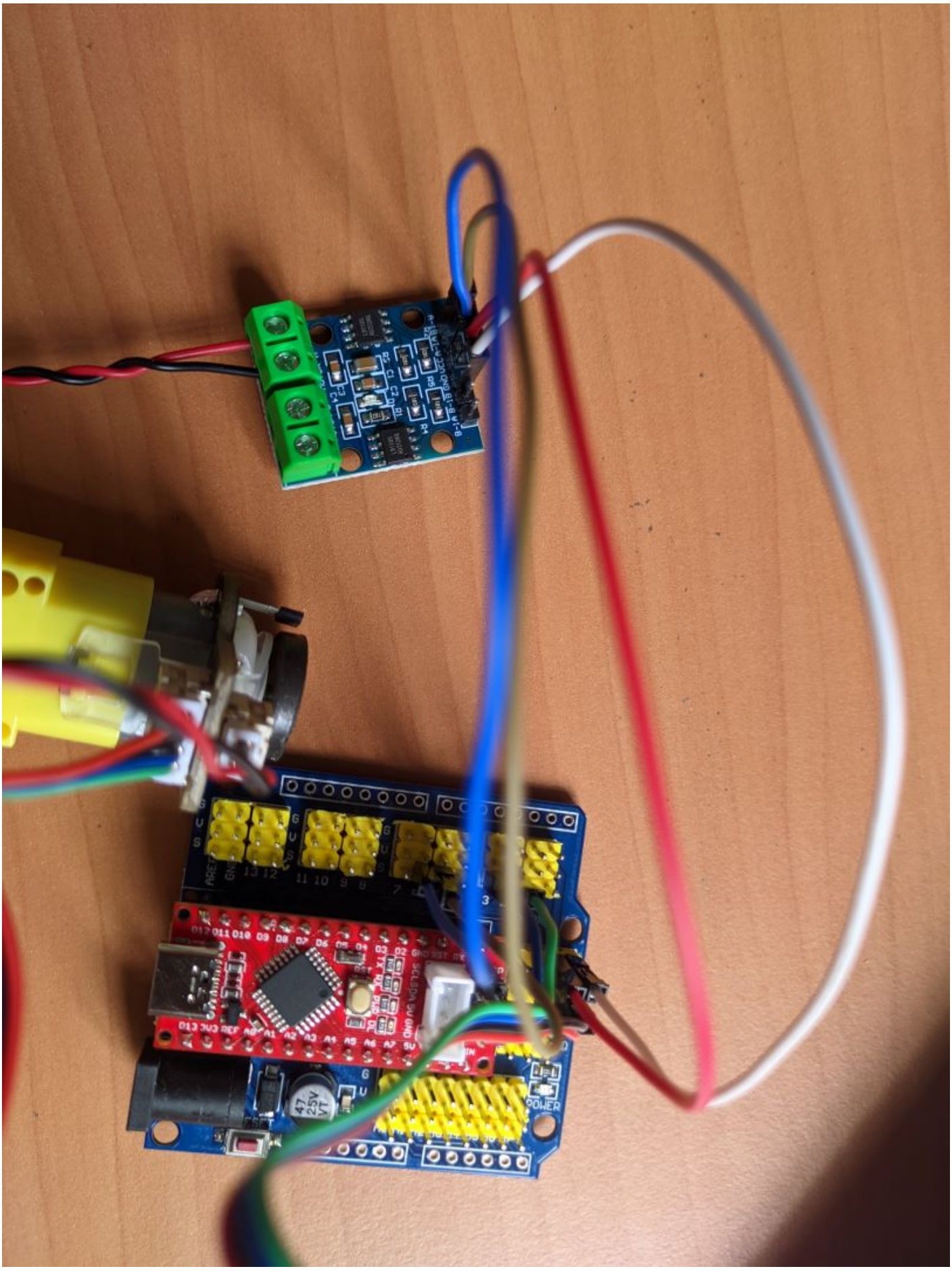
Moteur à courant continu "moteur jaune"

https://seafile.unistra.fr/d/16f415226d224af5bd8f/

\Seafile\Reseau_FabLab_Alsace_Nord\01_Robotique_Educative

Module de contrôle L9110S

Montage :



https://arduino.blaisepascal.fr/controleur-l9110s/

```
#define moteurA_1 5
#define moteurA_2 6
#define moteurB_1 10
#define moteurB_2 11
int vitesseA = 255;  // 0 à 255
int vitesseB = 255;  // 0 à 255
void setup() {
    // Configuration des ports en mode "sortie"
    pinMode(moteurA_1, OUTPUT);
    pinMode(moteurA_2, OUTPUT);
    pinMode(moteurB_1, OUTPUT);
    pinMode(moteurB_2, OUTPUT);
}
void loop() {
  digitalWrite(moteurA_1, LOW);
  analogWrite(moteurA_2, vitesseA);
  digitalWrite(moteurB_1, LOW);
  analogWrite(moteurB_2, vitesseB);
  delay(2000);

  analogWrite(moteurA_1, vitesseA);
  digitalWrite(moteurA_2, LOW);
  analogWrite(moteurB_1, vitesseB);
  digitalWrite(moteurB_2, LOW);

  delay(2000);
  digitalWrite(moteurA_1, LOW);
  digitalWrite(moteurA_2, LOW);
  digitalWrite(moteurB_1, LOW);
  digitalWrite(moteurB_2, LOW);

  delay(3000);
}
```

Module de contrôle L298

Attention le module entraîne une chute de tension d'environ 3V. Donc si vous voulez délivrer 5V à vos moteurs à courant continu il faudra une alimentation de 8V.

https://arduino.blaisepascal.fr/pont-en-h-l298n/

```
// Pont en H L298N

//Ports de commande du moteur B
int motorPin1 = 10;
int motorPin2 = 11;
int enablePin = 5;

// Vitesse du moteur
int state = 55;

void setup() {
    // Configuration des ports en mode "sortie"
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
    pinMode(enablePin, OUTPUT);

    // Initialisation du port série
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0)
    {
      // Lecture de l'entier passé au port série
      state = Serial.parseInt();

      //
      // Sens du mouvement
      //
      if (state > 0) // avant
      {
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        Serial.print("Avant ");
        Serial.println(state);
      }
      else if (state < 0) // arrière
```

```
      {
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        Serial.print("Arriere ");
        Serial.println(state);
      }
      else // Stop (freinage)
      {
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, HIGH);
        Serial.println("Stop");
      }

      //
      // Vitesse du mouvement
      //
      analogWrite(enablePin, abs(state));
    }
    delay(100);
}
```

PWM

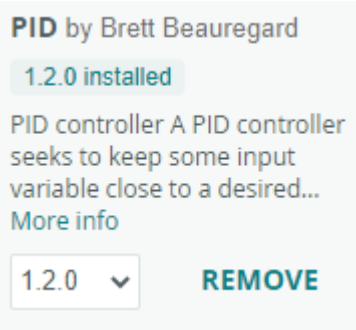
https://arduino.blaisepascal.fr/conversion-numeriqueanalogique-pwm/

