Robot à roues Mecanum

Robot à roues Mecanum

Qu'est-ce que Mecanum Wheel?

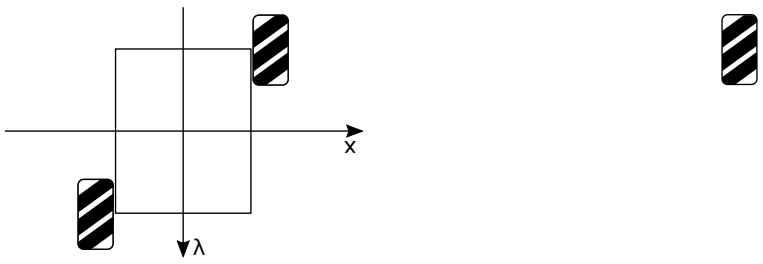
- La roue Mecanum est une roue omnidirectionnelle qui permet au véhicule robotique de se déplacer de chaque côté et le mouvement général avant et arrière. Bengt Erland Ilon a développé l'idée de roue Mecanum lorsqu'il était employé comme ingénieur dans une entreprise en Suède. Il a breveté l'idée aux États-Unis le 13 novembre 1972. L'autre nom répandu est la roue suédoise ou roue Ilon, d'après le nom du fondateur.

Conception du robot à roues Mecanum

La roue Mecanum est centrée sur une roue infatigable qui est reliée obliquement à toute la circonférence de sa jante avec un jeu de rouleaux externes caoutchoutés. Habituellement, ces ensembles de rouleaux externes caoutchoutés ont chacun un axe de rotation par rapport au plan de roue autour de 45 ° et avec la ligne d'axe autour de 45 °.

Dans chaque robot de roue Mecanum, sa roue a son groupe motopropulseur et est indépendamment de type non directionnel. Le groupe motopropulseur est chargé de générer une force de propulsion qui maintient un angle de 90 ° avec l'axe du rouleau pendant le mouvement de rotation qui peut être divisée en ses composantes vectorielles longitudinales et transversales.

Principe de commande des roues mecanum



La configuration typique d'un robot à roues Mecanum a une disposition à quatre roues, ce qui est

évident dans un exemple de robots mobiles omnidirectionnels appelé URANUS. Il a des rouleaux latéraux gauche et droit alternés avec des axes parallèles à la diagonale du châssis du véhicule au sommet de la roue. L'un des robots à roues Mecanum: URANUS est illustré ci-dessous.

Les robots à roues Mecanum pour se déplacer avec des exigences de vitesse minimales. Par example:

- Conduire les 4 roues à la même vitesse dans la même direction peut conduire à un mouvement d'avance / recul, car les vecteurs de force longitudinale s'additionneront, bien que les vecteurs transversaux s'annulent.
- Entraînement (toutes à la même vitesse) les deux roues dans un sens d'un côté et de l'autre dans le sens opposé. Cela se traduirait par la rotation stationnaire du véhicule lorsque le les vecteurs transversaux s'annulent tandis que les vecteurs longitudinaux les vecteurs s'apparient pour produire un couple autour de l'axe vertical central du véhicule.
- Pour vous déplacer latéralement, faites tourner les roues diagonales dans le même sens à la même vitesse et les deux autres roues diagonales seront à l'opposé. Cela entraîne l'ajout de vecteurs transversaux tout en annulant les vecteurs longitudinaux.

Les combinaisons de différents types de mouvements indépendants des roues dans le robot à roues Mecanum, aidées avec une certaine quantité de rotation, facilitent le mouvement du véhicule dans toutes les directions possibles.

Applications du robot à roues Mecanum

Le mouvement omnidirectionnel et la maniabilité extrême dans les environnements encombrés fournis par Mecanum Wheeled Robot ont trouvé leurs applications dans le domaine suivant:

- Le robot à roues Mecanum est utilisé dans diverses activités militaires et missions de recherche et de sauvetage.
- Mecanum Wheeled Robot est également utile dans les explorations planétaires, par exemple MarsCruiserOne, qui est un rover conceptuel habitable pour les futures missions spatiales.
- Mecanum Wheeled Robot est également utilisé dans les opérations minières, ce qui nécessite une bonne mobilité dans les espaces confinés.
- Les chariots élévateurs robotisés à roues Mecanum sont utilisés pour le transport de marchandises pour sa mobilité efficace et la gestion des espaces d'entrepôt.

- Les robots à roues Mecanum sont utilisés dans des projets de fauteuils roulants innovants, par exemple OMNI, car Mecanum Wheeled Robot facilite une mobilité élevée dans un environnement complexe et des degrés plus élevés d'indépendance.

Calculs savants ....

		Standard	Omni	Mecanum
kinematics	Vf	$\omega \cdot r$	$\omega \cdot r \cdot \sqrt{2}$	$\omega \cdot r$
	Vr	-	$\omega \cdot r \cdot \sqrt{2}$	$\omega \cdot r$
	Vd	-	$\omega \cdot r$	$\omega \cdot r / \sqrt{2}$
force	Ff	$4\tau / r$	$4\tau / (r\sqrt{2})$	$4\tau / r$
	Fr	-	$4\tau / (r\sqrt{2})$	$4\tau / r$
	Fd	-	$2\tau / r$	$2\tau\sqrt{2} / r$

The three columns are for standard, omni, and mecanum 4-wheeled vehicles, respectively. The omni vehicle's wheels are mounted at 45 degrees. All wheels same diameter. The first three rows are vehicle velocity: forward, strafe, and diagonal, for a given wheel speed ω (radians/sec)[1]. The second three rows are vehicle total pushing force: forward, strafe, and diagonal, for a given wheel torque τ. These last three rows assume a) frictionless mecanum and omni roller bearings, and b) sufficient traction to support the floor reaction forces.

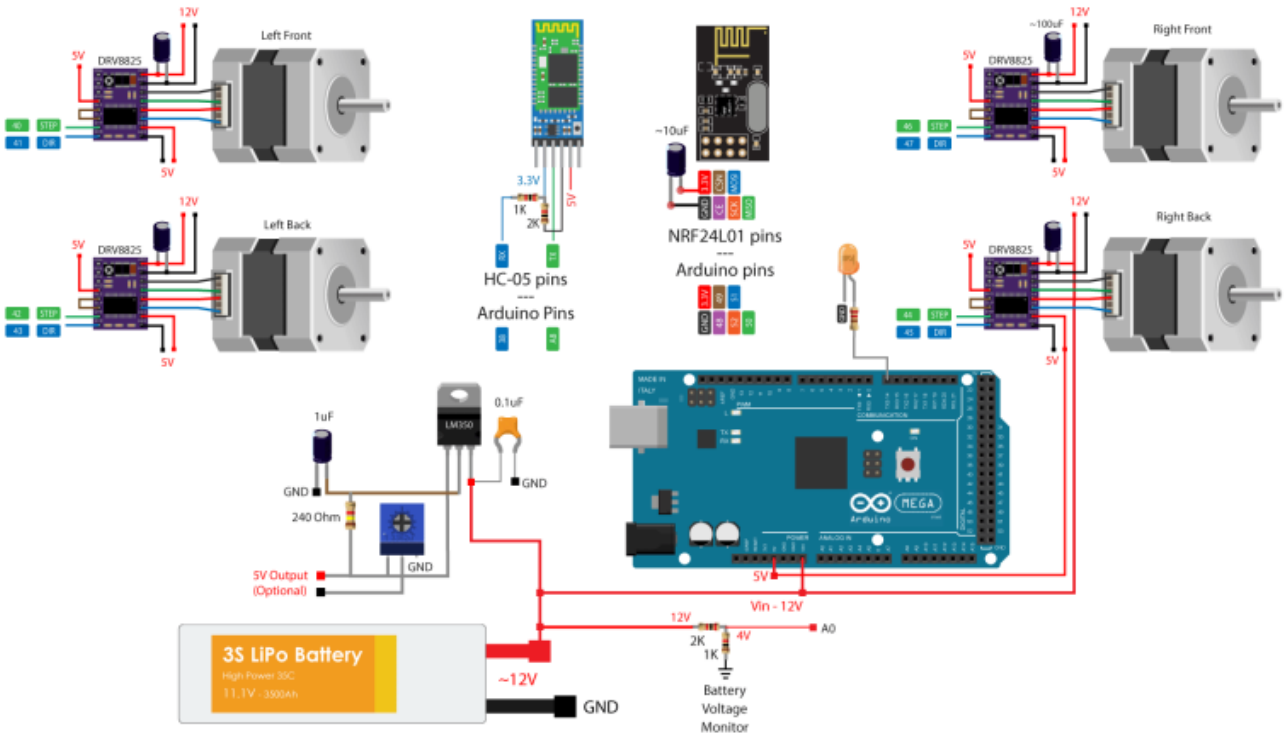
Bottom line: for the same wheel speeds, omni vehicle goes 41% faster than mecanum; for the same wheel torque, mecanum vehicle has 41% more pushing force than omni [2]

Cinématique d'une voiture à roue mecanum

Projet IMA 4 : Commande d'un robot mobile holonome et manipulateur

Modelisation_Mathematique_Mecanum.pdf

Arduino roue mecanum



Programmes

rouemecanumarduino.ino

rouemecanumarduino.ino

```
// à Tester ?

/*
   === Arduino Mecanum Wheels Robot ===
     Radio control with NRF24L01
  by Dejan, www.HowToMechatronics.com
  Libraries:
  RF24, https://github.com/tmrh20/RF24/
  AccelStepper by Mike McCauley:
http://www.airspayce.com/mikem/arduino/AccelStepper/index.html

*/

#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#include <AccelStepper.h>

RF24 radio(48, 49);   // nRF24L01 (CE, CSN)
```

```
const byte address[6] = "00001";
unsigned long lastReceiveTime = 0;
unsigned long currentTime = 0;

// Define the stepper motors and the pins the will use
AccelStepper LeftBackWheel(1, 42, 43);   // (Type:driver, STEP, DIR) - Stepper1
AccelStepper LeftFrontWheel(1, 40, 41);  // Stepper2
AccelStepper RightBackWheel(1, 44, 45);  // Stepper3
AccelStepper RightFrontWheel(1, 46, 47); // Stepper4

int wheelSpeed = 1500;

// Max size of this struct is 32 bytes - NRF24L01 buffer limit
struct Data_Package {
  byte j1PotX;
  byte j1PotY;
  byte j1Button;
  byte j2PotX;
  byte j2PotY;
  byte j2Button;
  byte pot1;
  byte pot2;
  byte tSwitch1;
  byte tSwitch2;
  byte button1;
  byte button2;
  byte button3;
  byte button4;
};
Data_Package data; //Create a variable with the above structure

void setup() {
  // Set initial seed values for the steppers
  LeftFrontWheel.setMaxSpeed(3000);
  LeftBackWheel.setMaxSpeed(3000);
  RightFrontWheel.setMaxSpeed(3000);
  RightBackWheel.setMaxSpeed(3000);

  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening(); //  Set the module as receiver

  Serial.begin(115200);
}

void loop() {
  // Check whether we keep receving data, or we have a connection
```

```
between the two modules
  currentTime = millis();
  if ( currentTime - lastReceiveTime > 1000 ) { // If current time is
more then 1 second since we have recived the last data, that means we
have lost connection
    resetData(); // If connection is lost, reset the data. It prevents
unwanted behavior, for example if a drone jas a throttle up, if we lose
connection it can keep flying away if we dont reset the function
  }
  // Check whether there is data to be received
  if (radio.available()) {
    radio.read(&data, sizeof(Data_Package)); // Read the whole data and
store it into the 'data' structure
    lastReceiveTime = millis(); // At this moment we have received the
data
  }
  // Set speed - left potentiometer
  wheelSpeed = map(data.pot1, 0, 255, 100, 3000);

  if (data.j1PotX > 150) {
    moveSidewaysLeft();
  }
  else if (data.j1PotX < 100) {
    moveSidewaysRight();
  }
  else if (data.j1PotY > 160) {
    moveForward();
  }
  else if (data.j1PotY < 100) {
    moveBackward();
  }
  else if (data.j2PotX < 100 & data.j2PotY > 160) {
    moveRightForward();
  }
  else if (data.j2PotX > 160 & data.j2PotY > 160) {
    moveLeftForward();
  }
  else if (data.j2PotX < 100 & data.j2PotY < 100) {
    moveRightBackward();
  }
  else if (data.j2PotX > 160 & data.j2PotY < 100) {
    moveLeftBackward();
  }
  else if (data.j2PotX < 100) {
    rotateRight();
  }
  else if (data.j2PotX > 150) {
    rotateLeft();
  }
```

```
  else {
    stopMoving();
  }
  // Execute the steps
  LeftFrontWheel.runSpeed();
  LeftBackWheel.runSpeed();
  RightFrontWheel.runSpeed();
  RightBackWheel.runSpeed();

  // Monitor the battery voltage
  int sensorValue = analogRead(A0);
  float voltage = sensorValue * (5.0 / 1023.00) * 3; // Convert the
reading values from 5v to suitable 12V i
  // If voltage is below 11V turn on the LED
  if (voltage < 11) {
    digitalWrite(led, HIGH);
  }
  else {
    digitalWrite(led, LOW);
  }
}

void moveForward() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
}
void moveBackward() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void moveSidewaysRight() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
}
void moveSidewaysLeft() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void rotateLeft() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
```

```
}
void rotateRight() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void moveRightForward() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(wheelSpeed);
}
void moveRightBackward() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(0);
}
void moveLeftForward() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(0);
}
void moveLeftBackward() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void stopMoving() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(0);
}


void resetData() {
  // Reset the values when there is no radio connection - Set initial
default values
  data.j1PotX = 127;
  data.j1PotY = 127;
  data.j2PotX = 127;
  data.j2PotY = 127;
  data.j1Button = 1;
  data.j2Button = 1;
  data.pot1 = 1;
```

```
  data.pot2 = 1;
  data.tSwitch1 = 1;
  data.tSwitch2 = 1;
  data.button1 = 1;
  data.button2 = 1;
  data.button3 = 1;
  data.button4 = 1;
}
```

