Club Robotique - Voiture RC

Description du projet

Introduction

Pour ce cycle, nous allons créer une voiture contrôlable avec notre smartphone

Compétences techniques et mécaniques

- **Assemblage de pièces mécaniques** : Les élèves apprendront à assembler une structure mobile en fixant des moteurs, roues, châssis et autres composants physiques.
- **Transmission et mouvement** : Comprendre comment le mouvement des moteurs se traduit en déplacement des roues et en direction du véhicule.
- **Résolution de problèmes pratiques** : Adapter ou modifier le montage mécanique en cas de désalignement, d'instabilité ou de contraintes physiques.
- **Utilisation d'outils simples** : Utiliser des tournevis, pinces et autres petits outils pour fixer les composants.

Compétences en électronique

- **Câblage et connexion de composants** : Relier correctement les moteurs, le module Bluetooth, le module de contrôle moteur et l'Arduino Nano en respectant les schémas électroniques qu'ils auront réalisés.
- **Compréhension des circuits de base** : Identifier les entrées/sorties, l'alimentation, la masse, et comprendre le rôle de chaque module.
- **Utilisation de capteurs pour asservir des moteurs** : Grâce aux encodeurs intégrés dans les moteurs, les élèves apprennent comment mesurer la vitesse ou la position d'un moteur afin de synchroniser deux moteurs pour qu'ils aillent à la même vitesse.
- **Sécurité électronique** : Apprendre à manipuler des composants électroniques sans les endommager et à vérifier les connexions pour éviter les courts-circuits.

Compétences en programmation

- **Écriture de programmes Arduino (langage C/C++)** : Les collégiens apprendront à écrire des programmes simples pour contrôler les moteurs selon des consignes reçues via Bluetooth.
- **Communication série** : Compréhension de la communication entre le smartphone et l'Arduino via le module HC-05.
- **Lecture des capteurs** : Exploiter les données des encodeurs pour améliorer la précision du mouvement (vitesse, distance, etc.).
- **Organisation et logique du code** : Structuration du code en fonctions, boucles, conditions, et apprentissage de la logique algorithmique pour la commande du véhicule.

Compétences transversales

- **Travail en équipe** : Collaboration pour répartir les tâches (mécanique, câblage, codage, documentation).
- **Gestion de projet** : Planification des étapes, gestion du temps, anticipation des besoins matériels.
- **Communication et documentation** : Rédiger un dossier technique clair, expliquer leur démarche, et présenter leur projet à d'autres.
- **Créativité et innovation** : Personnalisation de la voiture, ajout de fonctionnalités (phares, klaxon, application mobile plus évoluée).
- **Résolution de problèmes** : Développer une capacité à identifier des dysfonctionnements et à tester des solutions.

Description du déroulé des séances:

~9 séances :

- Introduction du projet (brainstorming)
- ~3 séances sur TinkerCAD et sur l'impression 3D (modification de certaines pièces du Little Bot)
- ~1 séance sur l'actionneur : servomoteur RC 360°(déplacement)
- ~1 séance sur le capteur : ultrason (distance)
- ~1 séance sur le couplage capteur moteur
- ~1 séance sur l'assemblage du LittleBot
- ~2 séances sur la programmation du LittleBot

Matériel

Projet inspiré de : https://www.thingiverse.com/thing:2417739

- 1 x Arduino Nano ou compatible (seeeduino, funduino,...) : ~10€
- 1 x câble USB C : ~1€
- 1 x Sensor Shield pour Arduino Nano : ~3€
- 2 x servomoteur à rotation 360° (DM-S0090D-R 9g/0.08s/1.6kg.cm) : ~2€
- 1 x Module ultrason (HC-SR04) : ~2€
- 2 x élastique pour les roues
- 1 x porte pile 4xaa : ~2€
- Une imprimante 3D pour imprimer les pièces du robot (fichier disponible à la prochaine page)(~5€ de filament) :