Moteur CC - Contrôler la vitesse avec une roue encodeuse

Utilisation des encodeurs

https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-SJ01_SKU__FIT0450

- Installer Arduino PID depuis le gestionnaire de librairies




```
//source Sample Code 2 https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-
SJ01_SKU__FIT0450#target_3
//The sample code for driving one way motor encoder
#include <PID_v1.h>
const byte encoder0pinA = 2;//A pin -> the interrupt pin 0
const byte encoder0pinB = 3;//B pin -> the digital pin 3
// int E_left =5; //The enabling of L298PDC motor driver board connection to the digital
interface port 5
// int M_left =4; //The enabling of L298PDC motor driver board connection to the digital
interface port 4
int MOTEUR_A_1 =5; //Connexion du pilote de moteur CC l9110s au port digital 5
int MOTEUR_A_2 =6; //Connexion du pilote de moteur CC l9110s au port digital 6
byte encoder0PinALast;
double duration,abs_duration;//the number of the pulses
boolean Direction;//the rotation direction
```

```
boolean result;

double val_output;//Power supplied to the motor PWM value.
double Setpoint;
double Kp=0.6, Ki=5, Kd=0;
PID myPID(&abs_duration, &val_output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup()
{
  Serial.begin(9600);//Initialize the serial port
   pinMode(MOTEUR_A_1, OUTPUT);   //L298P Control port settings DC motor driver board for the
output mode
   pinMode(MOTEUR_A_2, OUTPUT);
   Setpoint =80;  //Set the output value of the PID
   myPID.SetMode(AUTOMATIC);//PID is set to automatic mode
   myPID.SetSampleTime(100);//Set PID sampling frequency is 100ms
  EncoderInit();//Initialize the module
}

void loop()
{
      advance();//Motor Forward
      abs_duration=abs(duration);
      result=myPID.Compute();//PID conversion is complete and returns 1
      if(result)
      {
        Serial.print("Pluse: ");
        Serial.println(duration);
        duration = 0; //Count clear, wait for the next count
      }



}

void EncoderInit()
{
  Direction = true;//default -> Forward
  pinMode(encoder0pinB,INPUT);
  attachInterrupt(0, wheelSpeed, CHANGE);
}
```

```
void wheelSpeed()
{
  int Lstate = digitalRead(encoder0pinA);
  if((encoder0PinALast == LOW) && Lstate==HIGH)
  {
    int val = digitalRead(encoder0pinB);
    if(val == LOW && Direction)
    {
      Direction = false; //Reverse
    }
    else if(val == HIGH && !Direction)
    {
      Direction = true;  //Forward
    }
  }
  encoder0PinALast = Lstate;

  if(!Direction)  duration++;
  else  duration--;

}
void advance()//Motor Forward
{
     digitalWrite(MOTEUR_A_1,LOW);
     analogWrite(MOTEUR_A_2,val_output);
}
void back()//Motor reverse
{
     digitalWrite(MOTEUR_A_1,HIGH);
     analogWrite(MOTEUR_A_2,val_output);
}

void Stop()//Motor stops
{
     digitalWrite(MOTEUR_A_2, LOW);
}
```

Avec la librairie Motor PID

https://github.com/natnqweb/Motor_PID

Interruptions

https://www.best-microcontroller-projects.com/arduino-interrupt.html

https://forum.arduino.cc/t/solved-arduino-nano-and-interrupt-on-pin-6/1090623/3

Sources

https://www.robot-maker.com/shop/blog/32_Utilisation-des-encodeurs.html

Le module Bluetooth HC-05

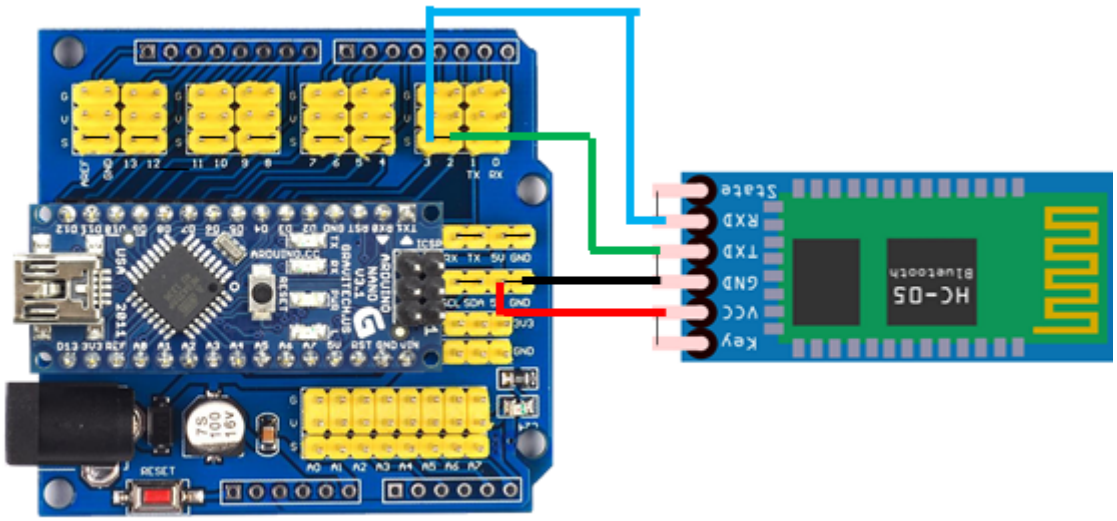
Test et configuration du module

- Suivre les instructions pour le module HC-05, qui peut être configuré en module maître ou esclave. Le module HC-06 lui ne peut être que esclave, mais il est plus simple à configurer.

https://www.aranacorp.com/fr/votre-arduino-communique-avec-le-module-hc-05/

- On peut communiquer avec le *module bluetooth HC-05* via la liaison série de l'*Arduino Nano*
- Se renseigner sur ce qu'est une liaison série, par ex. ici :

  https://arduino.blaisepascal.fr/bibliotheque-serial/
- Réaliser le câblage entre l'Arduino Nano et le module bluetooth HC-05