Robot-Arduino_Mecanum000.ino

Robot-Arduino_Mecanum000.ino

```
//A tester

/*
MECANUM WHEEL ROBOT - BLUETOOTH CONTROLLED v1.0
 - Allows you to control a mecanum robot via bluetooth
 - Tested with Arduino Mega 2560
 - Android application -
https://play.google.com/store/apps/details?id=pl.mobilerobots.vacuumcle
anerrobot&hl=pl
 - Project description -
http://www.instructables.com/id/Mecanum-wheel-robot-bluetooth-controlle
d
 Author: Adam Srebro
 www: http://www.mobilerobots.pl/

 Connections:
 Bluetooth (e.g HC-06)-> Arduino Mega 2560
 TXD - TX1 (19)
 RXD - RX1 (18)
 VCC - 5V
 GND - GND

 TB6612FNG Dual Motor Driver -> Arduino Mega 2560
 //PWM control
 RightFrontMotor_PWMA - 2
 LeftFrontMotor_PWMB - 3
 RightRearMotor_PWMA - 4
 LeftRearMotor_PWMB - 5
 //Control of rotation direction
 RightFrontMotor_AIN1 - 22
 RightFrontMotor_AIN2 - 23
 LeftFrontMotor_BIN1 - 24
 LeftFrontMotor_BIN2 - 25
 RightRearMotor_AIN1 - 26
 RightRearMotor_AIN2 - 27
```

```
 LeftRearMotor_BIN1 - 28
 LeftRearMotor_BIN2 - 29
 //The module and motors power supply
 STBY - Vcc
 VMOT - motor voltage (4.5 to 13.5 V) - 11.1V from LiPo battery
 Vcc - logic voltage (2.7 to 5.5) - 5V from Arduino
 GND - GND

 TB6612FNG Dual Motor Driver -> DC Motors
 MotorDriver1_AO1 - RightFrontMotor
 MotorDriver1_A02 - RightFrontMotor
 MotorDriver1_B01 - LeftFrontMotor
 MotorDriver1_B02 - LeftFrontMotor

 MotorDriver2_AO1 - RightRearMotor
 MotorDriver2_A02 - RightRearMotor
 MotorDriver2_B01 - LeftRearMotor
 MotorDriver2_B02 - LeftRearMotor
 */
#include <Wire.h>
#include <math.h>
/*TB6612FNG Dual Motor Driver Carrier*/
const int RightFrontMotor_PWM = 2; // pwm output
const int LeftFrontMotor_PWM = 3; // pwm output
const int RightRearMotor_PWM = 4; // pwm output
const int LeftRearMotor_PWM = 5; // pwm output
//Front motors
const int RightFrontMotor_AIN1 = 22; // control Input AIN1 - right
front motor
const int RightFrontMotor_AIN2 = 23; // control Input AIN2 - right
front motor
const int LeftFrontMotor_BIN1 = 24; // control Input BIN1 - left front
motor
const int LeftFrontMotor_BIN2 = 25; // control Input BIN2 - left front
motor
//Rear motors
const int RightRearMotor_AIN1 = 26; // control Input AIN1 - right rear
motor
const int RightRearMotor_AIN2 = 27; // control Input AIN2 - right rear
motor
const int LeftRearMotor_BIN1 = 28; // control Input BIN1 - left rear
motor
const int LeftRearMotor_BIN2 = 29; // control Input BIN2 - left rear
motor

long pwmLvalue = 255;
long pwmRvalue = 255;
byte pwmChannel;
const char startOfNumberDelimiter = '<';
```

```
const char endOfNumberDelimiter = '>';

void setup(){
  Serial1.begin(9600);// HC-06 default baudrate: 9600

  //Setup RightFrontMotor
  pinMode(RightFrontMotor_AIN1, OUTPUT); //Initiates Motor Channel A1
pin
  pinMode(RightFrontMotor_AIN2, OUTPUT); //Initiates Motor Channel A2
pin

  //Setup LeftFrontMotor
  pinMode(LeftFrontMotor_BIN1, OUTPUT); //Initiates Motor Channel B1
pin
  pinMode(LeftFrontMotor_BIN2, OUTPUT); //Initiates Motor Channel B2
pin

  //Setup RightFrontMotor
  pinMode(RightRearMotor_AIN1, OUTPUT); //Initiates Motor Channel A1
pin
  pinMode(RightRearMotor_AIN2, OUTPUT); //Initiates Motor Channel A2
pin

  //Setup LeftFrontMotor
  pinMode(LeftRearMotor_BIN1, OUTPUT); //Initiates Motor Channel B1 pin
  pinMode(LeftRearMotor_BIN2, OUTPUT); //Initiates Motor Channel B2 pin

  Wire.begin();
}// void setup()

void loop(){
  if (Serial1.available()) {
    processInput();
  }
}// void loop()

void motorControl(String motorStr,int mdirection, int mspeed){
  int IN1;
  int IN2;
  int motorPWM;
  if (motorStr == "rf") {       //right front
    IN1 = RightFrontMotor_AIN1;
    IN2 = RightFrontMotor_AIN2;
    motorPWM = RightFrontMotor_PWM;
  }
  else if (motorStr == "lf") { //left front
    IN1 = LeftFrontMotor_BIN1;
    IN2 = LeftFrontMotor_BIN2;
    motorPWM = LeftFrontMotor_PWM;
  }
  else if (motorStr == "rr") {
```

```
    IN1 = RightRearMotor_AIN1;
    IN2 = RightRearMotor_AIN2;
    motorPWM = RightRearMotor_PWM;
  }
  else if (motorStr == "lr") {
    IN1 = LeftRearMotor_BIN1;
    IN2 = LeftRearMotor_BIN2;
    motorPWM = LeftRearMotor_PWM;
  }
  if (mdirection == 1){
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  }
  else if (mdirection == -1){
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  }
  analogWrite(motorPWM, mspeed);
}

void processInput (){
  static long receivedNumber = 0;
  static boolean negative = false;
  byte c = Serial1.read ();

  switch (c){
  case endOfNumberDelimiter:
    if (negative)
      SetPWM(- receivedNumber, pwmChannel);
    else
      SetPWM(receivedNumber, pwmChannel);

    // fall through to start a new number
  case startOfNumberDelimiter:
    receivedNumber = 0;
    negative = false;
    pwmChannel = 0;
    break;

  case 'f': // Go FORWARD
    goForward(255);
    //Serial.println("forward");
    break;

  case 'b': // Go BACK
    goBackwad(255);
    //Serial.println("backward");
    break;
```

```
  case 'r':
    moveRight(255);
    break;

  case 'l':
    moveLeft(255);
    break;

  case 'i':
    turnRight(255);
    break;

  case 'j':
    turnLeft(255);
    break;

  case 'c': // Top Right
    moveRightForward(255);
    break;

  case 'd': // Top Left
    moveLeftForward(255);
    break;

  case 'e': // Bottom Right
    moveRightBackward(255);
    break;

  case 'h': // Bottom Left
    moveLeftBackward(255);
    break;

  case 's':
    hardStop();
    break;

  case 'x':
    pwmChannel = 1; // RightFrontMotor_PWM
    break;
  case 'y': // LeftFrontMotor_PWM
    pwmChannel = 2;
    break;

  case '0' ... '9':
    receivedNumber *= 10;
    receivedNumber += c - '0';
    break;

  case '-':
    negative = true;
    break;
```

```
  } // end of switch
} // void processInput ()

void goForward(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", 1, mspeed);
}// void goForward(int mspeed)

void goBackwad(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", -1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", -1, mspeed);
}// void goBackwad(int mspeed)

void moveRight(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", -1, mspeed);
}// void moveRight(int mspeed)

void moveLeft(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", -1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", 1, mspeed);
}// void moveLeft(int mspeed)

void moveRightForward(int mspeed){
  motorControl("rf", 1, 0);
  motorControl("lf", 1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", 1, 0);
}// void  moveRightForward(int mspeed)

void moveRightBackward(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", 1, 0);
  motorControl("rr", 1, 0);
  motorControl("lr", -1, mspeed);
}// void  moveRightBackward(int mspeed)

void moveLeftForward(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", 1, 0);
  motorControl("rr", 1, 0);
```

```
  motorControl("lr", 1, mspeed);
}// void  moveLeftForward(int mspeed)

void moveLeftBackward(int mspeed){
  motorControl("rf", 1, 0);
  motorControl("lf", -1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", 1, 0);
}// void  moveLeftBackward(int mspeed)

void turnRight(int mspeed){
  motorControl("rf", -1, mspeed);
  motorControl("lf", 1, mspeed);
  motorControl("rr", -1, mspeed);
  motorControl("lr", 1, mspeed);
}// void turnRight(int mspeed)

void turnLeft(int mspeed){
  motorControl("rf", 1, mspeed);
  motorControl("lf", -1, mspeed);
  motorControl("rr", 1, mspeed);
  motorControl("lr", -1, mspeed);
}// void turnRight(int mspeed)

void stopRobot(int delay_ms){
  analogWrite(RightFrontMotor_PWM, 0);
  analogWrite(LeftFrontMotor_PWM, 0);
  analogWrite(RightRearMotor_PWM, 0);
  analogWrite(LeftRearMotor_PWM, 0);
  delay(delay_ms);
}// void stopRobot(int delay_ms)

void hardStop(){
  analogWrite(RightFrontMotor_PWM, 0);
  analogWrite(LeftFrontMotor_PWM, 0);
  analogWrite(RightRearMotor_PWM, 0);
  analogWrite(LeftRearMotor_PWM, 0);
}// void stopRobot()

void SetPWM (const long pwm_num, byte pwm_channel){
  if(pwm_channel==1){ // DRIVE MOTOR
    analogWrite(RightFrontMotor_PWM, pwm_num);
    pwmRvalue = pwm_num;
  }
  else if(pwm_channel==2){ // STEERING MOTOR
    analogWrite(LeftFrontMotor_PWM, pwm_num);
    pwmLvalue = pwm_num;
  }
}// void SetPWM (const long pwm_num, byte pwm_channel)
```

