Robot mars

Introduction:

Composants materiels

1. **Arduino Board**

2. **Moteur shield**

3. **4*Moteur Dc**

4. **6*Batteries**

5. **2*MG996R Servomoteur**

6. Manette

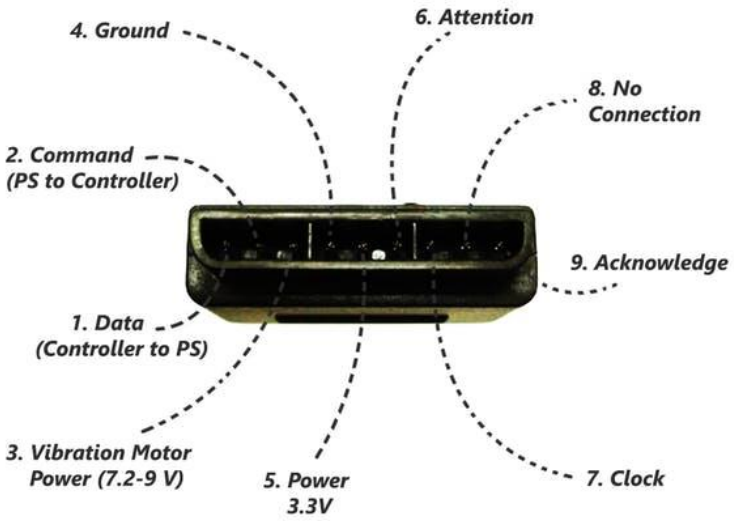
7. Carte fabrique

8. Des fils

Manette sans fil PS2:

I. Comment connecter le contrôleur PS2 à Arduino

Le récepteur du contrôleur PS2 comprend 9 broches :

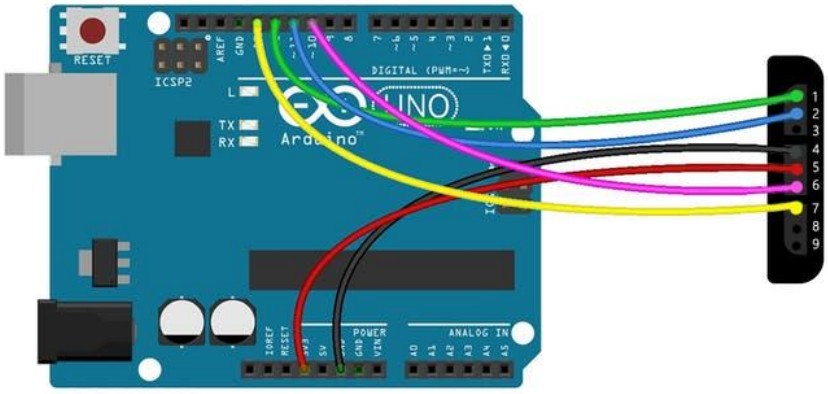


- Données (Data) : ligne maîtresse pour l'envoi de données à l'esclave (MOSI)
- Commande (Command) : ligne esclave pour l'envoi de données au maître (MISO)
- Vibration (Vibration) : alimentation des moteurs à vibration ; 7,2 volts à 9 volts
- Terre (Ground) : circuits terre
- VCC : alimentation des circuits ; 3,3 volts
- Attention : broche CS ou Chip Select pour appeler l'esclave et préparer la connexion
- Horloge (Clock) : équivalent à la broche SCK pour l'horloge
- Pas de connexion (No Connection) : inutile
- Acquitter (Acknowledge) : acquitter le signal du contrôleur vers le récepteur PS2

II. Interface PS2 et Arduino :

Pour utiliser un contrôleur PS2, vous devez introduire la clé du contrôleur dans Arduino. Choisissez ensuite une fonction appropriée pour chaque touche en fonction de votre projet.

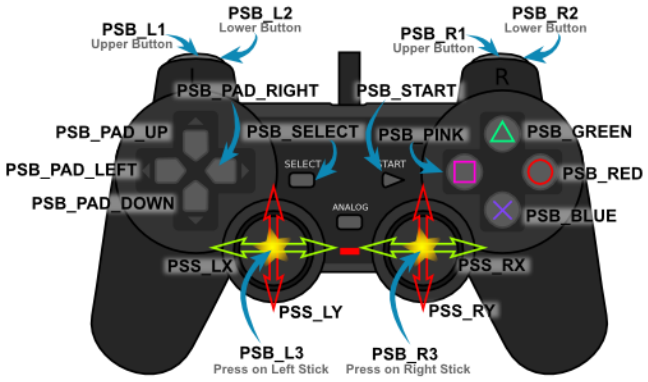
III. Circuit



IV. Code

Vous devez utiliser la bibliothèque PS2X pour ce code.

Après avoir ajouté la bibliothèque à Arduino, vous pouvez ouvrir l'exemple de bibliothèque PS2X ou copier le code suivant et le télécharger sur votre carte. Vous pouvez voir les résultats dans la fenêtre du moniteur série en appuyant sur différentes touches.