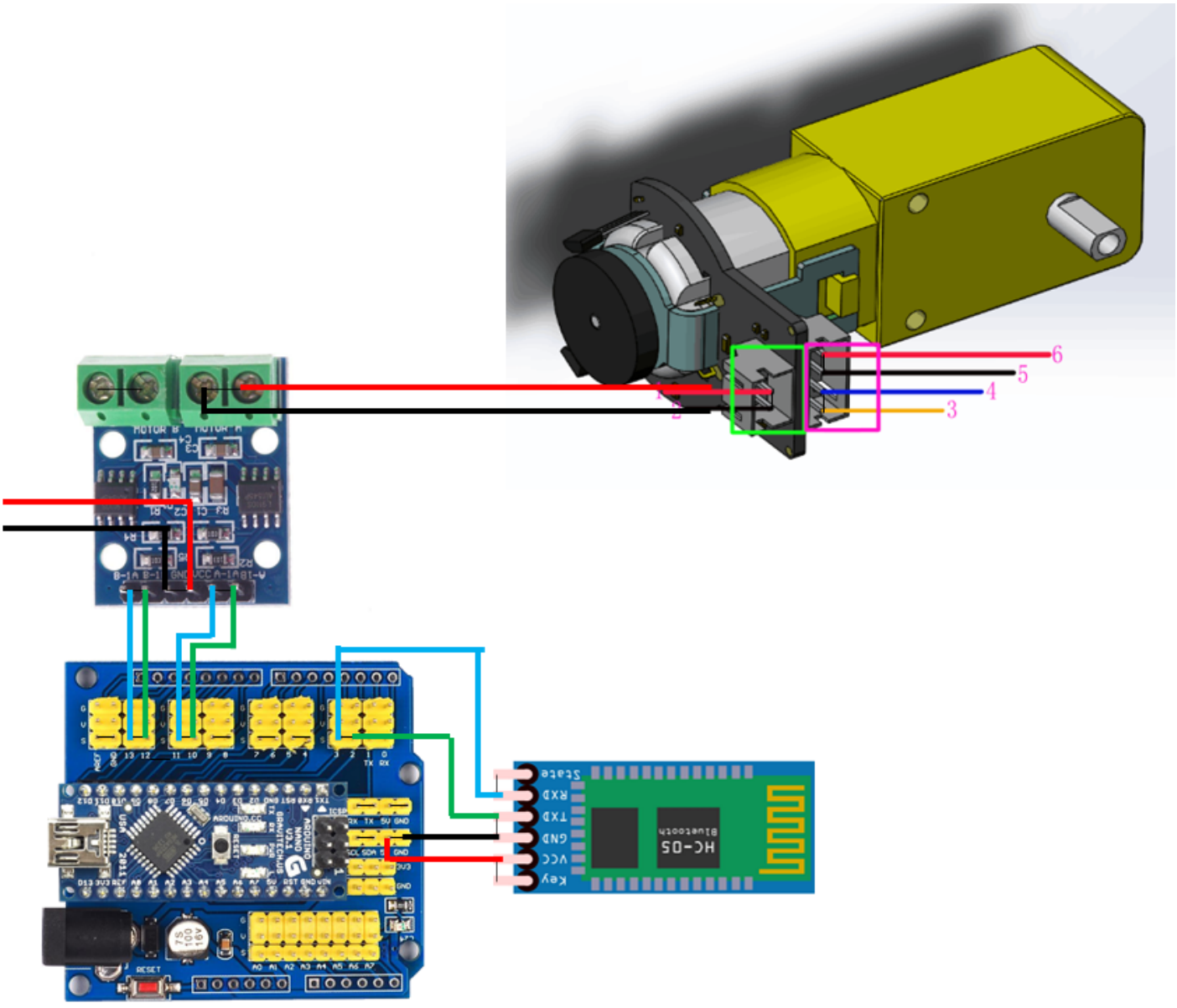
- Bien vérifier que le Pin qui **R**eçoit les données sur l'Arduino (**R**xDpin) soit connecté au Pin du HC-05 qui **T**ansmet (TxDpin). Et vice-versa, donc chaque câble transmet les données dans une direction opposée.
- Si on prend les Pins RX (Digital 0) et TX (Digital 1) de l'arduino pour connecter RXD et TXD du HC-05, cela rentre en conflit avec la connexion USB entre l'arduino et le PC.
- Donc on prend les Pins Digital 2 et 3 et on utilise la bibliothèque `SoftwareSerial`

```
//Test et Configuration du module Bluetooth HC-05

//Le port série matériel de l'Arduino Nano (Pins 0/RX et 1/TX) est déjà utilisé pour la liaison Arduino-USB avec l'ordinateur
//On utilise donc une liaison série logicielle pour la liaison Arduino-HC05 avec le module Bluetooth
#include <SoftwareSerial.h>  //Software Serial Port
#define RxDpin 2    //Pin Digital 2 pour arduino Rx (pin0=serial)
#define TxDpin 3    //Pin Digital 3 pour arduino Tx (pin1)
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut configurer le HC-05
//on maintient le bouton à côté de la PIN EN/KEY enfoncé au démarrage de l'Arduino
#define baudrate 38400 //Vitesse pour la liaion Arduino-HC05 en mode configuration
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut communiquer à travers le bluetooth depuis un smartphone,
//on appaire le HC-05 depuis l'appli avec le mot-de-passe par défaut : 1234
// #define baudrate 9600 //Vitesse pour la liaion Arduino-HC05 en mode utilisation smartphone
SoftwareSerial BTSerie(RxDpin,TxDpin);

char caractereTexte;
String phraseTexte;

void setup()

{
Serial.begin(9600); //Vitesse (baudRate) pour la liaison Arduino-USB
delay(500);
Serial.println("En mode communication USB - Pret pour les commandes AT");
Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");

//Configuration de la liaison SoftwareSerial avec le HC-05
pinMode(RxDpin, INPUT);   //Configuration du Pin RxD (Receive) en mode entrée
pinMode(TxDpin, OUTPUT);  //Configuration du Pin TxD (Transmit) en mode sortie
BTSerie.begin(baudrate);  //Vitesse pour la liaion Arduino-HC05
if (baudrate==38400) {
  Serial.println("En mode communication USB - Pret pour les commandes AT");
  Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");
}
else if (baudrate==9600){
  Serial.println("En mode smartphone - Pret pour être appairé");
```

```
  Serial.println("Le HC-05 doit clignoter rapidement avant d'être appairé");
}
else{
  Serial.println("La vitesse de communication (baudrate) a été personnalisée");
}

delay(500);
// Commandes AT pour le HC-05
// BTSerie.print("AT+NAME?");  //Demande le nom du module. Noter le ?
// BTSerie.print("AT+NAME=RCcar-HC-05-1");  //Définir le nom du module.
BTSerie.print("AT+VERSION?");  //Demande le N° de version. Noter le ?
// BTSerie.print("AT+UART?");  //Demande la vitesse série (baudrate). Noter le ?
// BTSerie.print("AT+UART=57600,0,0");  //Définir la vitesse série (baudrate).
// BTSerie.print("AT+ROLE?");  //Demande le mode du module, maitre ou esclave. Noter le ?
// BTSerie.print("AT+PSWD?");  //Demande le mot-de-passe du module. Noter le ?
// La console série de l'ordinateur d'où l'on envoie les commandes AT doit être réglée de
telle sorte que
// les fins de ligne soient « les deux, NL et CR », ce qui revient à envoyer \r\n à la fin de
chaque commande.
BTSerie.print("\r\n");     // sur HC-05, toutes les commandes doivent se terminer par \r\n
// afficher ce que le module bluetooth répond
  Serial.print( BTSerie.read() );     // afficher sur la console ce qui est lu sur BT
// pour AT+VERSION?, c'est le n° de version puis OK qui s'affiche

void loop(){
  //On lit ce qui est envoyé à l'Arduino depuis la console via la liaison Serial
  readSerialPort();
  //Et on l'Arduino l'envoie au HC-05 via la liaison SoftwareSerial
  if(phraseTexte!="") BTSerie.println(phraseTexte);
  //L'Arduino lit ce que le HC-05 envoie via la liaison SoftwareSerial et l'envoie vers la
console
  if (BTSerie.available()>0){
    Serial.write(BTSerie.read());
  }
}
void readSerialPort(){
  phraseTexte="";
  while (Serial.available()) {
    delay(10);
    if (Serial.available() >0) {
```

```
      caractereTexte = Serial.read(); //le port série envoie des caractères de texte octet par
octet (byte from serial buffer)
      phraseTexte += caractereTexte; //construction d'une phrase en concaténant les caractères
reçus
    }
 }
}
```