Autres exemples robot arduino avec roues mecanum

arduino mega mecanum

Robot-Arduino_Mecanum_001.ino

Robot-Arduino_Mecanum_001.ino

```
//A tester


/*
 * Arduino Mecanum Omni Direction Wheel Robot Car
 * Tutorial URL https://osoyoo.com/?p=43404
 * CopyRight www.osoyoo.com

 * After running the code, smart car will
 * go forward and go backward for 2 seconds,
 * left turn and right turn for 2 seconds,
 * right shift and left shift for 2 seconds,
 * left diagonal back and right diagonal forward for 2 seconds,
 * left diagonal forward and right diagonal back for 2 seconds,
 * then stop.
 *
 */
#define SPEED 100
#define TURN_SPEED 100
#define speedPinR 9   //  Front Wheel PWM pin connect Model-Y M_B ENA
#define RightMotorDirPin1  22    //Front Right Motor direction pin 1 to
Model-Y M_B IN1  (K1)
#define RightMotorDirPin2  24   //Front Right Motor direction pin 2 to
Model-Y M_B IN2   (K1)
#define LeftMotorDirPin1  26    //Front Left Motor direction pin 1 to
Model-Y M_B IN3 (K3)
#define LeftMotorDirPin2  28   //Front Left Motor direction pin 2 to
Model-Y M_B IN4 (K3)
#define speedPinL 10   //  Front Wheel PWM pin connect Model-Y M_B ENB

#define speedPinRB 11   //  Rear Wheel PWM pin connect Left Model-Y M_A
ENA
#define RightMotorDirPin1B  5    //Rear Right Motor direction pin 1 to
Model-Y M_A IN1 ( K1)
```

```
#define RightMotorDirPin2B 6    //Rear Right Motor direction pin 2 to
Model-Y M_A IN2 ( K1)
#define LeftMotorDirPin1B 7    //Rear Left Motor direction pin 1 to
Model-Y M_A IN3  (K3)
#define LeftMotorDirPin2B 8  //Rear Left Motor direction pin 2 to
Model-Y M_A IN4 (K3)
#define speedPinLB 12    //  Rear Wheel PWM pin connect Model-Y M_A ENB

/*motor control*/
void go_advance(int speed){
   RL_fwd(speed);
   RR_fwd(speed);
   FR_fwd(speed);
   FL_fwd(speed);
}
void go_back(int speed){
   RL_bck(speed);
   RR_bck(speed);
   FR_bck(speed);
   FL_bck(speed);
}
void right_shift(int speed_fl_fwd,int speed_rl_bck ,int
speed_rr_fwd,int speed_fr_bck) {
  FL_fwd(speed_fl_fwd);
  RL_bck(speed_rl_bck);
  RR_fwd(speed_rr_fwd);
  FR_bck(speed_fr_bck);
}
void left_shift(int speed_fl_bck,int speed_rl_fwd ,int speed_rr_bck,int
speed_fr_fwd){
   FL_bck(speed_fl_bck);
   RL_fwd(speed_rl_fwd);
   RR_bck(speed_rr_bck);
   FR_fwd(speed_fr_fwd);
}

void left_turn(int speed){
   RL_bck(0);
   RR_fwd(speed);
   FR_fwd(speed);
   FL_bck(0);
}
void right_turn(int speed){
 //  RL_fwd(speed);
//   RR_bck(0);
 //  FR_bck(0);
 //  FL_fwd(speed);

   RL_fwd(speed);
   RR_fwd(0);
   FR_fwd(0);
```

```
   FL_fwd(speed);
}
void left_back(int speed){
   RL_fwd(0);
   RR_bck(speed);
   FR_bck(speed);
   FL_fwd(0);
}
void right_back(int speed){
   RL_bck(speed);
   RR_fwd(0);
   FR_fwd(0);
   FL_bck(speed);
}
void clockwise(int speed){
   RL_fwd(speed);
   RR_bck(speed);
   FR_bck(speed);
   FL_fwd(speed);
}
void countclockwise(int speed){
   RL_bck(speed);
   RR_fwd(speed);
   FR_fwd(speed);
   FL_bck(speed);
}


void FR_fwd(int speed)  //front-right wheel forward turn
{
  digitalWrite(RightMotorDirPin1,HIGH);
  digitalWrite(RightMotorDirPin2,LOW);
  analogWrite(speedPinR,speed);
}
void FR_bck(int speed) // front-right wheel backward turn
{
  digitalWrite(RightMotorDirPin1,LOW);
  digitalWrite(RightMotorDirPin2,HIGH);
  analogWrite(speedPinR,speed);
}
void FL_fwd(int speed) // front-left wheel forward turn
{
  digitalWrite(LeftMotorDirPin1,HIGH);
  digitalWrite(LeftMotorDirPin2,LOW);
  analogWrite(speedPinL,speed);
}
void FL_bck(int speed) // front-left wheel backward turn
{
  digitalWrite(LeftMotorDirPin1,LOW);
```

```
  digitalWrite(LeftMotorDirPin2,HIGH);
  analogWrite(speedPinL,speed);
}

void RR_fwd(int speed)  //rear-right wheel forward turn
{
  digitalWrite(RightMotorDirPin1B, HIGH);
  digitalWrite(RightMotorDirPin2B,LOW);
  analogWrite(speedPinRB,speed);
}
void RR_bck(int speed)  //rear-right wheel backward turn
{
  digitalWrite(RightMotorDirPin1B, LOW);
  digitalWrite(RightMotorDirPin2B,HIGH);
  analogWrite(speedPinRB,speed);
}
void RL_fwd(int speed)  //rear-left wheel forward turn
{
  digitalWrite(LeftMotorDirPin1B,HIGH);
  digitalWrite(LeftMotorDirPin2B,LOW);
  analogWrite(speedPinLB,speed);
}
void RL_bck(int speed)    //rear-left wheel backward turn
{
  digitalWrite(LeftMotorDirPin1B,LOW);
  digitalWrite(LeftMotorDirPin2B,HIGH);
  analogWrite(speedPinLB,speed);
}

void stop_Stop()    //Stop
{
  analogWrite(speedPinLB,0);
  analogWrite(speedPinRB,0);
  analogWrite(speedPinL,0);
  analogWrite(speedPinR,0);
}


//Pins initialize
void init_GPIO()
{
  pinMode(RightMotorDirPin1, OUTPUT);
  pinMode(RightMotorDirPin2, OUTPUT);
  pinMode(speedPinL, OUTPUT);

  pinMode(LeftMotorDirPin1, OUTPUT);
  pinMode(LeftMotorDirPin2, OUTPUT);
  pinMode(speedPinR, OUTPUT);
  pinMode(RightMotorDirPin1B, OUTPUT);
  pinMode(RightMotorDirPin2B, OUTPUT);
  pinMode(speedPinLB, OUTPUT);
```

```
  pinMode(LeftMotorDirPin1B, OUTPUT);
  pinMode(LeftMotorDirPin2B, OUTPUT);
  pinMode(speedPinRB, OUTPUT);

  stop_Stop();
}

void setup()
{
  init_GPIO();

go_advance(SPEED);
     delay(1000);
     stop_Stop();
     delay(1000);

go_back(SPEED);
      delay(1000);
      stop_Stop();
      delay(1000);

left_turn(TURN_SPEED);
      delay(1000);
      stop_Stop();
      delay(1000);

right_turn(TURN_SPEED);
     delay(1000);
     stop_Stop();
     delay(1000);

right_shift(200,200,200,200); //right shift
     delay(1000);
     stop_Stop();
     delay(1000);

left_shift(200,200,200,200); //left shift
     delay(1000);
     stop_Stop();
     delay(1000);

left_shift(200,0,200,0); //left diagonal back
     delay(1000);
     stop_Stop();
     delay(1000);

right_shift(200,0,200,0); //right diagonal ahead
     delay(1000);
```

```
     stop_Stop();
     delay(1000);

left_shift(0,200,0,200); //left diagonal ahead
     delay(1000);
     stop_Stop();
     delay(1000);

right_shift(0,200,0,200); //right diagonal back
     delay(1000);
     stop_Stop();
     delay(1000);
}

void loop(){
}
```