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

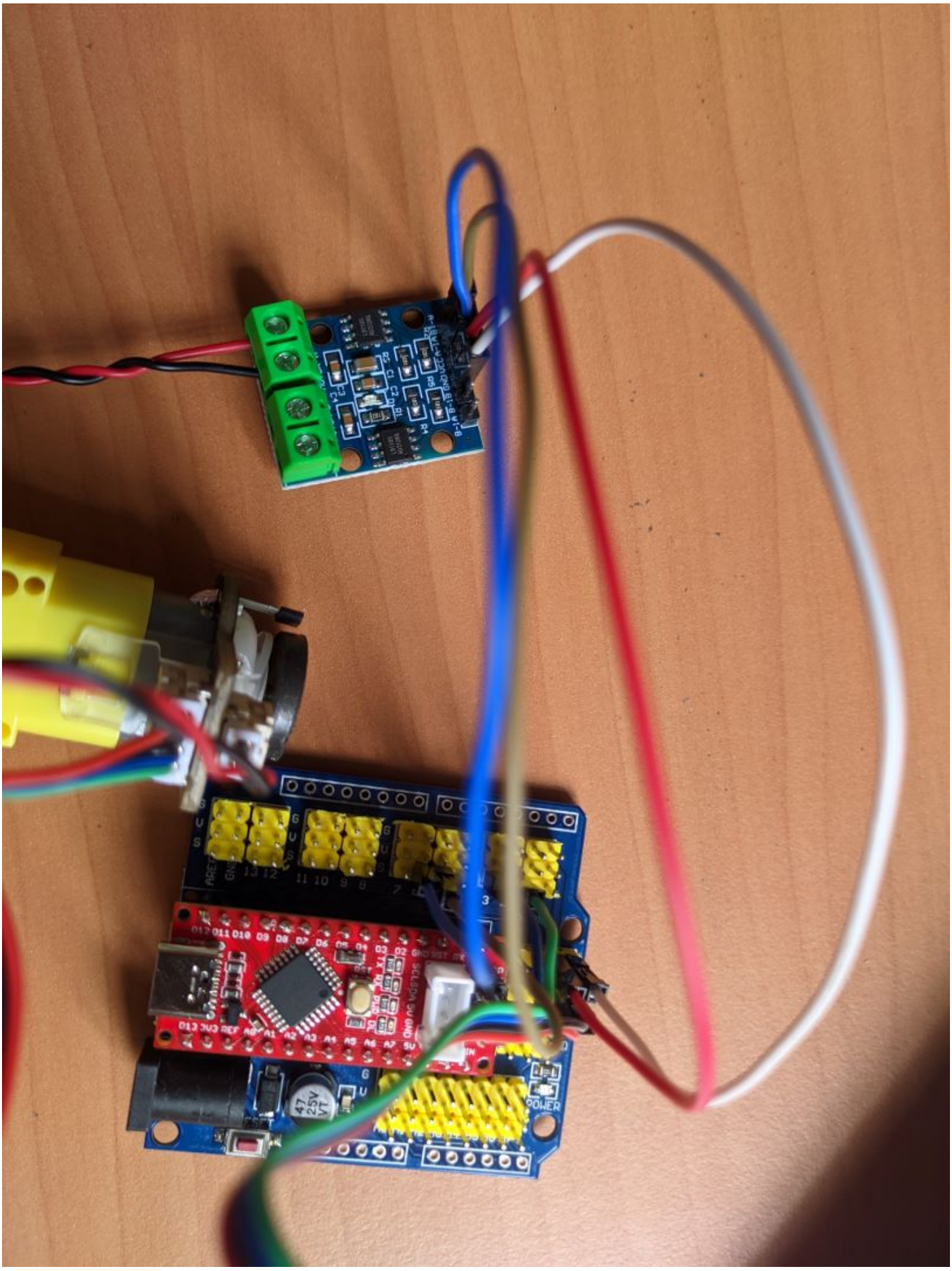
Moteur à courant continu "moteur jaune"

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

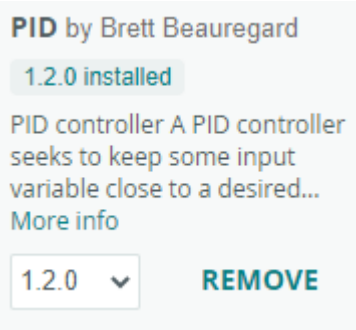
https://arduino.blaisepascal.fr/conversion-numeriqueanalogique-pwm/

