

# 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 liaison 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 liaison 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 liaison 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 liaison 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 liaison 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

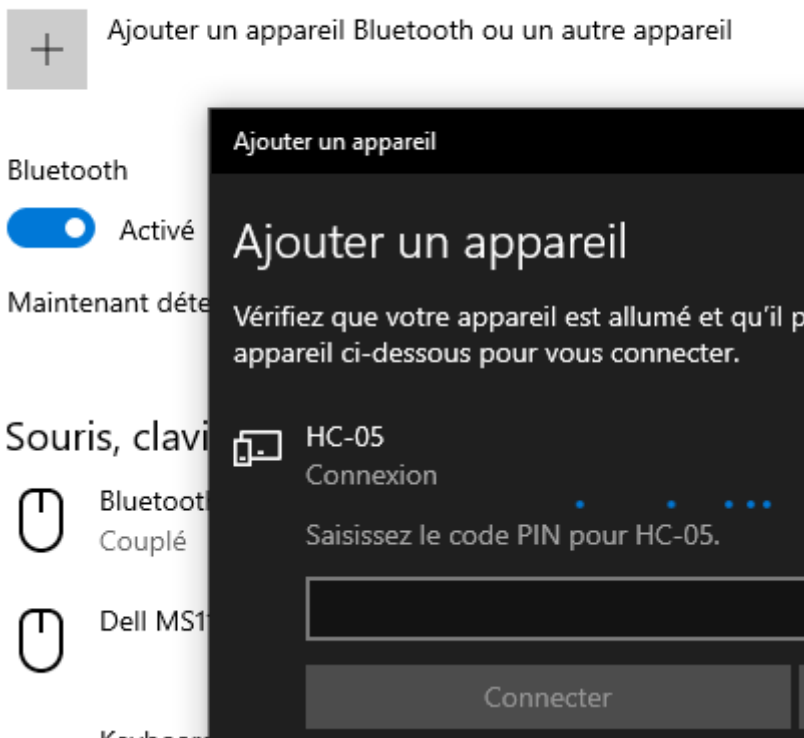
voir : <https://innovation.iha.unistra.fr/books/robotique-educative/page/le-module-bluetooth-hc-05#bkmrk-application-bluetoot>

## Depuis le moniteur série d'Arduino IDE

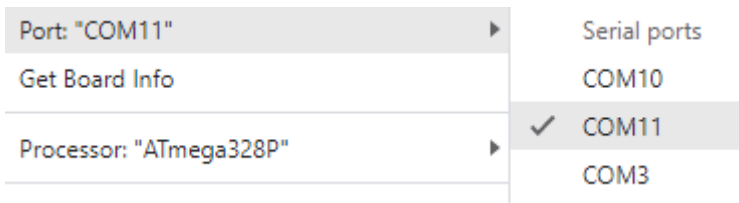
Source : <https://www.instructables.com/View-Serial-Monitor-Over-Bluetooth/#>

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

## Bluetooth et autres appareils



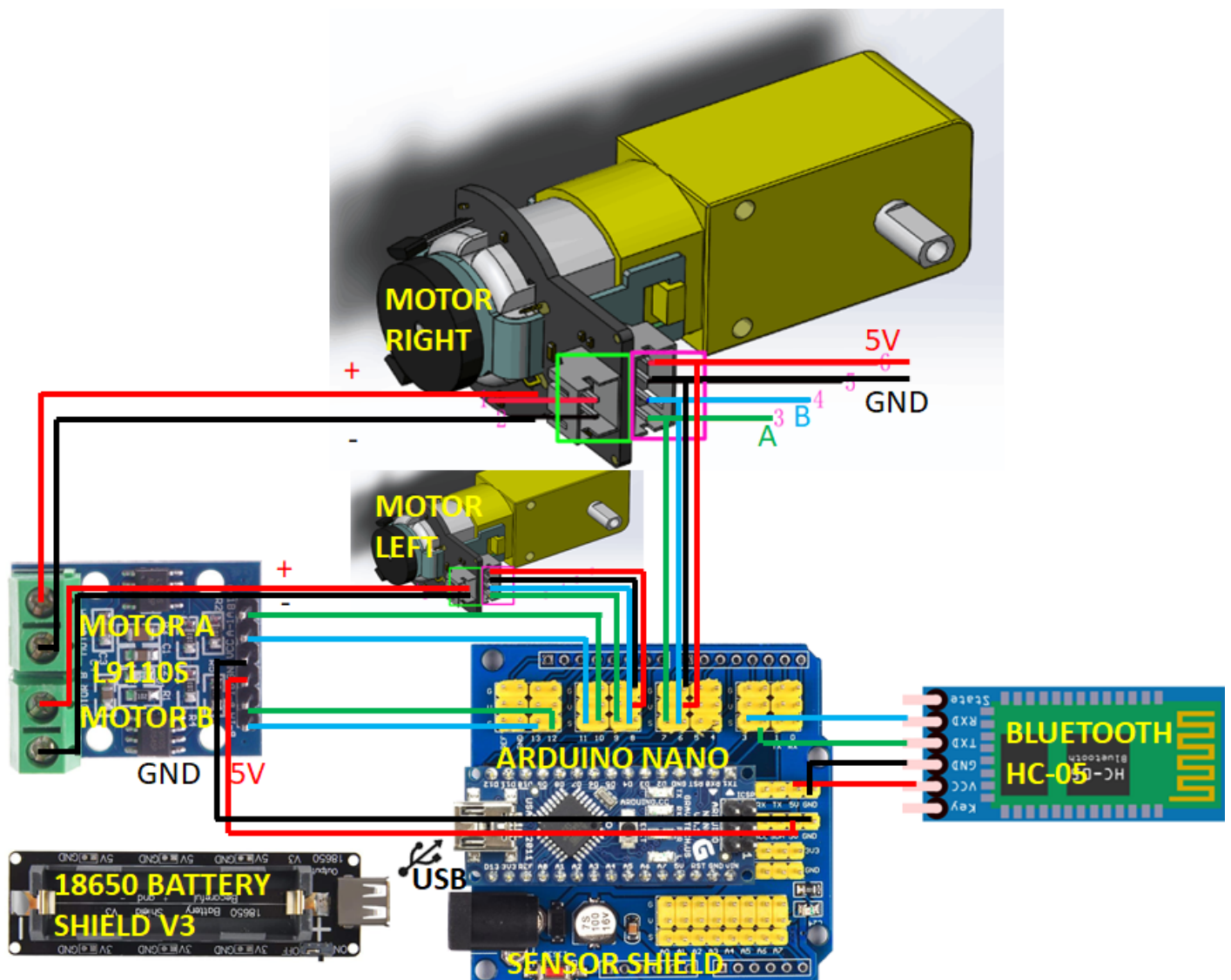
- Dans Arduino IDE
- Sélectionner le port série correspondant au HC-05. Il y en a deux, tester les deux



- 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 liaison 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 liaison 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 liaison 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't 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't 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>

---

Revision #11

Created 30 May 2024 11:41:00 by admin\_idf

Updated 19 June 2024 14:17:30 by admin\_idf