Robot-Arduino_Mecanum_002.ino

Robot-Arduino_Mecanum_001.ino

```
/*
 * / _ \ /___)/ _ \| | | |/ _ \ / _ \ / ___) _ \|    \
 *| |_| |___ | |_| | |_| | |_| | |_| ( (__| |_| | | | |
 * \___/(___/ \___/ \__  |\___/ \___(_)____)___/|_|_|_|
 *                  (____/
 * Omni direction Mecanum Wheel Arduino Smart Car Tutorial Lesson 2
Obstacle avoidance auto driving.
 * Tutorial URL https://osoyoo.com/?p=43404
 * CopyRight www.osoyoo.com

 * This project will show you how to make osoyoo robot car in auto
drive mode and avoid obstacles
 *
 *
 */
#include <Servo.h>
#define speedPinR 9   //  Front Wheel PWM pin connect Model-Y M_B ENA
#define RightMotorDirPin1  22    //Front Right Motor direction pin 1 to
Model-Y M_B IN1  (K1)
#define RightMotorDirPin2  24   //Front Right Motor direction pin 2 to
Model-Y M_B IN2   (K1)
#define LeftMotorDirPin1  26    //Front Left Motor direction pin 1 to
Model-Y M_B IN3 (K3)
#define LeftMotorDirPin2  28   //Front Left Motor direction pin 2 to
Model-Y M_B IN4 (K3)
#define speedPinL 10   //  Front Wheel PWM pin connect Model-Y M_B ENB
```

```
#define speedPinRB 11   //  Rear Wheel PWM pin connect Left Model-Y M_A
ENA
#define RightMotorDirPin1B  5    //Rear Right Motor direction pin 1 to
Model-Y M_A IN1 ( K1)
#define RightMotorDirPin2B 6    //Rear Right Motor direction pin 2 to
Model-Y M_A IN2 ( K1)
#define LeftMotorDirPin1B 7    //Rear Left Motor direction pin 1 to
Model-Y M_A IN3  (K3)
#define LeftMotorDirPin2B 8  //Rear Left Motor direction pin 2 to
Model-Y M_A IN4 (K3)
#define speedPinLB 12    //  Rear Wheel PWM pin connect Model-Y M_A ENB

#define LPT 2 // scan loop coumter

#define SERVO_PIN     13  //servo connect to D5

#define Echo_PIN    31 // Ultrasonic Echo pin connect to A5
#define Trig_PIN    30  // Ultrasonic Trig pin connect to A4

#define FAST_SPEED  160   //both sides of the motor speed
#define SPEED  120     //both sides of the motor speed
#define TURN_SPEED  120   //both sides of the motor speed
#define BACK_SPEED1  160   //back speed
#define BACK_SPEED2  90    //back speed

int leftscanval, centerscanval, rightscanval, ldiagonalscanval,
rdiagonalscanval;
const int distancelimit = 30; //distance limit for obstacles in front
const int sidedistancelimit = 30; //minimum distance in cm to obstacles
at both sides (the car will allow a shorter distance sideways)
int distance;
int numcycles = 0;
const int turntime = 250; //Time the robot spends turning (miliseconds)
const int backtime = 300; //Time the robot spends turning (miliseconds)

int thereis;
Servo head;
/*motor control*/
void go_Advance()  //Forward
{
FR_fwd();
FL_fwd();
RR_fwd();
RL_fwd();
}
void go_Left()  //Turn left
{
FR_fwd();
FL_bck();
```

```
RR_fwd();
RL_bck();
}
void go_Right()  //Turn right
{
FR_bck();
FL_fwd();
RR_bck();
RL_fwd();
}
void go_Back()  //Reverse
{
FR_bck();
FL_bck();
RR_bck();
RL_bck();
}



void stop_Stop()    //Stop
{
  digitalWrite(RightMotorDirPin1, LOW);
  digitalWrite(RightMotorDirPin2,LOW);
  digitalWrite(LeftMotorDirPin1,LOW);
  digitalWrite(LeftMotorDirPin2,LOW);
  digitalWrite(RightMotorDirPin1B, LOW);
  digitalWrite(RightMotorDirPin2B,LOW);
  digitalWrite(LeftMotorDirPin1B,LOW);
  digitalWrite(LeftMotorDirPin2B,LOW);
  set_Motorspeed(0,0,0,0);
}

/*set motor speed */
void set_Motorspeed(int leftFront,int rightFront,int leftBack,int
rightBack)
{
 analogWrite(speedPinL,leftFront);
 analogWrite(speedPinR,rightFront);
 analogWrite(speedPinLB,leftBack);
 analogWrite(speedPinRB,rightBack);

}

void FR_fwd()  //front-right wheel forward turn
{
  digitalWrite(RightMotorDirPin1,HIGH);
  digitalWrite(RightMotorDirPin2,LOW);
}
void FR_bck() // front-right wheel backward turn
{
```

```
  digitalWrite(RightMotorDirPin1,LOW);
  digitalWrite(RightMotorDirPin2,HIGH);
}
void FL_fwd() // front-left wheel forward turn
{
  digitalWrite(LeftMotorDirPin1,HIGH);
  digitalWrite(LeftMotorDirPin2,LOW);
}
void FL_bck() // front-left wheel backward turn
{
  digitalWrite(LeftMotorDirPin1,LOW);
  digitalWrite(LeftMotorDirPin2,HIGH);
}

void RR_fwd()  //rear-right wheel forward turn
{
  digitalWrite(RightMotorDirPin1B,HIGH);
  digitalWrite(RightMotorDirPin2B,LOW);
}
void RR_bck()  //rear-right wheel backward turn
{
  digitalWrite(RightMotorDirPin1B,LOW);
  digitalWrite(RightMotorDirPin2B,HIGH);
}
void RL_fwd()  //rear-left wheel forward turn
{
  digitalWrite(LeftMotorDirPin1B,HIGH);
  digitalWrite(LeftMotorDirPin2B,LOW);
}
void RL_bck()    //rear-left wheel backward turn
{
  digitalWrite(LeftMotorDirPin1B,LOW);
  digitalWrite(LeftMotorDirPin2B,HIGH);
}



/*detection of ultrasonic distance*/
int watch(){
  long echo_distance;
  digitalWrite(Trig_PIN,LOW);
  delayMicroseconds(5);
  digitalWrite(Trig_PIN,HIGH);
  delayMicroseconds(15);
  digitalWrite(Trig_PIN,LOW);
  echo_distance=pulseIn(Echo_PIN,HIGH);
  echo_distance=echo_distance*0.01657; //how far away is the object in
cm
 //Serial.println((int)echo_distance);
```

```
  return round(echo_distance);
}
//Meassures distances to the right, left, front, left diagonal, right
diagonal and asign them in cm to the variables rightscanval,
//leftscanval, centerscanval, ldiagonalscanval and rdiagonalscanval
(there are 5 points for distance testing)
String watchsurrounding(){
/*  obstacle_status is a binary integer, its last 5 digits stands for
if there is any obstacles in 5 directions,
 *   for example B101000 last 5 digits is 01000, which stands for Left
front has obstacle, B100111 means front, right front and right ha
 */

int obstacle_status =B100000;
  centerscanval = watch();
  if(centerscanval<distancelimit){
    stop_Stop();

    obstacle_status  =obstacle_status | B100;
    }
  head.write(120);
  delay(100);
  ldiagonalscanval = watch();
  if(ldiagonalscanval<distancelimit){
    stop_Stop();

     obstacle_status  =obstacle_status | B1000;
    }
  head.write(170); //Didn't use 180 degrees because my servo is not
able to take this angle
  delay(300);
  leftscanval = watch();
  if(leftscanval<sidedistancelimit){
    stop_Stop();

     obstacle_status  =obstacle_status | B10000;
    }

  head.write(90); //use 90 degrees if you are moving your servo through
the whole 180 degrees
  delay(100);
  centerscanval = watch();
  if(centerscanval<distancelimit){
    stop_Stop();

    obstacle_status  =obstacle_status | B100;
    }
  head.write(40);
  delay(100);
  rdiagonalscanval = watch();
  if(rdiagonalscanval<distancelimit){
```

```
    stop_Stop();

    obstacle_status  =obstacle_status | B10;
    }
  head.write(0);
  delay(100);
  rightscanval = watch();
  if(rightscanval<sidedistancelimit){
    stop_Stop();

    obstacle_status  =obstacle_status | 1;
    }
  head.write(90); //Finish looking around (look forward again)
  delay(300);
   String obstacle_str= String(obstacle_status,BIN);
  obstacle_str= obstacle_str.substring(1,6);

  return obstacle_str; //return 5-character string standing for 5
direction obstacle status
}

void auto_avoidance(){

  ++numcycles;
  if(numcycles>=LPT){ //Watch if something is around every LPT loops
while moving forward
     stop_Stop();
    String obstacle_sign=watchsurrounding(); // 5 digits of
obstacle_sign binary value means the 5 direction obstacle status
      Serial.print("begin str=");
        Serial.println(obstacle_sign);
                    if( obstacle_sign=="10000"){
     Serial.println("SLIT right");
          set_Motorspeed(FAST_SPEED,SPEED,FAST_SPEED,SPEED);
     go_Advance();

      delay(turntime);
      stop_Stop();
    }
        else    if( obstacle_sign=="00001"  ){
     Serial.println("SLIT LEFT");
       set_Motorspeed(SPEED,FAST_SPEED,SPEED,FAST_SPEED);
      go_Advance();

      delay(turntime);
      stop_Stop();
    }
    else if( obstacle_sign=="11100" || obstacle_sign=="01000" ||
obstacle_sign=="11000"  || obstacle_sign=="10100"  ||
```

```
obstacle_sign=="01100" ||obstacle_sign=="00100"
||obstacle_sign=="01000" ){
     Serial.println("hand right");
        go_Right();
       set_Motorspeed(TURN_SPEED,TURN_SPEED,TURN_SPEED,TURN_SPEED);
       delay(turntime);
       stop_Stop();
    }
    else if( obstacle_sign=="00010" || obstacle_sign=="00111" ||
obstacle_sign=="00011"  || obstacle_sign=="00101" ||
obstacle_sign=="00110" || obstacle_sign=="01010" ){
    Serial.println("hand left");
     go_Left();//Turn left
     set_Motorspeed(TURN_SPEED,TURN_SPEED,TURN_SPEED,TURN_SPEED);
      delay(turntime);
      stop_Stop();
    }

    else if(  obstacle_sign=="01111" ||  obstacle_sign=="10111" ||
obstacle_sign=="11111"  ){
    Serial.println("hand back left");
      go_Back();
        set_Motorspeed(
BACK_SPEED1,BACK_SPEED2,BACK_SPEED1,BACK_SPEED2);
       delay(backtime);
          stop_Stop();
        }
         else if( obstacle_sign=="11011"  ||    obstacle_sign=="11101"
||  obstacle_sign=="11110"  || obstacle_sign=="01110"  ){
    Serial.println("hand back right");
   go_Back();
    set_Motorspeed(BACK_SPEED2,BACK_SPEED1,BACK_SPEED2,BACK_SPEED1);
       delay(backtime);
          stop_Stop();
        }

        else Serial.println("no handle");
    numcycles=0; //Restart count of cycles
  } else {
     set_Motorspeed(SPEED,SPEED,SPEED,SPEED);
     go_Advance();  // if nothing is wrong go forward using go()
function above.
        delay(backtime);
          stop_Stop();
  }

  //else  Serial.println(numcycles);

  distance = watch(); // use the watch() function to see if anything is
ahead (when the robot is just moving forward and not looking around it
will test the distance in front)
```

```
  if (distance<distancelimit){ // The robot will just stop if it is
completely sure there's an obstacle ahead (must test 25 times) (needed
to ignore ultrasonic sensor's false signals)
 Serial.println("final go back");
    go_Back();
  set_Motorspeed(BACK_SPEED1,BACK_SPEED2,BACK_SPEED1,BACK_SPEED2);
  delay(backtime);
      ++thereis;}
  if (distance>distancelimit){
      thereis=0;} //Count is restarted
  if (thereis > 25){
  Serial.println("final stop");
    stop_Stop(); // Since something is ahead, stop moving.
    thereis=0;
  }
}

void setup() {
  /*setup L298N pin mode*/
  pinMode(RightMotorDirPin1, OUTPUT);
  pinMode(RightMotorDirPin2, OUTPUT);
  pinMode(speedPinL, OUTPUT);

  pinMode(LeftMotorDirPin1, OUTPUT);
  pinMode(LeftMotorDirPin2, OUTPUT);
  pinMode(speedPinR, OUTPUT);
  pinMode(RightMotorDirPin1B, OUTPUT);
  pinMode(RightMotorDirPin2B, OUTPUT);
  pinMode(speedPinLB, OUTPUT);

  pinMode(LeftMotorDirPin1B, OUTPUT);
  pinMode(LeftMotorDirPin2B, OUTPUT);
  pinMode(speedPinRB, OUTPUT);
  stop_Stop();//stop move
  /*init HC-SR04*/
  pinMode(Trig_PIN, OUTPUT);
  pinMode(Echo_PIN,INPUT);
  /*init buzzer*/

  digitalWrite(Trig_PIN,LOW);
  /*init servo*/
  head.attach(SERVO_PIN);
    head.write(0);
   delay(1000);
     head.write(170);
   delay(1000);
  head.write(90);
   delay(8000);
```

```
  Serial.begin(9600);



  stop_Stop();//Stop

}

void loop() {
auto_avoidance();
// Serial.println( watchsurrounding());
}
```

Achat robot Mecanum

Achat Aliexpres ~ 50€

Achat RobotShop ~420€..