Pilotage d'un moteur CC via bluetooth



```
//Voiture modélisée radiocommandée (RC car) avec deux moteurs CC à l'arrière et un servomoteur de direction

#include <Servo.h>
Servo myservo;  // create servo object to control a servo

//Le port série matériel de l'Arduino Nano (Pins 0/RX et 1/TX) est déjà utilisé pour la liaison Arduino-USB avec l'ordinateur
//On utilise donc une liaison série logicielle pour la liaison Arduino-HC05 avec le module Bluetooth
#include <SoftwareSerial.h>  //Software Serial Port
#define RxDpin 2    //Pin Digital 2 pour arduino Rx (pin0=serial)
#define TxDpin 3    //Pin Digital 3 pour arduino Tx (pin1)
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut configurer le HC-05
//on maintient le bouton à côté de la PIN EN/KEY enfoncé au démarrage de l'Arduino
// #define baudrate 38400 //Vitesse pour la liaion Arduino-HC05 en mode configuration
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut communiquer à travers le bluetooth depuis un smartphone,
//on appaire le HC-05 depuis l'appli avec le mot-de-passe par défaut : 1234
#define baudrate 9600 //Vitesse pour la liaion Arduino-HC05 en mode utilisation smartphone
SoftwareSerial BTSerie(RxDpin,TxDpin);

char caractereTexte;
String phraseTexte;

void setup()

{
Serial.begin(9600); //Vitesse (baudRate) pour la liaison Arduino-USB
delay(500);

//Configuration de la liaison SoftwareSerial avec le HC-05
pinMode(RxDpin, INPUT);   //Configuration du Pin RxD (Receive) en mode entrée
pinMode(TxDpin, OUTPUT);  //Configuration du Pin TxD (Transmit) en mode sortie
BTSerie.begin(baudrate);  //Vitesse pour la liaion Arduino-HC05
if (baudrate==38400) {
  Serial.println("En mode communication USB - Pret pour les commandes AT");
  Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");
}
else if (baudrate==9600){
```

```
  Serial.println("En mode smartphone - Pret pour être appairé");
  Serial.println("Le HC-05 doit clignoter rapidement avant d'être appairé");
}
else{
  Serial.println("La vitesse de communication (baudrate) a été personnalisée");
}

delay(500);

pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse
pinMode(9,OUTPUT);   //Led
pinMode(5,OUTPUT);   //SG90 steering motor
myservo.attach(5);  // attaches the servo on pin 5 to the servo object
}

void loop() {
  // Serial.write(blueToothSerial.read());

  //On lit caractere par caractere sur la liaion Arduino-USB et on affiche sur la liaison
Arduino-HC05
  if (Serial.available()) {
    caractereTexte = Serial.read();
    BTSerie.write(caractereTexte);
    // Serial.println("Caractere envoye vers bluetooth : ");
    // Serial.println(caractereRecu);
  }
  //On lit caractere par caractere sur la liaion Arduino-HC05 et on affiche sur la liaison
Arduino-USB
  if (BTSerie.available()) {
    caractereTexte = BTSerie.read();
    Serial.print(caractereTexte);
  // }

if(caractereTexte == 'F'){            //move forward(all motors rotate in forward direction)
  digitalWrite(10,LOW);
  digitalWrite(12,LOW);
  digitalWrite(13,HIGH);
```

```
  digitalWrite(11,HIGH);
  // myservo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'B'){      //move reverse (all motors rotate in reverse direction)
  digitalWrite(13,LOW);
  digitalWrite(11,LOW);
  digitalWrite(12,HIGH);
  digitalWrite(10,HIGH);
  // myservo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'L'){      //turn right (left side motors rotate in forward
direction, right side motors doesn'caractereRecu rotate)
  digitalWrite(10,LOW);
  digitalWrite(12,LOW);
  digitalWrite(13,LOW);
  digitalWrite(11,HIGH);
  // myservo.write(60);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'R'){      //turn left (right side motors rotate in forward
direction, left side motors doesn'caractereRecu rotate)
  digitalWrite(10,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(13,HIGH);
  // myservo.write(120);                  // sets the servo position according to the scaled
value
}


else if(caractereTexte == 'S'){      //STOP (all motors stop)
  digitalWrite(13,LOW);
  digitalWrite(12,LOW);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW);
```

```
  // myservo.write(90);
}
delay(100);
  }
}
```

Depuis le clavier de PC

voir : https://innovation.iha.unistra.fr/books/robotique-educative/page/rc-car-voiture-modelisme-radiocommandee#bkmrk-depuis-le-moniteur-s

Depuis une application bluetooth de smartphone Android

- Ajouter le HC-05 depuis les paramètres Bluetooth Android
- Rentrer le code PIN `1234`
- Démarrer l'Application Bluetooth
- Appuyer sur les flèches / le joystick
- Si les flèches ne fonctionnent pas c'est qu'elles n'envoient pas les bons caractères
- Vous pouvez envoyer directement les caractères via un terminal série d'une Application Bluetooth

Voici les caractères Android :

- F pour Forward : Avancer
- B pour Backward : Reculer
- R pour Right : Aller à droite
- L pour Left : Aller à gauche
- S pour Stop : Arrêter

Voici le fonctionnement du joystick de l'application E&E: Arduino Automation par Engineers & Electronics

- Quand on appuie sur la flèche du haut, le caractère `F` (objet `char`) est envoyé par le téléphone au HC-05 puis à l'Arduino via la liaison série Software Arduino-HC05 qui est configurée par défaut en baudrate 9600 (vérifiable en envoyant la commande `BTSerie.print("AT+UART?");` au HC-05)
- Quand on relâche le bouton, le caractère `S` est envoyé

L'application E&E: Arduino Automation par Engineers & Electronics est testée ne semble pas fonctionner avec les Android récents ni iOS :

- https://play.google.com/store/apps/details?id=com.himanshu.ArduinoAutomation
- Chercher une application alternative en tapant les mots-clé bluetooth arduino

- Autres applications possibles : Carino, bluetooth2, BLE RobotCar, Arduino Bluetooth Controller (giumig)

Sources

https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/

RC Car - Voiture modélisme radiocommandée

Pilotage depuis le clavier d'un PC

via la liaison série Arduino-USB

- Sans contrôle de vitesse des moteurs CC

```
//Voiture modélisée radiocommandée (RC car) avec deux moteurs CC à l'arrière et un servomoteur de direction

  // A COMPLETER pour le servo //

//Le port série matériel de l'Arduino Nano (Pins 0/RX et 1/TX) est déjà utilisé pour la liaison Arduino-USB avec l'ordinateur
//On utilise donc une liaison série logicielle pour la liaison Arduino-HC05 avec le module Bluetooth
#include <SoftwareSerial.h>  //Software Serial Port
#define RxDpin 2    //Pin Digital 2 pour arduino Rx (pin0=serial)
#define TxDpin 3    //Pin Digital 3 pour arduino Tx (pin1)
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut configurer le HC-05
//on maintient le bouton à côté de la PIN EN/KEY enfoncé au démarrage de l'Arduino
// #define baudrate 38400 //Vitesse pour la liaion Arduino-HC05 en mode configuration
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut communiquer à travers le bluetooth depuis un smartphone,
//on appaire le HC-05 depuis l'appli avec le mot-de-passe par défaut : 1234
#define baudrate 9600 //Vitesse pour la liaion Arduino-HC05 en mode utilisation smartphone
SoftwareSerial BTSerie(RxDpin,TxDpin);

char caractereTexte;
String phraseTexte;

void setup()
```

```
{
Serial.begin(9600); //Vitesse (baudRate) pour la liaison Arduino-USB
delay(500);

//Configuration de la liaison SoftwareSerial avec le HC-05
pinMode(RxDpin, INPUT);   //Configuration du Pin RxD (Receive) en mode entrée
pinMode(TxDpin, OUTPUT);  //Configuration du Pin TxD (Transmit) en mode sortie
BTSerie.begin(baudrate);  //Vitesse pour la liaion Arduino-HC05
if (baudrate==38400) {
  Serial.println("En mode communication USB - Pret pour les commandes AT");
  Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");
}
else if (baudrate==9600){
  Serial.println("En mode smartphone - Pret pour être appairé");
  Serial.println("Le HC-05 doit clignoter rapidement avant d'être appairé");
}
else{
  Serial.println("La vitesse de communication (baudrate) a été personnalisée");
}

delay(500);

pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse
pinMode(9,OUTPUT);   //Led
pinMode(5,OUTPUT);   //SG90 steering motor
  // A COMPLETER pour le servo //
//myservo.attach(5);  // attaches the servo on pin 5 to the servo object
}

void loop() {
  // Serial.write(blueToothSerial.read());

  //On lit caractere par caractere sur la liaion Arduino-HC05 et on affiche sur la liaison
Arduino-USB
  if (BTSerie.available()) {
    caractereTexte = BTSerie.read();
```

```
    Serial.print(caractereTexte);
  }
  //On lit caractere par caractere sur la liaion Arduino-USB et on affiche sur la liaison Arduino-HC05
  if (Serial.available()) {
    caractereTexte = Serial.read();
    BTSerie.write(caractereTexte);
    // Serial.println("Caractere envoye vers bluetooth : ");
    // Serial.println(caractereRecu);
  // }

if(caractereTexte == 'F'){            //F comme Forward - avancer (tous les moteurs vers l'avant, servo à 90°)
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
  digitalWrite(12,LOW);
  digitalWrite(13,HIGH);
}

else if(caractereTexte == 'B'){      //B comme Backward reculer (tous les moteurs en sens inverse, servo à 90°)
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
  // A COMPLETER //
}

else if(caractereTexte == 'L'){      //L comme Left - tourner à gauche (moteur de droite vers l'avant, gauche à l'arrêt, servo à 120°)
  // A COMPLETER //
}

else if(caractereTexte == 'R'){      //R comme Right - tourner à droite (moteur de gauche vers l'avant, droite à l'arrêt, servo à 60°)
  // A COMPLETER //
}



else if(caractereTexte == 'S'){      //STOP (tous les moteurs à l'arrêt, servo à 90°)
  // A COMPLETER //
}
```

```
delay(100);
  }
}
```