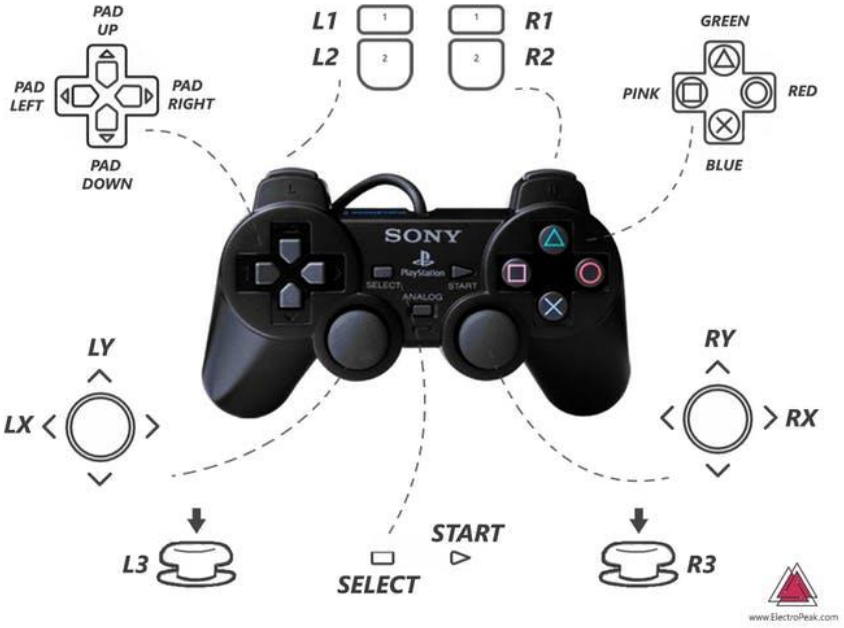
Les fonctions les plus pratiques de cette bibliothèque sont :

a. **ps2x.config_gamepad(clock, command, attention, data, Pressures? Rumble?);** La fonction règle la broche du contrôleur et la sensibilité à la pression et aux vibrations des moteurs. Si vous souhaitez des touches insensibles à la pression ou si les moteurs n'ont pas de vibrations, définissez Pressions et Rumble comme faux. Cette fonction renvoie la valeur de l'erreur.
b. **ready();** détermine le type de contrôleur détecté. 0 signifie que le contrôleur n'est pas détecté correctement, 1 signifie la détection du

contrôleur DualShock et 2 signifie la détection du contrôleur GuitarHero.

c. **read_gamepad(boolean motor1, byte motor2);** la fonction commence à lire l'état des touches lorsque l'état de vibration du moteur est déterminé. (le moteur 2 est le plus gros.)
d. **Button (but type) ;** La fonction renvoie 1 lorsque la touche spécifique de l'argument de fonction est enfoncée. Dans le contrôleur DualShock, les clés sont nommées comme suit :

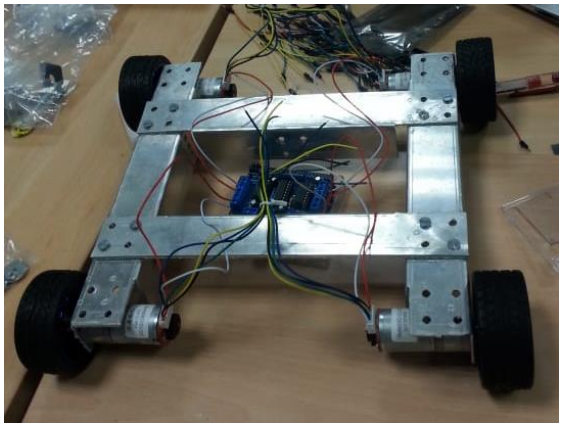
Key	Function	Digital/Analog
PSB_SELECT	OK	Digital
PSB_START	OK	Digital
PSB_PAD_UP	UP	Analog
PSB_PAD_DOWN	DOWN	Analog
PSB_PAD_LEFT	LEFT	Analog
PSB_PAD_RIGHT	RIGHT	Analog
PSB_BLUE	X	Analog
PSB_GREEN	Triangle	Analog
PSB_PINK	Square	Analog
PSB_RED	Circle	Analog
PSB_L3	L3	Digital
PSB_R3	R3	Digital
PSB_L2	L2	Analog
PSB_R2	R2	Analog
PSB_L1	L1	Analog
PSB_R1	R1	Analog
PSB_RX	Joystick right x	Analog
PSB_RY	Joystick right y	Analog
PSB_LX	Joystick left x	Analog
PSB_LY	Joystick left y	Analog



e. **Analog (but type);** La fonction renvoie la valeur des clés analogiques, vous pouvez alors décider de votre opération en conséquence.

f. **CODE**
https://create.arduino.cc/projecthub/code_files/256376/download

Moteur DC et moteur shield



- **Definir la bibliothèque et declarer les 4 moteur DC dans la carte de moteur shield**

```
#include <AFMotor.h>
AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);
```

- **Déterminer la vitesse de ces moteurs dans la fonction loop()**

```
motor1.setSpeed(150);
motor2.setSpeed(150);
motor3.setSpeed(150);
motor4.setSpeed(150);
```