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

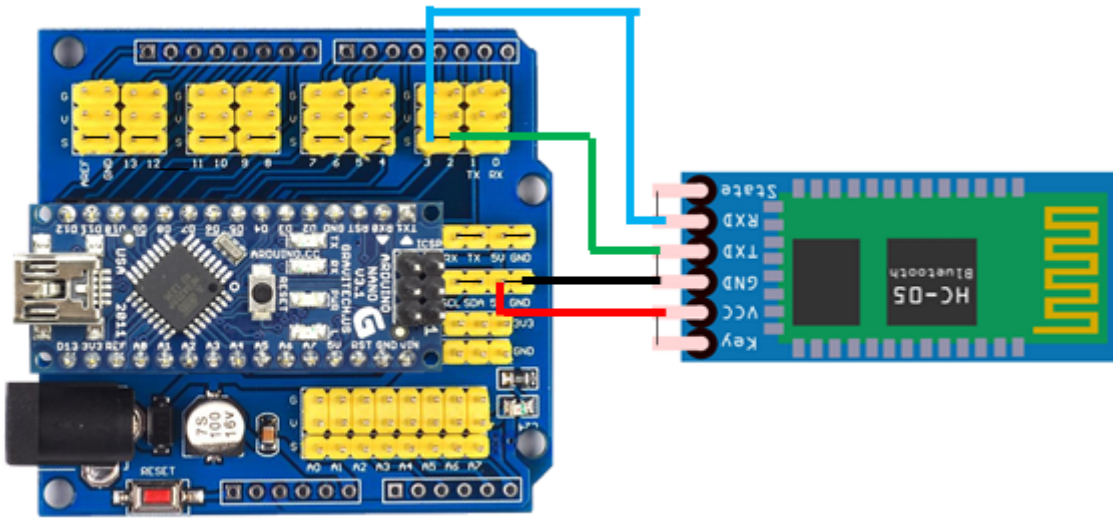
Test et configuration du module

- 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



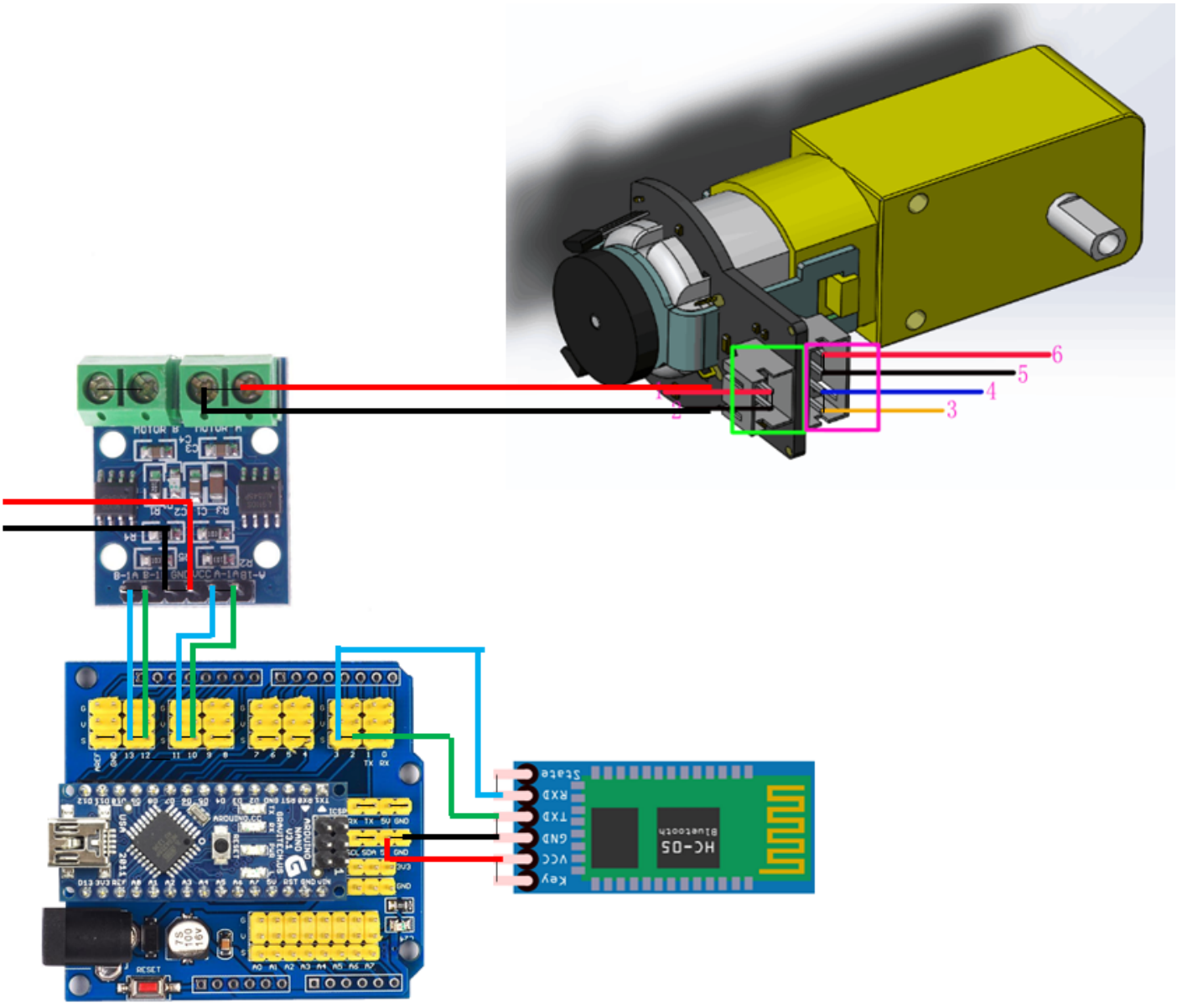
- Bien vérifier que le Pin qui **R**eçoit les données sur l'Arduino (**R**xDpin) soit connecté au Pin du HC-05 qui **T**ansmet (TxDpin). Et vice-versa, donc chaque câble transmet les données dans une direction opposée.
- Si on prend les Pins RX (Digital 0) et TX (Digital 1) de l'arduino pour connecter 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