Pilotage bluetooth avec contrôle de la vitesse des moteur

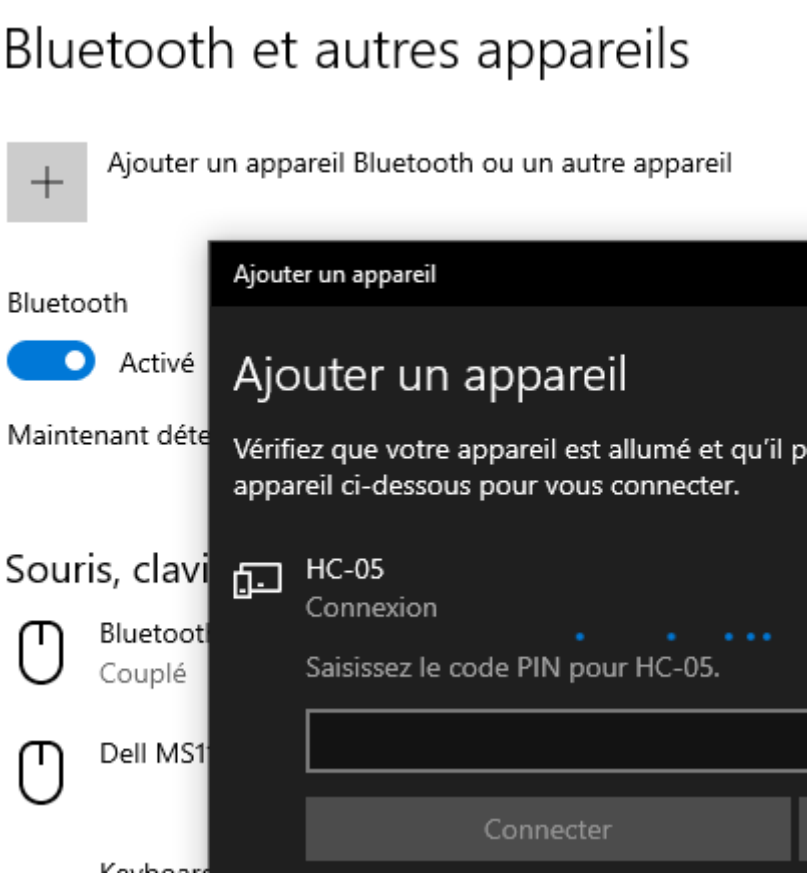
Depuis une application bluetooth de smartphone Android

voir : https://innovation.iha.unistra.fr/books/robotique-educative/page/le-module-bluetooth-hc-05#bkmrk-application-bluetoot

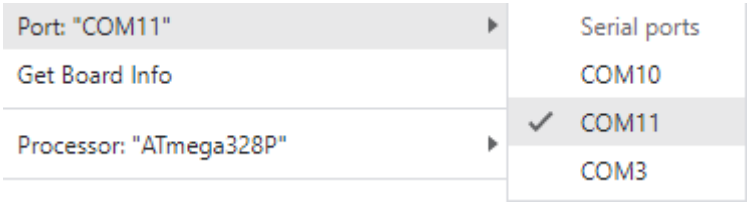
Depuis le moniteur série d'Arduino IDE

Source : https://www.instructables.com/View-Serial-Monitor-Over-Bluetooth/#

- Ajouter le HC-05 dans Windows depuis les périphériques bluetooth et saisir le code PIN `1234`

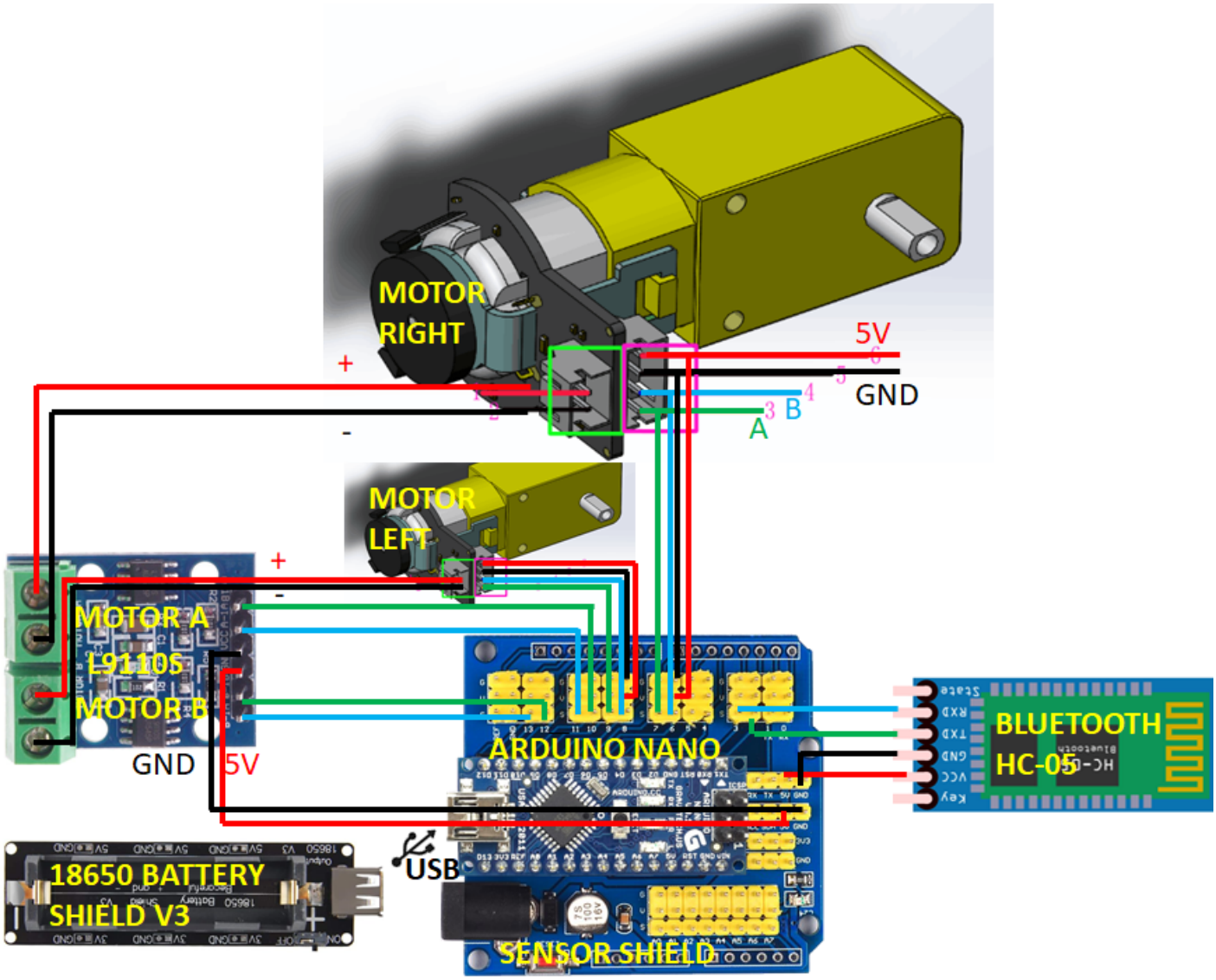


- Dans Arduino IDE
- Sélectionner le port série correspondant au HC-05. Il y en a deux, tester les deux