Achat KUBII ~ 130€

Liens Projets Arduino Robot Mecanum

Projet 1 Robot Mecanum EN

Projet 2 Robot Mecanum EN

Projet 3 Robot Mecanum EN

Programme Robot Mecanum

Robot Mecanum 001

robot_Original_Mecanum.ino

```
// Tester par GL; Fonctionne mais ne permet pas tout les Mouvements
//2020.11.22
//Configure THE PWM control pin
const int PWM2A = 11;      //M1 motor
const int PWM2B = 3;       //M2 motor
const int PWM0A = 6;       //M3 motor
const int PWM0B = 5;       //M4 motor

const int DIR_CLK = 4;     // Data input clock line
const int DIR_EN = 7;      //Equip the L293D enabling pins
const int DATA = 8;        // USB cable
const int DIR_LATCH = 12;  // Output memory latch clock
//Define the pin of ultrasonic obstacle avoidance sensor
const int Trig = A2;       //A2 is defined as the pin Trig connected to
the ultrasonic sensor
const int Echo = A3;       //A3 is defined as the pin Echo connected to
the ultrasonic sensor
//Define motion state
const int Forward = 39;    //39存放到Forward
const int Back = 216;      //216存放到Back
const int Left = 57;       //57存放到Left变量中
const int Right = 198;     //The right amount of change
const int Stop = 0;        //Parking variable
//Set the default speed between 1 and 255
int Speed1 = 180; //PWM0B -M3
int Speed2 = 180; //PWM0A -M4
int Speed3 = 180; //PWM2A -M3
int Speed4 = 180; //PWM2B -M4

int distance = 0;           //Variables for storing ultrasonic sensor
measurements
char serialData;            //把串口接收的数据存放serialData中
char cmd;                   //Store bluetooth receive commands

void setup()
{
    Serial.begin(9600);//Set the serial port baud rate 9600
```

```
    //Configure as output mode
    pinMode(DIR_CLK,OUTPUT);
    pinMode(DATA,OUTPUT);
    pinMode(DIR_EN,OUTPUT);
    pinMode(DIR_LATCH,OUTPUT);
    pinMode(PWM0B,OUTPUT);
    pinMode(PWM0A,OUTPUT);
    pinMode(PWM2A,OUTPUT);
    pinMode(PWM2B,OUTPUT);

    pinMode(Trig,OUTPUT);//The Trig pin connected to the ultrasound is
set to output mode
    pinMode(Echo,INPUT);//The Echo pin connected to the ultrasound is
set to input mode

    void Motor(int Dri,int Speed1,int Speed2,int Speed3,int Speed4);
    int SR04(int Trig,int Echo);
    void AvoidingObstacles();
    void HC05();
}

void loop()
{
    distance = SR04(Trig,Echo);       //Acquisition of ultrasonic
distance

    HC05(); //Call the Bluetooth car control function
}


/* Function name: Motor();
* Function: Change the movement direction and speed of the car through
the entrance parameters
* Entry parameter 1: Dri car movement direction
* Entry parameters 2~3: Speed1~Speed4 motor speed, value range 0~255
* Dri value description (forward :39; Back: 216; Left translation: 57;
Right translation: 198; Stop: 0;
* Right rotation: 106; Left rotation: 149)
* Return value: None
 */
void Motor(int Dir,int Speed1,int Speed2,int Speed3,int Speed4)
{
    analogWrite(PWM2A,Speed1); //Motor PWM speed regulation
    analogWrite(PWM2B,Speed2); //Motor PWM speed regulation
    analogWrite(PWM0A,Speed3); //Motor PWM speed regulation
    analogWrite(PWM0B,Speed4); //Motor PWM speed regulation

    digitalWrite(DIR_LATCH,LOW); //DIR_LATCH sets the low level and
writes the direction of motion in preparation
```

```
    shiftOut(DATA,DIR_CLK,MSBFIRST,Dir);//Write Dir motion direction
value

    digitalWrite(DIR_LATCH,HIGH);//DIR_LATCH sets the high level and
outputs the direction of motion
}

/*
Function name: SR04()
Function: Obtain ultrasonic ranging data
Entry parameters: Trig, Echo
Function return value: cm
*/
int SR04(int Trig,int Echo)
{
    float cm = 0;

    digitalWrite(Trig,LOW);     //Trig is set to low level
    delayMicroseconds(2);       //Wait 2 microseconds
    digitalWrite(Trig,HIGH);    //Trig is set to high level
    delayMicroseconds(15);      //Wait 15 microseconds
    digitalWrite(Trig,LOW);     //Trig is set to low

    cm = pulseIn(Echo,HIGH)/58.8; //Convert the ranging time to CM
    cm = (int(cm * 100.0))/100.0; //Leave 2 as a decimal
    Serial.print("Distance:");    //Character Distance displayed in
serial port monitor window:
    Serial.print(cm);
    Serial.println("cm");

    return cm;      //Returns cm value ranging data
}
/*
* Function name: HC05()
* Function: Receive Bluetooth data, control the car movement direction
* Entry parameters: None
* Return value: None
*/
void HC05()
{
    if(Serial.available() > 0)      //Determine if the received data is
greater than 0
    {
        serialData = Serial.read(); //Receiving function

        if     ('F' == serialData )  cmd = 'F';     //If the data
received by the serial port is character F, save F to CMD
        else if('B' == serialData )  cmd = 'B';     //If the data
received by the serial port is character B, save F to CMD
```

```
        else if('L' == serialData )  cmd = 'L';     //If the serial
port receives data as the character L, save F to CMD
        else if('R' == serialData )  cmd = 'R';     //If the serial
port receives data as the character R, save F to CMD
        else if('S' == serialData )  cmd = 'S';     //If the serial
port receives data as character S, save F to CMD

        else if( serialData == '+' && Speed1 < 245)//If you receive a
string plus, the speed increases
        {
            Speed1 += 10;   //We're going to increase the velocity by
10 at a time
            Speed2 = Speed1;
            Speed3 = Speed1;
            Speed4 = Speed1;
        }
        else if( serialData == '-' && Speed1 > 30)//When I receive a
string -- the speed decreases
        {
            Speed1 -= 10;   //I'm going to subtract 10 at a time
            Speed2 = Speed1;
            Speed3 = Speed1;
            Speed4 = Speed1;
        }

         else if('A' == serialData) //Bluetooth received the string R,
car right translation
        {
            Motor(106,Speed1,Speed2,Speed3,Speed4);      //
            delay(100);
        }
        else if('C' == serialData) //Bluetooth received the string R,
car right translation
        {
            Motor(149,Speed1,Speed2,Speed3,Speed4);      //
            delay(100);
        }
    }

    if('F' == cmd)   //If Bluetooth receives the string F, the dolly
moves forward and enables obstacle avoidance
    {
       AvoidingObstacles();//The ultrasonic obstacle avoidance function
is called to realize the obstacle avoidance function
    }
    else if('B' == cmd)     //Bluetooth receives string B, car backs up
    {
        Motor(Back,Speed1,Speed2,Speed3,Speed4);
    }
    else if('L' == cmd)    //Bluetooth received the string L, car left
translation
```

```
    {
        Motor(Left,Speed1,Speed2,Speed3,Speed4);
    }
    else if('R' == cmd)     //Bluetooth received the string R, car
right translation
    {
        Motor(Right,Speed1,Speed2,Speed3,Speed4);      //right
translation
    }

    else if('S' == serialData)      //When the string S is received,
the cart stops moving
    {
        Motor(Stop,0,0,0,0);
    }
}

void AvoidingObstacles()
{
    if((distance > 20 ) || cmd == 'F')//If the distance is greater than
20cm or bluetooth receives a command equal to F
    {
        delay(100);//Delay of 100 ms
        if(distance > 20)//Again, determine if the distance is really
greater than 20cm
        {
            Motor(Forward,Speed1,Speed2,Speed3,Speed4); //Call forward
function
        }
        else //Otherwise the distance is less than 20
        {
            Motor(Back,Speed1,Speed2,Speed3,Speed4);//retreat
            delay(500);
            Motor(106,Speed1,Speed2,Speed3,Speed4);//Turn left to
change the direction of the car
            delay(500);
        }
    }
}
```

Robot Mecanum Version 3 modifié par GL

Robot_Mecanum003.ino

```
// Tester par GL le 19 Aout 2022
//Avec compensation vitesse moteur gauche ( M1 et M2 )
```

```
#include <MotorDriver.h>

//Define the pin of ultrasonic obstacle avoidance sensor
const int Trig = A2;       //A2 is defined as the pin Trig connected to
the ultrasonic sensor
const int Echo = A3;       //A3 is defined as the pin Echo connected to
the ultrasonic sensor

MotorDriver m;


int vit=180;
int comp = vit /3; // Compensation vitesse pour les moteurs M1 et M2
int vitcomp= (vit - comp);
;
int tempo = 1000;
int tempo10 = 1000;
int dure= 2;

int distance = 0;           //Variables for storing ultrasonic sensor
measurements
char serialData;
char cmd;

void arret()
{
  m.motor(1,RELEASE,0);
  m.motor(2,RELEASE,0);
  m.motor(3,RELEASE,0);
  m.motor(4,RELEASE,0);
}

void avant()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}

void arriere()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
  m.motor(4,BACKWARD,vit);
}

void LatDroite()
{
```

```
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
  m.motor(3,FORWARD,vit);
  m.motor(4,BACKWARD,vit);
}

void LatGauche()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
  m.motor(4,FORWARD,vit);
}
void BiaisDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,RELEASE,0);
  m.motor(3,FORWARD,vit);
  m.motor(4,RELEASE,0);
}


void BiaisGauche()
{
  m.motor(1,RELEASE,0);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,RELEASE,0);
  m.motor(4,FORWARD,vit);

}

void TourneDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,RELEASE,0);
  m.motor(4,RELEASE,0);
}

void TourneGauche()
{
  m.motor(1,RELEASE,0);
  m.motor(2,RELEASE,0);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}


void DemiTourDroite()
```

```
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
  m.motor(4,BACKWARD,vit);
}

void DemiTourGauche()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}

int SR04(int Trig,int Echo)
{
    float cm = 0;

    digitalWrite(Trig,LOW);     //Trig is set to low level
    delayMicroseconds(2);       //Wait 2 microseconds
    digitalWrite(Trig,HIGH);    //Trig is set to high level
    delayMicroseconds(15);      //Wait 15 microseconds
    digitalWrite(Trig,LOW);     //Trig is set to low

    cm = pulseIn(Echo,HIGH)/58.8; //Convert the ranging time to CM
    cm = (int(cm * 100.0))/100.0; //Leave 2 as a decimal
    //Serial.print("Distance:");    //Character Distance displayed in
serial port monitor window:
    //Serial.print(cm);
    //Serial.println("cm");

    return cm;      //Returns cm value ranging data
}

void AvoidingObstacles()
{
    if((distance > 20 ) || cmd == 'F')//If the distance is greater than
20cm or bluetooth receives a command equal to F
    {
        delay(100);//Delay of 100 ms
        if(distance > 20)//Again, determine if the distance is really
greater than 20cm
        {
           avant();
        }
        else //Otherwise the distance is less than 20
        {
            arriere();
            delay(500);
            TourneGauche();
```

```
            delay(500);
        }
    }
}


void HC05()
{
    if(Serial.available() == -1) {
       arret();
       cmd='S';
          }
    if(Serial.available() > 0)      //Determine if the received data is
greater than 0
    {
        serialData = Serial.read(); //Receiving function

        if     ('F' == serialData )  cmd = 'F';    //Avant
        else if('B' == serialData )  cmd = 'B';    //Arriere
        else if('L' == serialData )  cmd = 'L';    //Tourne Gauche
        else if('R' == serialData )  cmd = 'R';    //Tourne Droite
        else if('S' == serialData )  cmd = 'S';    //Stop
        else if('A' == serialData )  cmd = 'A';    //Lateral Droite
        else if('C' == serialData )  cmd = 'C';    //Lateral Gauche
        else if('D' == serialData )  cmd = 'D';    //Demi Tour
        else if('J' == serialData )  cmd = 'J';    //Biais Droite
        else if('K' == serialData )  cmd = 'K';    //Biais Gauche



        else if( serialData == '+' && vit < 245)//If you receive a
string plus, the speed increases
        {
            comp += 5;   //We're going to increase the velocity by 10
at a time

        }
        else if( serialData == '-' && vit > 30)//When I receive a
string -- the speed decreases
        {
           comp -= 5;   //I'm going to subtract 10 at a time
        }

    }

    if('F' == cmd)   //If Bluetooth receives the string F, the dolly
moves forward and enables obstacle avoidance
    {
        avant();
```

```
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();









      //AvoidingObstacles();//The ultrasonic obstacle avoidance
function is called to realize the obstacle avoidance function
    }
    else if('A' == serialData)
    {

         LatDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('J' == serialData)
    {

         BiaisDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('K' == serialData)
    {

         BiaisGauche();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('C' == serialData)
    {



         LatGauche();
         delay(tempo10);
         cmd ='S';
```

```
          serialData = 'S';
          arret();

    }
    else if('B' == cmd)     //Bluetooth receives string B, car backs up
    {
        arriere();
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }
    else if('L' == cmd)    //Bluetooth received the string L, car left
translation
    {
        TourneGauche();
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }

    else if('R' == cmd)     //Bluetooth received the string R, car
right translation
    {
        TourneDroite();     //right translation
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }

    else if('S' == serialData)      //When the string S is received,
the cart stops moving
    {
        arret();
    }


    else if('D' == serialData)
    {
         DemiTourDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();
```

```
    }
}
void setup()
{
 Serial.begin(9600);

 pinMode(Trig,OUTPUT);//The Trig pin connected to the ultrasound is set
to output mode
pinMode(Echo,INPUT);//The Echo pin connected to the ultrasound is set
to input mode

void Motor(int Dri,int Speed1,int Speed2,int Speed3,int Speed4);
int SR04(int Trig,int Echo);
void AvoidingObstacles();
void HC05();

}

void loop()
{
    //distance = SR04(Trig,Echo);       //Acquisition of ultrasonic
distance

    HC05(); //Call the Bluetooth car control function
}
```