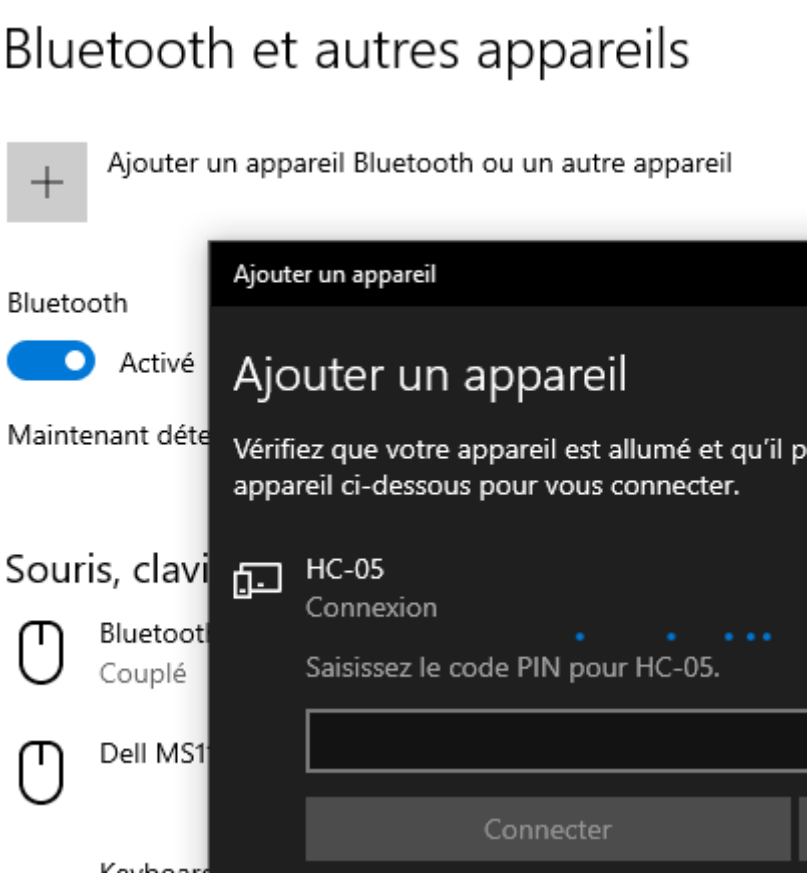
Depuis une application bluetooth de smartphone Android