- Ouvrir le moniteur série

- Envoyer les commandes de pilotage : `F`, `B`, `L`, `R`, `S`.




```
//source Sample Code 2 https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-
SJ01_SKU__FIT0450#target_3
//The sample code for driving one way motor encoder
#include <PID_v1.h>
```

```
const byte encoder0pinA = 0;//A pin -> the interrupt pin 0
const byte encoder0pinB = 1;//B pin -> the digital pin 3
const byte encoder0pinA_2 = 8;//A pin -> the interrupt pin 0
const byte encoder0pinB_2 = 9;//B pin -> the digital pin 3
// int E_left =5; //The enabling of L298PDC motor driver board connection to the digital
interface port 5
// int M_left =4; //The enabling of L298PDC motor driver board connection to the digital
interface port 4
int MOTEUR_A_1 =12; //Connexion du pilote de moteur CC l9110s au port digital 5
int MOTEUR_A_2 =13; //Connexion du pilote de moteur CC l9110s au port digital 6
int MOTEUR_B_1 =10; //Connexion du pilote de moteur CC l9110s au port digital 5
int MOTEUR_B_2 =11; //Connexion du pilote de moteur CC l9110s au port digital 6
byte encoder0PinALast;
double duration,abs_duration;//the number of the pulses
boolean Direction;//the rotation direction
boolean result;

double val_output;//Power supplied to the motor PWM value.
double Setpoint;
double Kp=0.6, Ki=5, Kd=0;
PID myPID(&abs_duration, &val_output, &Setpoint, Kp, Ki, Kd, DIRECT);

#include <Servo.h>
#define trigPin 6
#define echoPin 7
// #define EncoderInit
Servo servo1;
Servo servo2;
Servo monServo;

//Le port série matériel de l'Arduino Nano (Pins 0/RX et 1/TX) est déjà utilisé pour la
liaison Arduino-USB avec l'ordinateur
//On utilise donc une liaison série logicielle pour la liaison Arduino-HC05 avec le module
Bluetooth
#include <SoftwareSerial.h>  //Software Serial Port
#define RxDpin 2    //Pin Digital 2 pour arduino Rx (pin0=serial)
#define TxDpin 3    //Pin Digital 3 pour arduino Tx (pin1)
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut configurer le HC-05
//on maintient le bouton à côté de la PIN EN/KEY enfoncé au démarrage de l'Arduino
// #define baudrate 38400 //Vitesse pour la liaion Arduino-HC05 en mode configuration
```

```
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut communiquer à travers le
bluetooth depuis un smartphone,
//on appaire le HC-05 depuis l'appli avec le mot-de-passe par défaut : 1234
#define baudrate 9600
#include <Servo.h>
Servo myservo;  // create servo object to control a servo


SoftwareSerial BTSerie(RxDpin,TxDpin);

char caractereTexte;
String phraseTexte;

void setup()
{
  Serial.begin(9600);//Initialize the serial port
   pinMode(MOTEUR_A_1, OUTPUT);   //L298P Control port settings DC motor driver board for the
output mode
   pinMode(MOTEUR_A_2, OUTPUT);
   pinMode(MOTEUR_B_1, OUTPUT);   //L298P Control port settings DC motor driver board for the
output mode
   pinMode(MOTEUR_B_2, OUTPUT);
   Setpoint =80;  //Set the output value of the PID
   myPID.SetMode(AUTOMATIC);//PID is set to automatic mode
   myPID.SetSampleTime(100);//Set PID sampling frequency is 100ms
  EncoderInit();//Initialize the module

pinMode(RxDpin, INPUT);   //Configuration du Pin RxD (Receive) en mode entrée
pinMode(TxDpin, OUTPUT);  //Configuration du Pin TxD (Transmit) en mode sortie
BTSerie.begin(baudrate);
// Commandes AT pour le HC-05
// BTSerie.print("AT+NAME?");  //Demande le nom du module. Noter le ?
// BTSerie.print("AT+NAME=RCcar-HC-05-1");  //Définir le nom du module.
BTSerie.print("AT+VERSION?");  //Demande le N° de version. Noter le ?
// BTSerie.print("AT+UART?");  //Demande la vitesse série (baudrate). Noter le ?
// BTSerie.print("AT+UART=57600,0,0");  //Définir la vitesse série (baudrate).
// BTSerie.print("AT+ROLE?");  //Demande le mode du module, maitre ou esclave. Noter le ?
// BTSerie.print("AT+PSWD?");  //Demande le mot-de-passe du module. Noter le ?
// La console série de l'ordinateur d'où l'on envoie les commandes AT doit être réglée de
telle sorte que
```

```
// les fins de ligne soient « les deux, NL et CR », ce qui revient à envoyer \r\n à la fin de chaque commande.
BTSerie.print("\r\n");     // sur HC-05, toutes les commandes doivent se terminer par \r\n
// afficher ce que le module bluetooth répond
  Serial.print( BTSerie.read() );     // afficher sur la console ce qui est lu sur BT
// pour AT+VERSION?, c'est le n° de version puis OK qui s'affiche

if (baudrate==38400) {
  Serial.println("En mode communication USB - Pret pour les commandes AT");
  Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");
}
else if (baudrate==9600){
  Serial.println("En mode smartphone - Pret pour être appairé");
  Serial.println("Le HC-05 doit clignoter rapidement avant d'être appairé");
}
else{
  Serial.println("La vitesse de communication (baudrate) a été personnalisée");
}
monServo.attach(5);
delay(500);

pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse
pinMode(9,OUTPUT);   //Led
pinMode(5,OUTPUT);   //SG90 steering motor
  // A COMPLETER pour le servo //
//myservo.attach(5);  // attaches the servo on pin 5 to the servo object
}


void loop()
{

//On lit caractere par caractere sur la liaion Arduino-USB et on affiche sur la liaison Arduino-HC05
if (Serial.available()) {
caractereTexte = Serial.read();
BTSerie.write(caractereTexte);
```

```
// Serial.println("Caractere envoye vers bluetooth : ");
// Serial.println(caractereRecu);
}

//On lit caractere par caractere sur la liaion Arduino-HC05 et on affiche sur la liaison
Arduino-USB
if (BTSerie.available()) {
caractereTexte = BTSerie.read();
Serial.print(caractereTexte);
// }

if(caractereTexte == 'F'){            //move forward(all motors rotate in forward direction)
  advance();//Motor Forward
  monServo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'B'){      //move reverse (all motors rotate in reverse direction)
  back();//Motor reverse
  monServo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'L'){      //turn right (left side motors rotate in forward
direction, right side motors doesn'caractereRecu rotate)
  left();
  monServo.write(60);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'R'){      //turn left (right side motors rotate in forward
direction, left side motors doesn'caractereRecu rotate)
  right();
  monServo.write(120);                  // sets the servo position according to the scaled
value
}


else if(caractereTexte == 'S'){      //STOP (all motors stop)
  Stop();
```

```
  monServo.write(90);
}
abs_duration=abs(duration);
      result=myPID.Compute();//PID conversion is complete and returns 1
      if(result)
      {
        Serial.print("Pluse: ");
        Serial.println(duration);
        duration = 0; //Count clear, wait for the next count
      }              // sets the servo position according to the scaled value
}
}
void EncoderInit()
{
  Direction = true;//default -> Forward
  pinMode(encoder0pinB,INPUT);
  attachInterrupt(0, wheelSpeed, CHANGE);
}

void wheelSpeed()
{
  int Lstate = digitalRead(encoder0pinA);
  if((encoder0PinALast == LOW) && Lstate==HIGH)
  {
    int val = digitalRead(encoder0pinB);
    if(val == LOW && Direction)
    {
      Direction = false; //Reverse
    }
    else if(val == HIGH && !Direction)
    {
      Direction = true;  //Forward
    }
  }
  encoder0PinALast = Lstate;

  if(!Direction)  duration++;
  else  duration--;

}
```