- **Definir la fonction de mouvment de ces Moteur DC**

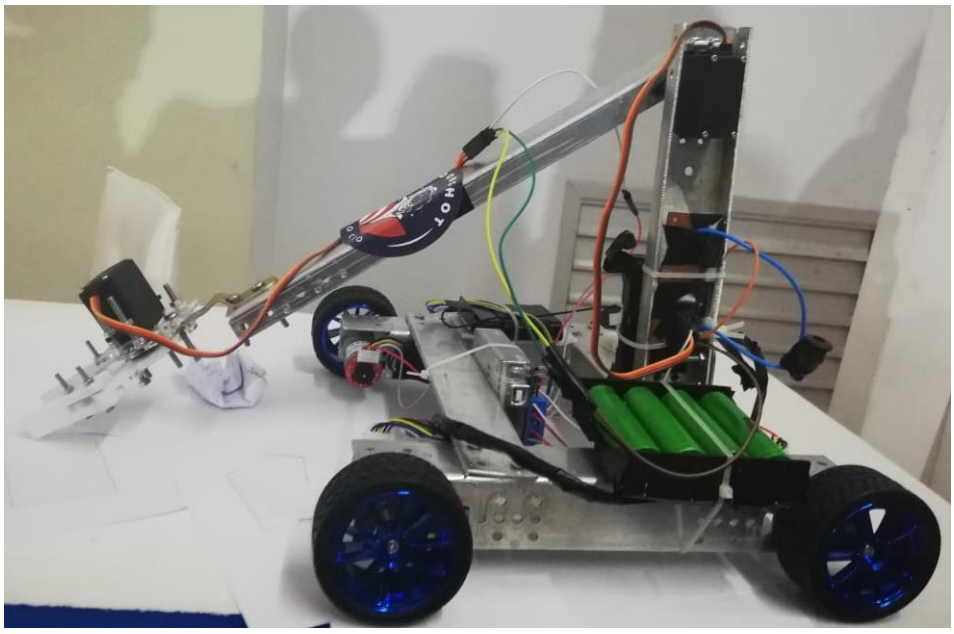
```
void backward(){
 //back
 motor1.run(BACKWARD);

 motor2.run(BACKWARD);

 motor3.run(BACKWARD);

 motor4.run(BACKWARD);

 }
void forward(){
  //forward
  motor1.run(FORWARD);

  motor2.run(FORWARD);

  motor3.run(FORWARD);

  motor4.run(FORWARD);

  }
  void left(){
      //left
  motor1.run(FORWARD);

  motor2.run(FORWARD);

  motor3.run(BACKWARD);

  motor4.run(BACKWARD);

    }
```

```
void right(){
 //right
   motor1.run(BACKWARD);

 motor2.run(BACKWARD);

 motor3.run(FORWARD);

 motor4.run(FORWARD);

 }
  void stopp(){


motor1.run(RELEASE);

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(RELEASE);

    }
```

- **Determiner la Bouton de la manette pour utilise la fonction de Moteur**

```
  Serial.print("Up held this hard: ");
  Serial.println(ps2x.Analog(PSAB_PAD_UP), DEC);
  forward();
 }
 else if(ps2x.Button(PSB_PAD_RIGHT)){
  Serial.print("Right held this hard: ");
   Serial.println(ps2x.Analog(PSAB_PAD_RIGHT), DEC);
 right();
 }
 else if(ps2x.Button(PSB_PAD_LEFT)){
  Serial.print("LEFT held this hard: ");
   Serial.println(ps2x.Analog(PSAB_PAD_LEFT), DEC);
 left();
 }
 else if(ps2x.Button(PSB_PAD_DOWN)){
  Serial.print("DOWN held this hard: ");
Serial.println(ps2x.Analog(PSAB_PAD_DOWN), DEC);
 backward();
 }
 else {
   stopp();
```

Servo Moteur



Code

- ✓ **Definir la bibliothèque et servo dans bibliothèque**

```
#include <Servo.h>
Servo servo01;
Servo servo02;
```

- ✓ **Definir la variable nécessaire pour utiliser servomoteur**

```
int servo1PPos;
int j=120;
int i=31;
```

- ✓ **Definir la pin utlisee dans le moteur shiled**

```
servo02.attach(A4);
servo01.attach(A5);
```

- ✓ **Definir ler postion de ces servomoteurs**

```
servo01.write(120);
servo02.write(30);
```

- ✓ **Determiner la Bouton de la manette pour utilise la fonction de Moteur**

Servo 1 :

```
if(ps2x.Button(PSB_R2)){
 Serial.print("R2 pressed");
 servo01.write(j);
 j--;
}
if(ps2x.Button(PSB_R1))
{
  Serial.print("R1");
 servo01.write(j);
     j++;
}
```

Servo 2 :

```
if(ps2x.ButtonPressed(PSB_RED)){
      Serial.println("Circle just pressed");
      servo02.write(i);
      i=i-5;}
if(ps2x.NewButtonState(PSB_BLUE)) {
      Serial.println("X just changed");
     servo02.write(i);
      i=i+5;
```