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

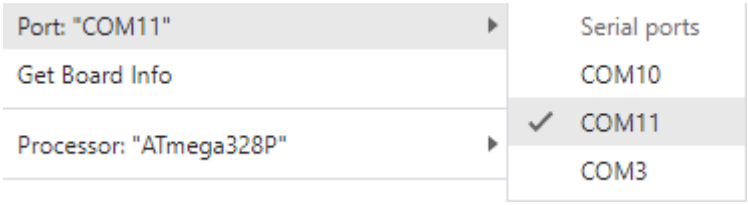
Depuis le moniteur série d'Arduino IDE

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`

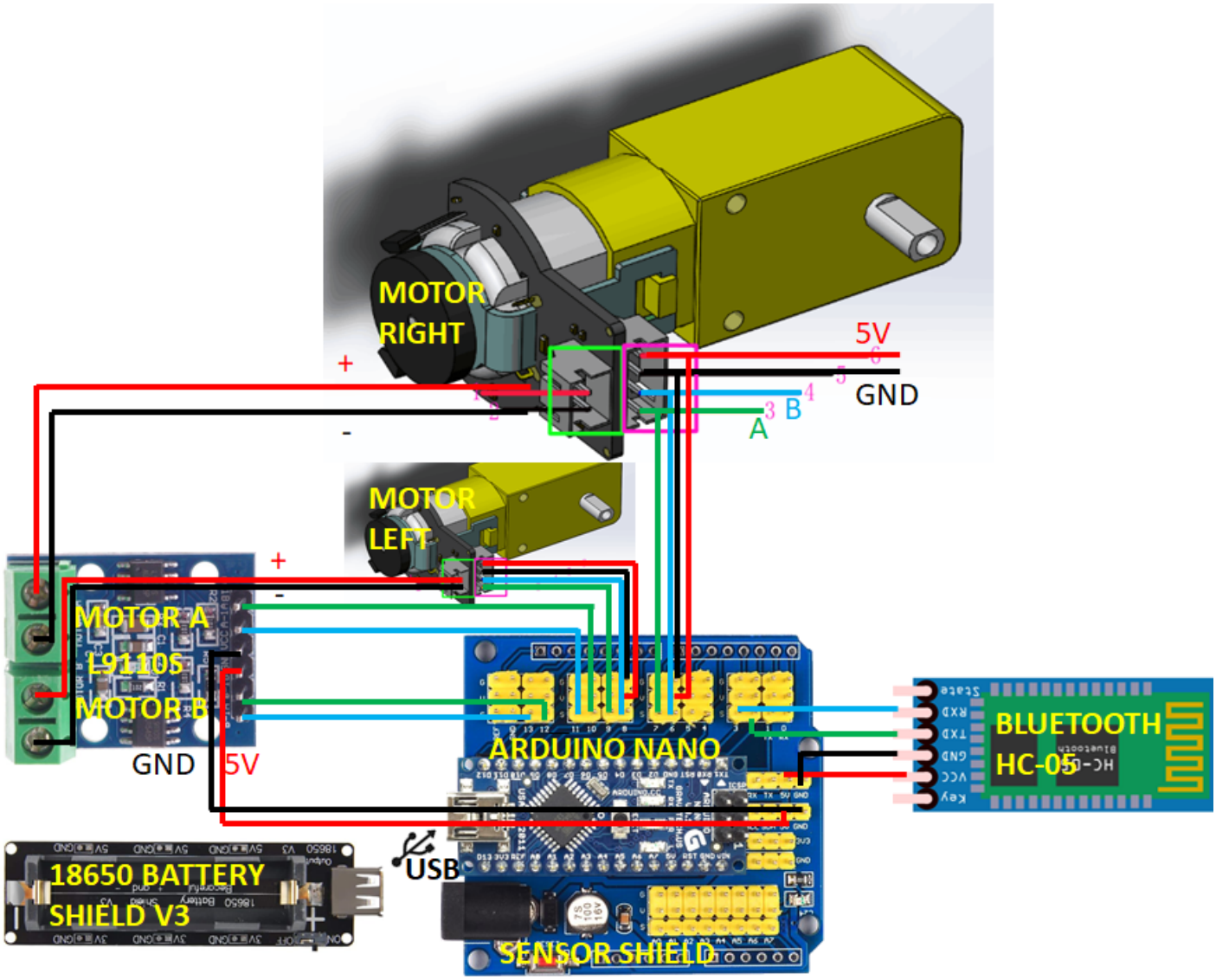


- Dans Arduino IDE
- Sélectionner le port série correspondant au HC-05. Il y en a deux, tester les deux



- 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--;