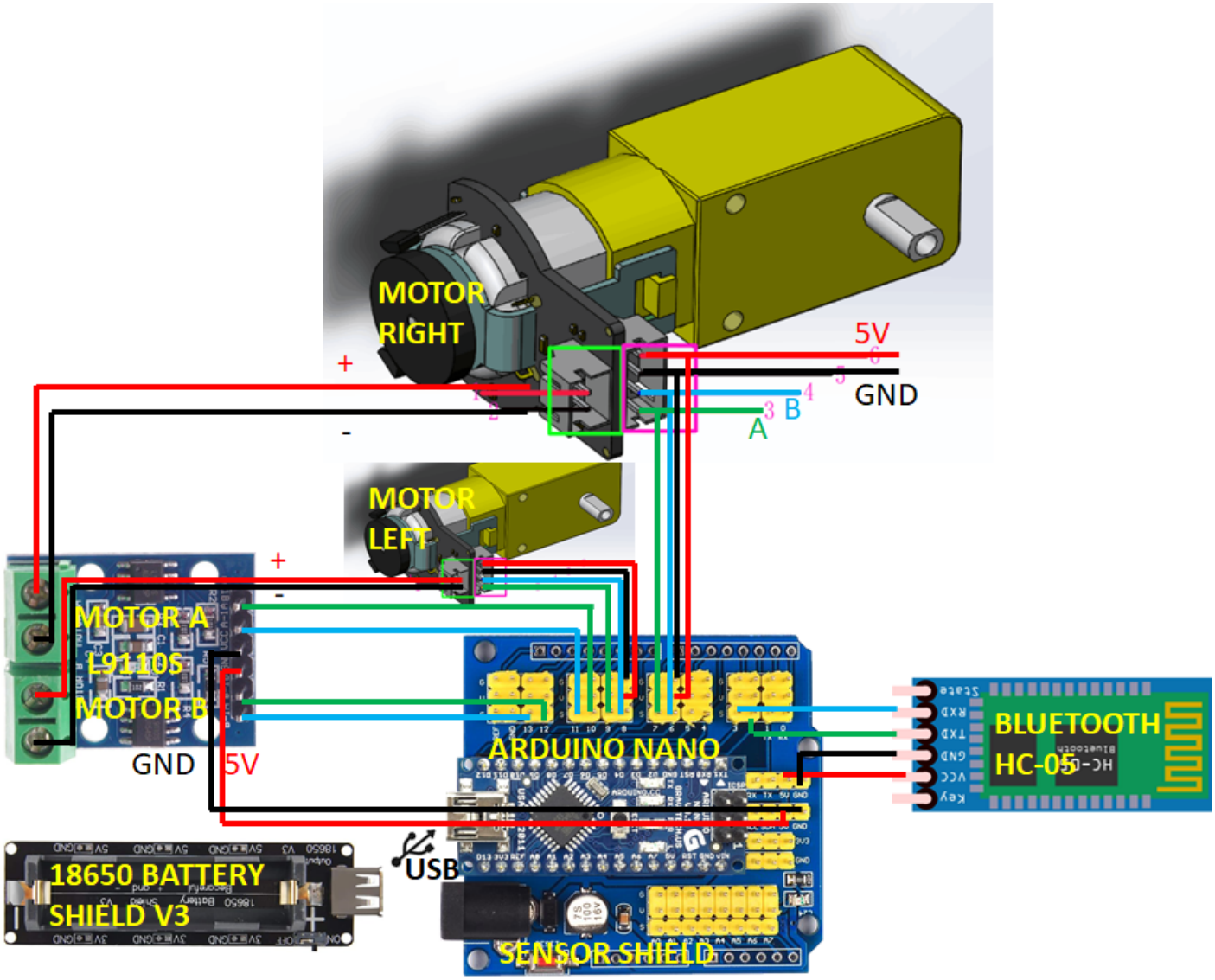
```
void advance()//Motor Forward
{
     digitalWrite(MOTEUR_A_1,LOW);
     digitalWrite(MOTEUR_A_2,val_output);
     digitalWrite(MOTEUR_B_1,val_output);
     digitalWrite(MOTEUR_B_2,LOW);
}
void back()//Motor reverse
{
     digitalWrite(MOTEUR_A_1,val_output);
     digitalWrite(MOTEUR_A_2,LOW);
     digitalWrite(MOTEUR_B_1,LOW);
     digitalWrite(MOTEUR_B_2,val_output);

}
void left()
{
     digitalWrite(MOTEUR_A_1,LOW);
     digitalWrite(MOTEUR_A_2,val_output);
     digitalWrite(MOTEUR_B_1,LOW);
     digitalWrite(MOTEUR_B_2,LOW);
}
void right()
{
     digitalWrite(MOTEUR_A_1,LOW);
     digitalWrite(MOTEUR_A_2,LOW);
     digitalWrite(MOTEUR_B_1,val_output);
     digitalWrite(MOTEUR_B_2,LOW);
}

void Stop()//Motor stops
{
     digitalWrite(MOTEUR_A_1, LOW);
     digitalWrite(MOTEUR_A_2, LOW);
     digitalWrite(MOTEUR_B_1, LOW);
     digitalWrite(MOTEUR_B_2, LOW);
}
```