Robot Mecanum Version 4 modifié par GL

Robot_Mecanum004.ino

```
// Tester par GL le 22 Aout 2022
// Avec touche "G" symbole "0" pour le programme de demonstration
//Avec compensation vitesse moteur gauche ( M1 et M2 )
/*
                              ^
                    M1------|------M4
                            |
                            |
                            |
                            |
                            |
                    M2------|------M3

*/
#include <MotorDriver.h>

//Define the pin of ultrasonic obstacle avoidance sensor
```

```
const int Trig = A2;       //A2 is defined as the pin Trig connected to
the ultrasonic sensor
const int Echo = A3;       //A3 is defined as the pin Echo connected to
the ultrasonic sensor

MotorDriver m;


int vit=180;
int comp = vit /3;
int vitcomp= (vit - comp);
;
int tempo = 1000;
int tempo10 = 1000;
int dure= 2;

int distance = 0;           //Variables for storing ultrasonic sensor
measurements
char serialData;
char cmd;

void arret()
{
  m.motor(1,RELEASE,0);
  m.motor(2,RELEASE,0);
  m.motor(3,RELEASE,0);
  m.motor(4,RELEASE,0);
}

void avant()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}

void arriere()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
  m.motor(4,BACKWARD,vit);
}

void LatDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
```

```
  m.motor(3,FORWARD,vit);
  m.motor(4,BACKWARD,vit);
}

void LatGauche()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
  m.motor(4,FORWARD,vit);
}
void BiaisDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,RELEASE,0);
  m.motor(3,FORWARD,vit);
  m.motor(4,RELEASE,0);
}


void BiaisGauche()
{
  m.motor(1,RELEASE,0);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,RELEASE,0);
  m.motor(4,FORWARD,vit);

}

void TourneDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,RELEASE,0);
  m.motor(4,RELEASE,0);
}

void TourneGauche()
{
  m.motor(1,RELEASE,0);
  m.motor(2,RELEASE,0);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}


void DemiTourDroite()
{
  m.motor(1,FORWARD,vitcomp);
  m.motor(2,FORWARD,vitcomp);
  m.motor(3,BACKWARD,vit);
```

```
  m.motor(4,BACKWARD,vit);
}

void DemiTourGauche()
{
  m.motor(1,BACKWARD,vitcomp);
  m.motor(2,BACKWARD,vitcomp);
  m.motor(3,FORWARD,vit);
  m.motor(4,FORWARD,vit);
}



void Demo()
{
for (int i=0; i < 3 ; i++){
  avant();
  delay(tempo10);
  arriere();
  delay(tempo10);
  LatDroite();
  delay(tempo10);
  LatGauche();
  delay(tempo10);
  BiaisDroite();
  delay(tempo10);
  BiaisGauche();
  delay(tempo10);
  TourneDroite();
  delay(tempo10);
  TourneGauche();
  delay(tempo10);
  DemiTourDroite();
  delay(tempo10);
  DemiTourGauche();
  delay(tempo10);
  }
}
int SR04(int Trig,int Echo)
{
    float cm = 0;

    digitalWrite(Trig,LOW);     //Trig is set to low level
    delayMicroseconds(2);       //Wait 2 microseconds
    digitalWrite(Trig,HIGH);    //Trig is set to high level
    delayMicroseconds(15);      //Wait 15 microseconds
    digitalWrite(Trig,LOW);     //Trig is set to low

    cm = pulseIn(Echo,HIGH)/58.8; //Convert the ranging time to CM
```

```
    cm = (int(cm * 100.0))/100.0; //Leave 2 as a decimal
    //Serial.print("Distance:");    //Character Distance displayed in
serial port monitor window:
    //Serial.print(cm);
    //Serial.println("cm");

    return cm;      //Returns cm value ranging data
}

void AvoidingObstacles()
{
    if((distance > 20 ) || cmd == 'F')//If the distance is greater than
20cm or bluetooth receives a command equal to F
    {
        delay(100);//Delay of 100 ms
        if(distance > 20)//Again, determine if the distance is really
greater than 20cm
        {
           avant();
        }
        else //Otherwise the distance is less than 20
        {
            arriere();
            delay(500);
            TourneGauche();
            delay(500);
        }
    }
}


void HC05()
{
    if(Serial.available() == -1) {
       arret();
       cmd='S';
          }
    if(Serial.available() > 0)      //Determine if the received data is
greater than 0
    {
        serialData = Serial.read(); //Receiving function

        if     ('F' == serialData )  cmd = 'F';    //Avant
        else if('B' == serialData )  cmd = 'B';    //Arriere
        else if('L' == serialData )  cmd = 'L';    //Tourne Gauche
        else if('R' == serialData )  cmd = 'R';    //Tourne Droite
        else if('S' == serialData )  cmd = 'S';    //Stop
        else if('A' == serialData )  cmd = 'A';    //Lateral Droite
        else if('C' == serialData )  cmd = 'C';    //Lateral Gauche
        else if('D' == serialData )  cmd = 'D';    //Demi Tour
        else if('J' == serialData )  cmd = 'J';    //Biais Droite
```

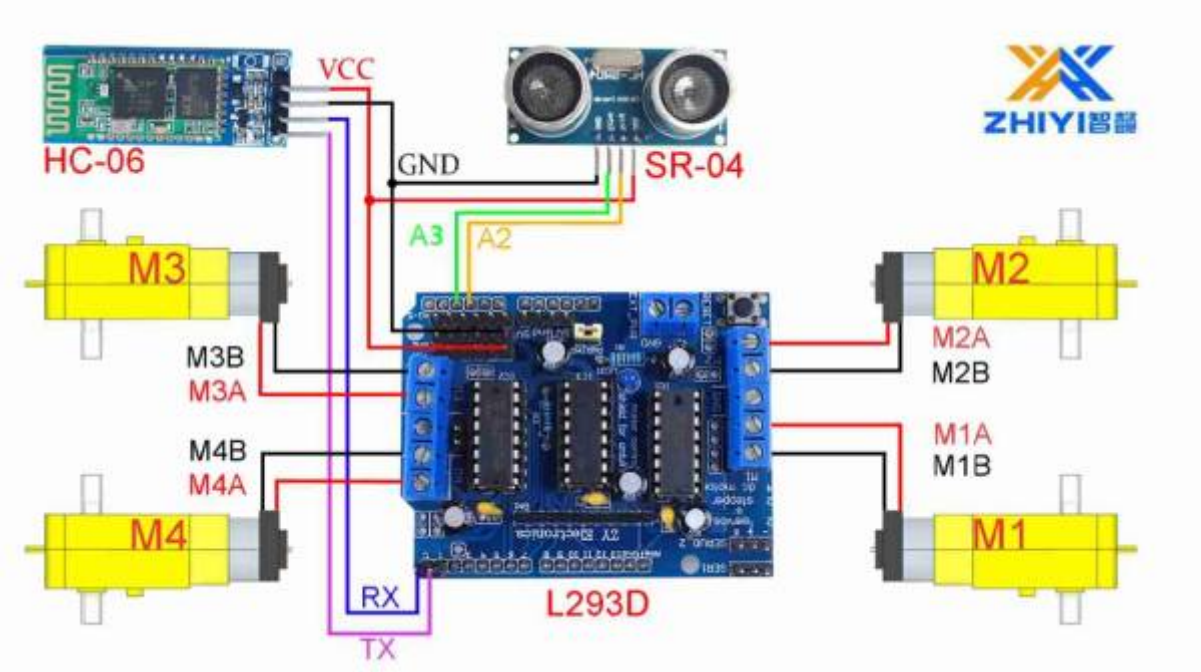
```
        else if('K' == serialData )  cmd = 'K';    //Biais Gauche
        else if('G' == serialData )  cmd = 'G';    //Demo



        else if( serialData == '+' && vit < 245)//If you receive a
string plus, the speed increases
        {
            comp += 5;   //We're going to increase the velocity by 10
at a time

        }
        else if( serialData == '-' && vit > 30)//When I receive a
string -- the speed decreases
        {
           comp -= 5;   //I'm going to subtract 10 at a time
        }

    }

    if('F' == cmd)   //If Bluetooth receives the string F, the dolly
moves forward and enables obstacle avoidance
    {
        avant();
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();









      //AvoidingObstacles();//The ultrasonic obstacle avoidance
function is called to realize the obstacle avoidance function
    }
    else if('A' == serialData)
    {

         LatDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('J' == serialData)
```

```
    {

         BiaisDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('K' == serialData)
    {

         BiaisGauche();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('C' == serialData)
    {


         LatGauche();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('B' == cmd)     //Bluetooth receives string B, car backs up
    {
        arriere();
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }
    else if('L' == cmd)    //Bluetooth received the string L, car left
translation
    {
        TourneGauche();
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }

    else if('R' == cmd)     //Bluetooth received the string R, car
```