}
```

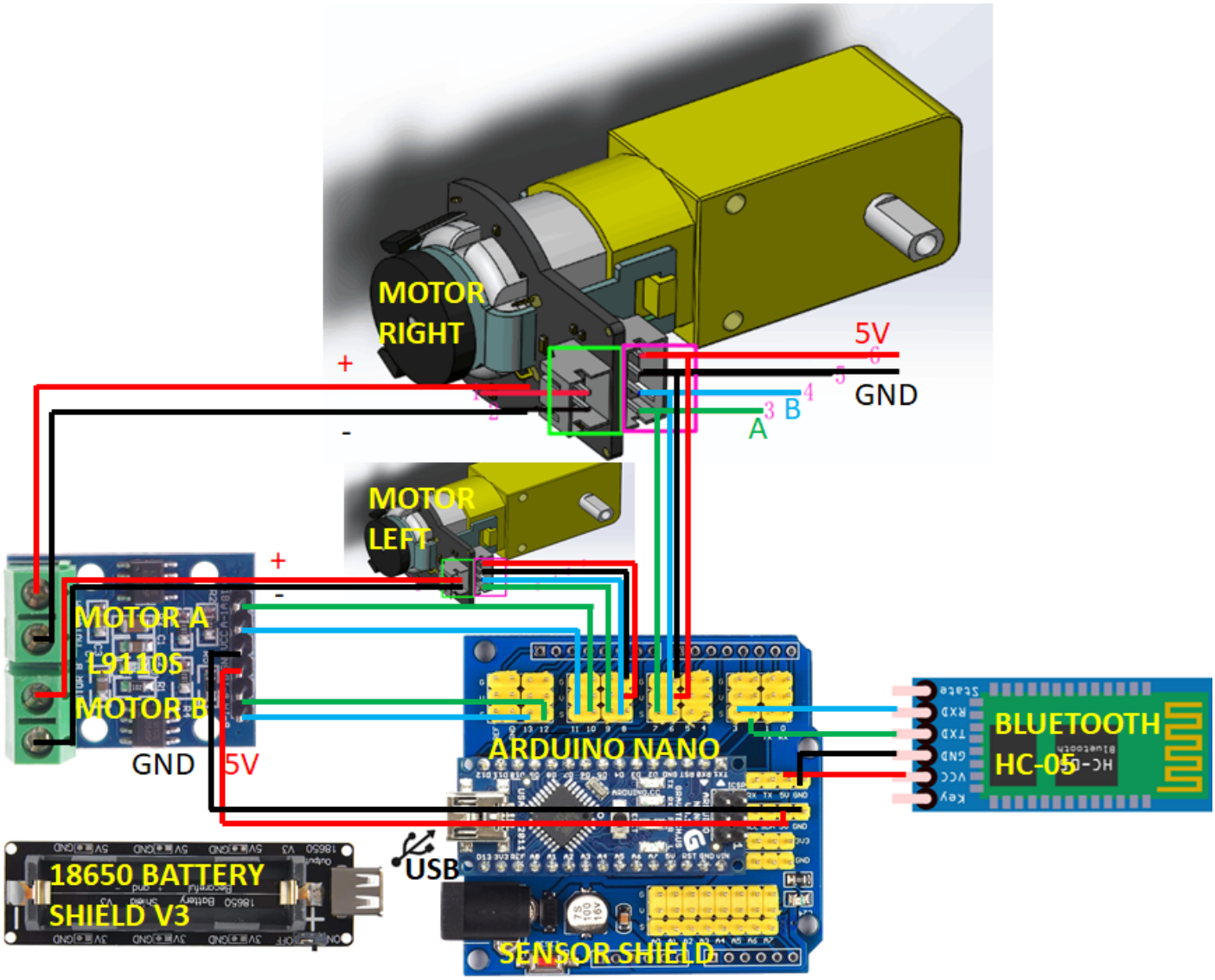
```
void advance()//Motor Forward
{
     digitalWrite(MOTEUR_A_1,LOW);
     digitalWrite(MOTEUR_A_2,val_output);
     digitalWrite(MOTEUR_B_1,val_output);
     digitalWrite(MOTEUR_B_2,LOW);
}
void back()//Motor reverse
{
     digitalWrite(MOTEUR_A_1,val_output);
     digitalWrite(MOTEUR_A_2,LOW);
  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