Sources

https://github.com/himanshus2847/Bluetooth-Controlled-Robot-using-Arduino

https://www.youtube.com/watch?v=o-aRCxh9IhE

Code voiture RC




```
//source Sample Code 2 https://wiki.dfrobot.com/Micro_DC_Motor_with_Encoder-
SJ01_SKU__FIT0450#target_3
//The sample code for driving one way motor encoder
#include <PID_v1.h>
const byte encoder0pinA = 0;//A pin -> the interrupt pin 0
const byte encoder0pinB = 1;//B pin -> the digital pin 3
const byte encoder0pinA_2 = 8;//A pin -> the interrupt pin 0
const byte encoder0pinB_2 = 9;//B pin -> the digital pin 3
// int E_left =5;  //The enabling of L298PDC motor driver board connection to the digital
interface port 5
```

```
// int M_left =4; //The enabling of L298PDC motor driver board connection to the digital interface port 4
int MOTEUR_A_1 =12; //Connexion du pilote de moteur CC l9110s au port digital 5
int MOTEUR_A_2 =13; //Connexion du pilote de moteur CC l9110s au port digital 6
int MOTEUR_B_1 =10; //Connexion du pilote de moteur CC l9110s au port digital 5
int MOTEUR_B_2 =11; //Connexion du pilote de moteur CC l9110s au port digital 6
byte encoder0PinALast;
double duration,abs_duration;//the number of the pulses
boolean Direction;//the rotation direction
boolean result;

double val_output;//Power supplied to the motor PWM value.
double Setpoint;
double Kp=0.6, Ki=5, Kd=0;
PID myPID(&abs_duration, &val_output, &Setpoint, Kp, Ki, Kd, DIRECT);

#include <Servo.h>
#define trigPin 6
#define echoPin 7
// #define EncoderInit
Servo servo1;
Servo servo2;
Servo monServo;

//Le port série matériel de l'Arduino Nano (Pins 0/RX et 1/TX) est déjà utilisé pour la liaison Arduino-USB avec l'ordinateur
//On utilise donc une liaison série logicielle pour la liaison Arduino-HC05 avec le module Bluetooth
#include <SoftwareSerial.h>  //Software Serial Port
#define RxDpin 2    //Pin Digital 2 pour arduino Rx (pin0=serial)
#define TxDpin 3    //Pin Digital 3 pour arduino Tx (pin1)
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut configurer le HC-05
//on maintient le bouton à côté de la PIN EN/KEY enfoncé au démarrage de l'Arduino
// #define baudrate 38400 //Vitesse pour la liaion Arduino-HC05 en mode configuration
//Pour la liaison SoftwareSerial Arduino-HC05, quand on veut communiquer à travers le bluetooth depuis un smartphone,
//on appaire le HC-05 depuis l'appli avec le mot-de-passe par défaut : 1234
#define baudrate 9600
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
```

```
SoftwareSerial BTSerie(RxDpin,TxDpin);

char caractereTexte;
String phraseTexte;

void setup()
{
  Serial.begin(9600);//Initialize the serial port
   pinMode(MOTEUR_A_1, OUTPUT);   //L298P Control port settings DC motor driver board for the
output mode
   pinMode(MOTEUR_A_2, OUTPUT);
   pinMode(MOTEUR_B_1, OUTPUT);   //L298P Control port settings DC motor driver board for the
output mode
   pinMode(MOTEUR_B_2, OUTPUT);
   Setpoint =80;  //Set the output value of the PID
   myPID.SetMode(AUTOMATIC);//PID is set to automatic mode
   myPID.SetSampleTime(100);//Set PID sampling frequency is 100ms
  EncoderInit();//Initialize the module

pinMode(RxDpin, INPUT);   //Configuration du Pin RxD (Receive) en mode entrée
pinMode(TxDpin, OUTPUT);  //Configuration du Pin TxD (Transmit) en mode sortie
BTSerie.begin(baudrate);
// Commandes AT pour le HC-05
// BTSerie.print("AT+NAME?");  //Demande le nom du module. Noter le ?
// BTSerie.print("AT+NAME=RCcar-HC-05-1");  //Définir le nom du module.
BTSerie.print("AT+VERSION?");  //Demande le N° de version. Noter le ?
// BTSerie.print("AT+UART?");  //Demande la vitesse série (baudrate). Noter le ?
// BTSerie.print("AT+UART=57600,0,0");  //Définir la vitesse série (baudrate).
// BTSerie.print("AT+ROLE?");  //Demande le mode du module, maitre ou esclave. Noter le ?
// BTSerie.print("AT+PSWD?");  //Demande le mot-de-passe du module. Noter le ?
// La console série de l'ordinateur d'où l'on envoie les commandes AT doit être réglée de
telle sorte que
// les fins de ligne soient « les deux, NL et CR », ce qui revient à envoyer \r\n à la fin de
chaque commande.
BTSerie.print("\r\n");     // sur HC-05, toutes les commandes doivent se terminer par \r\n
// afficher ce que le module bluetooth répond
  Serial.print( BTSerie.read() );     // afficher sur la console ce qui est lu sur BT
// pour AT+VERSION?, c'est le n° de version puis OK qui s'affiche

if (baudrate==38400) {
```

```
  Serial.println("En mode communication USB - Pret pour les commandes AT");
  Serial.println("Le HC-05 doit clignoter lentement (2 secondes)");
}
else if (baudrate==9600){
  Serial.println("En mode smartphone - Pret pour être appairé");
  Serial.println("Le HC-05 doit clignoter rapidement avant d'être appairé");
}
else{
  Serial.println("La vitesse de communication (baudrate) a été personnalisée");
}
monServo.attach(5);
delay(500);

pinMode(13,OUTPUT);   //left motors forward
pinMode(12,OUTPUT);   //left motors reverse
pinMode(11,OUTPUT);   //right motors forward
pinMode(10,OUTPUT);   //right motors reverse
pinMode(9,OUTPUT);   //Led
pinMode(5,OUTPUT);   //SG90 steering motor
  // A COMPLETER pour le servo //
//myservo.attach(5);  // attaches the servo on pin 5 to the servo object
}



void loop()
{

  //On lit caractere par caractere sur la liaion Arduino-HC05 et on affiche sur la liaison
Arduino-USB
  if (BTSerie.available()) {
    caractereTexte = BTSerie.read();
    Serial.print(caractereTexte);
  }

//On lit caractere par caractere sur la liaion Arduino-USB et on affiche sur la liaison
Arduino-HC05
  if (Serial.available()) {
    caractereTexte = Serial.read();
    BTSerie.write(caractereTexte);
    // Serial.println("Caractere envoye vers bluetooth : ");
    // Serial.println(caractereRecu);
```

```
  // }

if(caractereTexte == 'F'){            //move forward(all motors rotate in forward direction)
  advance();//Motor Forward
  monServo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'B'){      //move reverse (all motors rotate in reverse direction)
  back();//Motor reverse
  monServo.write(90);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'L'){      //turn right (left side motors rotate in forward
direction, right side motors doesn'caractereRecu rotate)
  left();
  monServo.write(60);                  // sets the servo position according to the scaled
value
}

else if(caractereTexte == 'R'){      //turn left (right side motors rotate in forward
direction, left side motors doesn'caractereRecu rotate)
  right();
  monServo.write(120);                  // sets the servo position according to the scaled
value
}


else if(caractereTexte == 'S'){      //STOP (all motors stop)
  Stop();
  monServo.write(90);
}
abs_duration=abs(duration);
      result=myPID.Compute();//PID conversion is complete and returns 1
      if(result)
      {
        Serial.print("Pluse: ");
        Serial.println(duration);
        duration = 0; //Count clear, wait for the next count
      }              // sets the servo position according to the scaled value
```

```
}
}
void EncoderInit()
{
  Direction = true;//default -> Forward
  pinMode(encoder0pinB,INPUT);
  attachInterrupt(0, wheelSpeed, CHANGE);
}

void wheelSpeed()
{
  int Lstate = digitalRead(encoder0pinA);
  if((encoder0PinALast == LOW) && Lstate==HIGH)
  {
    int val = digitalRead(encoder0pinB);
    if(val == LOW && Direction)
    {
      Direction = false; //Reverse
    }
    else if(val == HIGH && !Direction)
    {
      Direction = true;  //Forward
    }
  }
  encoder0PinALast = Lstate;

  if(!Direction)  duration++;
  else  duration--;

}
void advance()//Motor Forward
{
     digitalWrite(MOTEUR_A_1,LOW);
     digitalWrite(MOTEUR_A_2,val_output);
     digitalWrite(MOTEUR_B_1,val_output);
     digitalWrite(MOTEUR_B_2,LOW);
}
void back()//Motor reverse
{
     digitalWrite(MOTEUR_A_1,val_output);
     digitalWrite(MOTEUR_A_2,LOW);
```

```
      digitalWrite(MOTEUR_B_1,LOW);
      digitalWrite(MOTEUR_B_2,val_output);

}
void left()
{
      digitalWrite(MOTEUR_A_1,LOW);
      digitalWrite(MOTEUR_A_2,val_output);
      digitalWrite(MOTEUR_B_1,LOW);
      digitalWrite(MOTEUR_B_2,LOW);
}
void right()
{
      digitalWrite(MOTEUR_A_1,LOW);
      digitalWrite(MOTEUR_A_2,LOW);
      digitalWrite(MOTEUR_B_1,val_output);
      digitalWrite(MOTEUR_B_2,LOW);
}

void Stop()//Motor stops
{
      digitalWrite(MOTEUR_A_1, LOW);
      digitalWrite(MOTEUR_A_2, LOW);
      digitalWrite(MOTEUR_B_1, LOW);
      digitalWrite(MOTEUR_B_2, LOW);
}
```