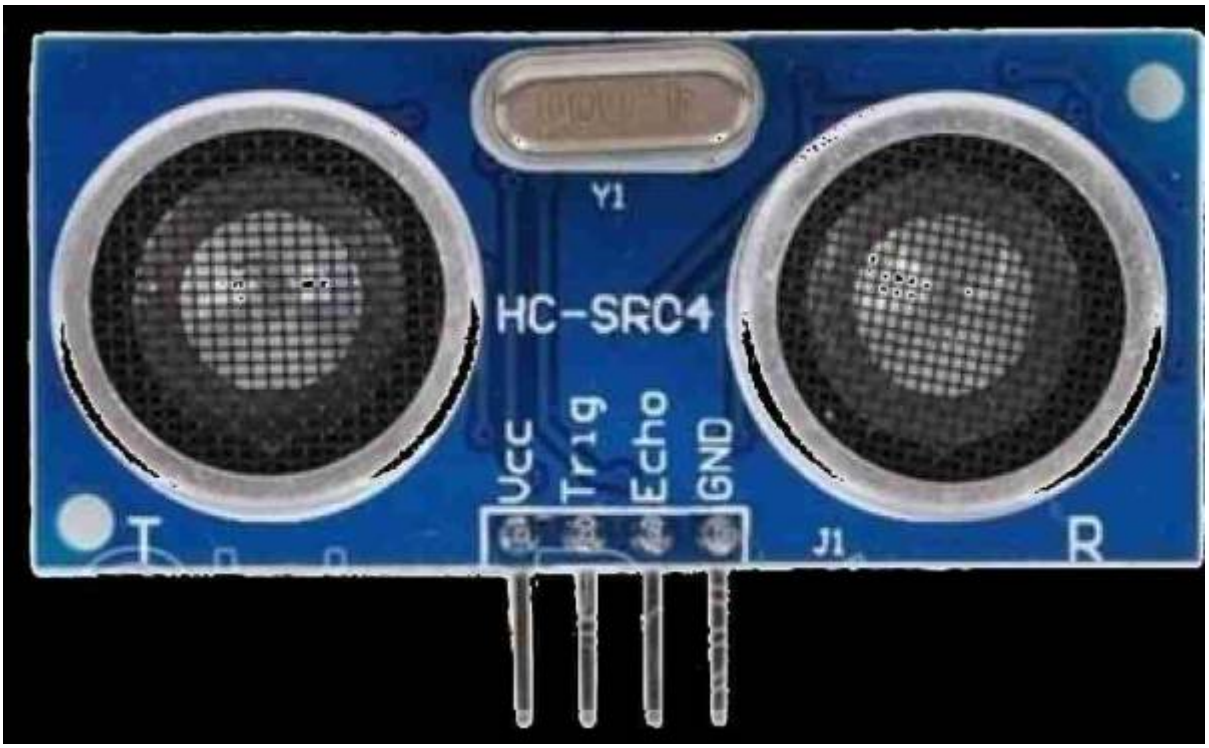
```
right translation
    {
        TourneDroite();     //right translation
        delay(tempo10);
        cmd ='S';
        serialData = 'S';
        arret();

    }

    else if('S' == serialData)      //When the string S is received,
the cart stops moving
    {
        arret();
    }


    else if('D' == serialData)
    {
         DemiTourDroite();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
    else if('G' == serialData)
    {
         Demo();
         delay(tempo10);
         cmd ='S';
         serialData = 'S';
         arret();

    }
}
void setup()
{
 Serial.begin(9600);

 pinMode(Trig,OUTPUT);//The Trig pin connected to the ultrasound is set
to output mode
pinMode(Echo,INPUT);//The Echo pin connected to the ultrasound is set
to input mode

void Motor(int Dri,int Speed1,int Speed2,int Speed3,int Speed4);
int SR04(int Trig,int Echo);
void AvoidingObstacles();
void HC05();
```

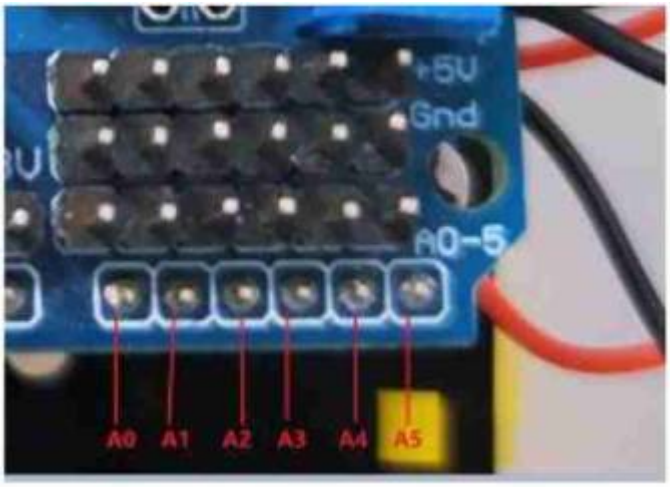
```
}

void loop()
{
    //distance = SR04(Trig,Echo);       //Acquisition of ultrasonic
distance

    HC05(); //Call the Bluetooth car control function
}
```

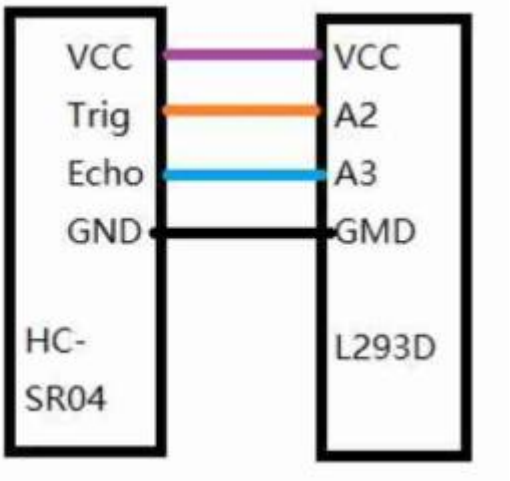
Cablage Robot Mecanum



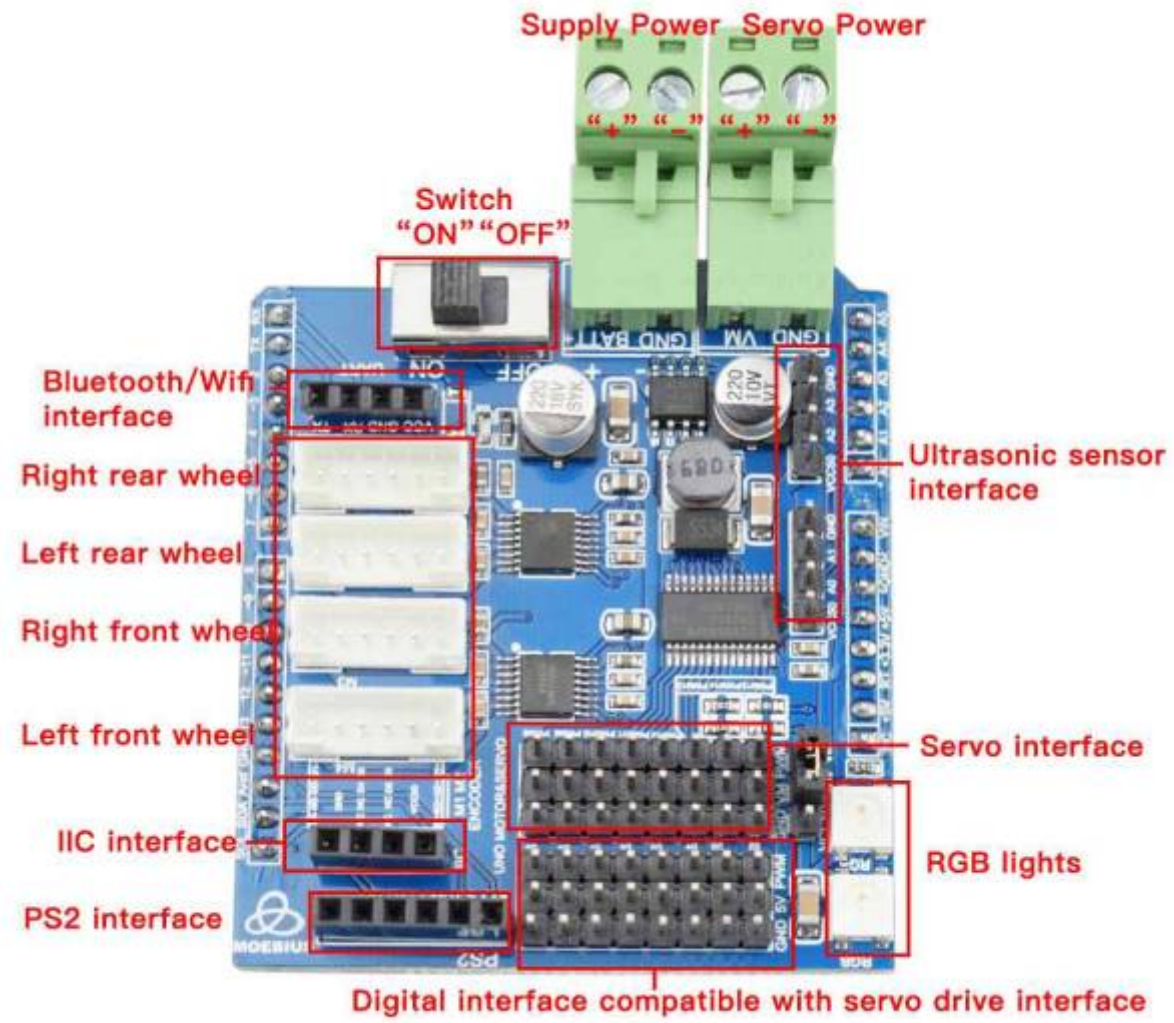
Cablage SR04







Robot Mecanum commandé via PS2 sans fil



programme1 test mecanum PS2

mecanum-test001.ino

```
#include <PS2X_lib.h>  //for MOEBIUS
#include "FaBoPWM_PCA9685.h"

//#include "servo.hpp"

FaBoPWM faboPWM;
int pos = 0;
int MAX_VALUE = 2000;    // 电机速度限制 motor Motor_PWM
int MIN_VALUE = 300;

//PS2手柄引脚□PS2 handle Pin
#define PS2_DAT        13
#define PS2_CMD        11
#define PS2_SEL        10
```

```
#define PS2_CLK        12

//MOTOR CONTROL Pin
#define DIRA1 0
#define DIRA2 1
#define DIRB1 2
#define DIRB2 3
#define DIRC1 4
#define DIRC2 5
#define DIRD1 6
#define DIRD2 7

char speed;
// #define pressures   true
#define pressures   false
// #define rumble      true
#define rumble      false

PS2X ps2x; // create PS2 Controller Class

//right now, the library does NOT support hot pluggable controllers,
meaning
//you must always either restart your Arduino after you connect the
controller,
//or call config_gamepad(pins) again after connecting the controller.

int error = 0;
byte type = 0;
byte vibrate = 0;

void (* resetFunc) (void) = 0;

//电机控制，前进、后退、停止   motor control advance\back\stop
#define MOTORA_FORWARD(pwm)
do{faboPWM.set_channel_value(DIRA1,pwm);faboPWM.set_channel_value(DIRA2
, 0);}while(0)
#define MOTORA_STOP(x)
do{faboPWM.set_channel_value(DIRA1,0);faboPWM.set_channel_value(DIRA2,
0);}while(0)
#define MOTORA_BACKOFF(pwm)
do{faboPWM.set_channel_value(DIRA1,0);faboPWM.set_channel_value(DIRA2,
pwm);}while(0)

#define MOTORB_FORWARD(pwm)
do{faboPWM.set_channel_value(DIRB1,pwm);faboPWM.set_channel_value(DIRB2
, 0);}while(0)
#define MOTORB_STOP(x)
do{faboPWM.set_channel_value(DIRB1,0);faboPWM.set_channel_value(DIRB2,
0);}while(0)
```

```
#define MOTORB_BACKOFF(pwm)
do{faboPWM.set_channel_value(DIRB1,0);faboPWM.set_channel_value(DIRB2,
pwm);}while(0)

#define MOTORC_FORWARD(pwm)
do{faboPWM.set_channel_value(DIRC1,pwm);faboPWM.set_channel_value(DIRC2
, 0);}while(0)
#define MOTORC_STOP(x)
do{faboPWM.set_channel_value(DIRC1,0);faboPWM.set_channel_value(DIRC2,
0);}while(0)
#define MOTORC_BACKOFF(pwm)
do{faboPWM.set_channel_value(DIRC1,0);faboPWM.set_channel_value(DIRC2,
pwm);}while(0)

#define MOTORD_FORWARD(pwm)
do{faboPWM.set_channel_value(DIRD1,pwm);faboPWM.set_channel_value(DIRD2
, 0);}while(0)
#define MOTORD_STOP(x)
do{faboPWM.set_channel_value(DIRD1,0);faboPWM.set_channel_value(DIRD2,
0);}while(0)
#define MOTORD_BACKOFF(pwm)
do{faboPWM.set_channel_value(DIRD1,0);faboPWM.set_channel_value(DIRD2,
pwm);}while(0)

#define SERIAL  Serial

//#define SERIAL  Serial3

#define LOG_DEBUG

#ifdef LOG_DEBUG
#define M_LOG SERIAL.print
#else
#define M_LOG
#endif

//PWM参数
#define MAX_PWM   2000
#define MIN_PWM   300

int Motor_PWM = 1900;
int Motor_PWM_mod = Motor_PWM - 400;

//------AVANCE-----------------------------------------------------------
-----------
//    ↑A-----B↑
//     |  ↑  |
//     |  |  |
//    ↑C-----D↑
void ADVANCE(uint8_t pwm_A,uint8_t pwm_B,uint8_t pwm_C,uint8_t pwm_D)
{
```

```
  MOTORA_FORWARD(Motor_PWM_mod);
  MOTORB_FORWARD(Motor_PWM);
  MOTORC_FORWARD(Motor_PWM_mod);
  MOTORD_FORWARD(Motor_PWM);
}
//-------RECUL-----------------------------------------------------------
---------------
//     ↓A-----B↓
//      |  |  |
//      |  ↓  |
//     ↓C-----D↓
void BACK()
{
  MOTORA_BACKOFF(Motor_PWM_mod);MOTORB_BACKOFF(Motor_PWM);
  MOTORC_BACKOFF(Motor_PWM_mod);MOTORD_BACKOFF(Motor_PWM);
}
//-------GAUCHE AVANT----------------------------------------------------
----------------------
//     =B-----A↑
//      |   ↖ |
//      | ↖   |
//     ↑C-----D=
void LEFT_1()
{
  MOTORA_FORWARD(Motor_PWM);MOTORB_STOP(Motor_PWM);
  MOTORC_FORWARD(Motor_PWM);MOTORD_STOP(Motor_PWM);
}
//-------GAUCHE----------------------------------------------------------
-------------------
//
//
//     ↓A-----B↑
//      |  ←  |
//      |  ←  |
//     ↑C-----D↓
void LEFT_2()
{
  MOTORA_BACKOFF(Motor_PWM);MOTORB_FORWARD(Motor_PWM);
  MOTORC_FORWARD(Motor_PWM);MOTORD_BACKOFF(Motor_PWM);
}
//---------GAUCHE ARRIERE--------------------------------------------------
-----------------------------
//
//
//     ↓A-----B=
//      | ↙   |
//      |   ↙ |
//     =C-----D↓
void LEFT_3()
```

```
{
  MOTORA_BACKOFF(Motor_PWM);MOTORB_STOP(Motor_PWM);
  MOTORC_STOP(Motor_PWM);MOTORD_BACKOFF(Motor_PWM);
}
//-----------DROITE AVANT-----------------------------------------------
-------------------------
//     ↑A-----B=
//      | ↗   |
//      |   ↗ |
//     =C-----D↑
void RIGHT_1()
{
  MOTORA_FORWARD(Motor_PWM);MOTORB_STOP(Motor_PWM);
  MOTORC_STOP(Motor_PWM);MOTORD_FORWARD(Motor_PWM);
}
//--------------DROITE-----------------------------------------------------
----------------------
//     ↑A-----B↓
//      |  →  |
//      |  →  |
//     ↓C-----D↑
void RIGHT_2()
{
  MOTORA_FORWARD(Motor_PWM);MOTORB_BACKOFF(Motor_PWM);
  MOTORC_BACKOFF(Motor_PWM);MOTORD_FORWARD(Motor_PWM);
}
//-----------------DROITE ARRIERE---------------------------------------
-------------------------
//     =A-----B↓
//      |   ↘ |
//      | ↘   |
//     ↓C-----D=
void RIGHT_3()
{
  MOTORA_STOP(Motor_PWM);MOTORB_BACKOFF(Motor_PWM);
  MOTORC_BACKOFF(Motor_PWM);MOTORD_STOP(Motor_PWM);
}
//--------------------TOURNE---DROITE--------------------------------
----------------------
//     ↑A-----B↓
//      | ↗ ↘ |
//      | ↖ ↙ |
//     ↑C-----D↓
void rotate_1()  //tate_1(uint8_t pwm_A,uint8_t pwm_B,uint8_t
pwm_C,uint8_t pwm_D)
{
  MOTORA_FORWARD(Motor_PWM);MOTORB_BACKOFF(Motor_PWM);
  MOTORC_FORWARD(Motor_PWM);MOTORD_BACKOFF(Motor_PWM);
}
//---------------------TOURNE GAUCHE----------------------------------
-------------------------
```

```
//    ↓A-----B↑
//      | ↙ ↖ |
//      | ↘ ↗ |
//    ↓C-----D↑
void rotate_2()  // rotate_2(uint8_t pwm_A,uint8_t pwm_B,uint8_t
pwm_C,uint8_t pwm_D)
{
  MOTORA_BACKOFF(Motor_PWM);MOTORB_FORWARD(Motor_PWM);
  MOTORC_BACKOFF(Motor_PWM);MOTORD_FORWARD(Motor_PWM);
}
//---------------------------ARRET-------------------------------------
------------------
//    =A-----B=
//      |  =  |
//      |  =  |
//    =C-----D=
void STOP()
{
  MOTORA_STOP(Motor_PWM);MOTORB_STOP(Motor_PWM);
  MOTORC_STOP(Motor_PWM);MOTORD_STOP(Motor_PWM);
}
//---------------------------------------------------------------------
--------------

void UART_Control()
{
  char Uart_Date=0;
  if(SERIAL.available())
  {
    Uart_Date = SERIAL.read();
  }
  switch(Uart_Date)
  {
     case 'A':  ADVANCE(500,500,500,500);  M_LOG("Run!\r\n");
break;
     case 'B':  RIGHT_1();  M_LOG("Right up!\r\n");     break;
     case 'C':  rotate_2();                            break;
     case 'D':  RIGHT_3();  M_LOG("Right down!\r\n");   break;
     case 'E':  BACK();     M_LOG("Run!\r\n");          break;
     case 'F':  LEFT_3();   M_LOG("Left down!\r\n");    break;
     case 'G':  rotate_1();                             break;
     case 'H':  LEFT_1();   M_LOG("Left up!\r\n");     break;
     case 'Z':  STOP();     M_LOG("Stop!\r\n");        break;
     case 'z':  STOP();     M_LOG("Stop!\r\n");        break;
     case 'd':  LEFT_2();   M_LOG("Left!\r\n");        break;
     case 'b':  RIGHT_2();  M_LOG("Right!\r\n");        break;
     case 'L':  Motor_PWM = 1500;                      break;
     case 'M':  Motor_PWM = 500;                       break;
   }
```

```
}

void IO_init()
{
  STOP();
}

void setup()
{
   IO_init();
   SERIAL.begin(9600);
  if(faboPWM.begin())
  {
    Serial.println("Find PCA9685");
    faboPWM.init(300);
  }
  faboPWM.set_hz(50);
  SERIAL.print("Start");

   delay(300) ; //added delay to give wireless ps2 module some time to
startup, before configuring it
  //CHANGES for v1.6 HERE!!! **************PAY ATTENTION*************

  //setup pins and settings: GamePad(clock, command, attention, data,
Pressures?, Rumble?) check for error
  error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT,
pressures, rumble);

  if (error == 0) {
    Serial.print("Found Controller, configured successful ");
    Serial.print("pressures = ");
    if (pressures)
      Serial.println("true ");
    else
      Serial.println("false");
    Serial.print("rumble = ");
    if (rumble)
      Serial.println("true)");
    else
      Serial.println("false");
    Serial.println("Try out all the buttons, X will vibrate the
controller, faster as you press harder;");
    Serial.println("holding L1 or R1 will print out the analog stick
values.");
    Serial.println("Note: Go to www.billporter.info for updates and to
report bugs.");
  }
  else if (error == 1)
  {
    Serial.println("No controller found, check wiring, see readme.txt
to enable debug. visit www.billporter.info for troubleshooting tips");
```

```
    resetFunc();

  }

  else if (error == 2)
    Serial.println("Controller found but not accepting commands. see
readme.txt to enable debug. Visit www.billporter.info for
troubleshooting tips");

  else if (error == 3)
    Serial.println("Controller refusing to enter Pressures mode, may
not support it. ");

//  Serial.print(ps2x.Analog(1), HEX);

  type = ps2x.readType();
  switch (type) {
  case 0:
    Serial.print("Unknown Controller type found ");
    break;
  case 1:
    Serial.print("DualShock Controller found ");
    break;
  case 2:
    Serial.print("GuitarHero Controller found ");
    break;
  case 3:
    Serial.print("Wireless Sony DualShock Controller found ");
    break;
  }
}



void loop()
{
   // UART_Control();
    //CAR_Control();
      /* You must Read Gamepad to get new values and set vibration
values
    ps2x.read_gamepad(small motor on/off, larger motor strenght from
0-255)
    if you don't enable the rumble, use ps2x.read_gamepad(); with no
values
    You should call this at least once a second
  */
  if (error == 1) //skip loop if no controller found
    return;
```

```
  if (type == 2) { //Guitar Hero Controller
    return;
  }
  else  { //DualShock Controller
    ps2x.read_gamepad(false, vibrate); //read controller and set large
motor to spin at 'vibrate' Motor_PWM


//start
    if (ps2x.Button(PSB_START))  {
      Serial.println("Start is being held");
      ADVANCE(500,500,500,500);


    }

    if (ps2x.Button(PSB_PAD_UP)) {
      Serial.println("Up held this hard: ");
     ADVANCE(500,500,500,500);
    }


    if (ps2x.Button(PSB_PAD_DOWN)) {
      Serial.print("Down held this hard: ");
      BACK();
    }


    if (ps2x.Button(PSB_PAD_LEFT)) {
      Serial.println("turn left ");
      LEFT_2();
    }


    if (ps2x.Button(PSB_PAD_RIGHT)) {
      Serial.println("turn right");
      RIGHT_2();
    }
// Stop
    if (ps2x.Button(PSB_SELECT)) {
      Serial.println("stop");
      STOP();
    }
// 左平移
    if (ps2x.Button(PSB_PINK)) {
      Serial.println("motor_pmove_left");
      LEFT_1();
    }

    if (ps2x.Button(PSB_RED)) {
      Serial.println("motor_pmove_right");
```

```
      RIGHT_1();
    }
    delay(20);

  }
  if (ps2x.Button(PSB_L1) || ps2x.Button(PSB_R1)) { //print stick
values if either is TRUE
    Serial.print("Stick Values:");
    Serial.print(ps2x.Analog(PSS_LY), DEC); //Left stick, Y axis. Other
options: LX, RY, RX
    Serial.print(",");
    Serial.print(ps2x.Analog(PSS_LX), DEC);
    Serial.print(",");
    Serial.print(ps2x.Analog(PSS_RY), DEC);
    Serial.print(",");
    Serial.println(ps2x.Analog(PSS_RX), DEC);

    int LY = ps2x.Analog(PSS_LY);
    int LX = ps2x.Analog(PSS_LX);
    int RY = ps2x.Analog(PSS_RY);
    int RX = ps2x.Analog(PSS_RX);

    if (LY < 127)
    {

     // Motor_PWM = 1.5 * (127 - LY);
      ADVANCE(500,500,500,500);
      //Motor_PWM = 1.5 * (127 - LX);
      delay(20);
      Serial.print("ADVANCE");
    }

    if (LY > 127)
    {
      //Motor_PWM = 1.5 * (LY - 128);
      BACK();
      delay(20);
      Serial.print("ARRIERE");
    }

    if (LX < 128)
    {
      //Motor_PWM = 1.5 * (127 - LX);
      LEFT_2();
      delay(20);
      Serial.print("LEFT_2");
    }

    if (LX > 128)
```

```
    {
      //Motor_PWM = 1.5 * (LX - 128);
      RIGHT_2();
      delay(20);
      Serial.print("RIGHT_2");
    }

    if (LY == 127 && LX == 128)
    {
      STOP();
      delay(20);
    }

  }
}
```