

```

#include <Servo.h>
#include <PS2X_lib.h>

#define PS2_DAT      A0
#define PS2_CMD      A1
#define PS2_SEL      A2
#define PS2_CLK      A3
#define pressures    true
#define pressures    false
#define rumble        true
#define rumble        false

const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;

PS2X ps2x;
int error = 0;
byte type = 0;
byte vibrate = 0;

Servo swivel;

int pwm_a = 3;    // Channel A speed
int pwm_b = 6;    // Channel B speed
int dir_a0 = 4;   // Channel A direction 0
int dir_a1 = 5;   // Channel A direction 1
int dir_b0 = 7;   // Channel B direction 0
int dir_b1 = 8;   // Channel B direction 1

char inbit; // A place to store serial input

int swivelpos = 90; // Servo position

void setup()
{
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Pour a bowl of serial

  swivel.attach(11); // Attach servo to pin 11
  swivel.write(swivelpos);

  pinMode(pwm_a, OUTPUT); // Set control pins to be outputs

```

```

pinMode(pwm_b, OUTPUT);
pinMode(dir_a0, OUTPUT);
pinMode(dir_a1, OUTPUT);
pinMode(dir_b0, OUTPUT);
pinMode(dir_b1, OUTPUT);
//delay(1)
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");

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. ");

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 rightforward(int speed) // Move RightForward
{
```

```
digitalWrite(dir_a0, 0);
digitalWrite(dir_a1, 1);
digitalWrite(dir_b0, 0);
digitalWrite(dir_b1, 0);
```

```
analogWrite(pwm_a, speed);
analogWrite(pwm_b, speed);
```

```
}
```

```
void shutoff() // Stop Motors w/o braking
{
```

```
digitalWrite(dir_a0, 0);
digitalWrite(dir_a1, 0);
digitalWrite(dir_b0, 0);
digitalWrite(dir_b1, 0);
```

```
analogWrite(pwm_a, 0);
analogWrite(pwm_b, 0);
```

```
}
```

```
void leftforward(int speed) // Move LeftForward
{
```

```
digitalWrite(dir_a0, 0);
digitalWrite(dir_a1, 0);
digitalWrite(dir_b0, 0);
digitalWrite(dir_b1, 1);
```

```

analogWrite(pwm_a, speed);
analogWrite(pwm_b, speed);

}
void rightbackward(int speed) // Move RightBackward
{

digitalWrite(dir_a0, 1);
digitalWrite(dir_a1, 0);
digitalWrite(dir_b0, 0);
digitalWrite(dir_b1, 0);

analogWrite(pwm_a, speed);
analogWrite(pwm_b, speed);

}
void leftbackward(int speed) // Move LeftBackward
{

digitalWrite(dir_a0, 0);
digitalWrite(dir_a1, 0);
digitalWrite(dir_b0, 1);
digitalWrite(dir_b1, 0);

analogWrite(pwm_a, speed);
analogWrite(pwm_b, speed);

}
void loop() {

    if(error == 1)
        return;

    if(type == 2){
        ps2x.read_gamepad();

        if(ps2x.ButtonPressed(GREEN_FRET))
            Serial.println("Green Fret Pressed");
        if(ps2x.ButtonPressed(RED_FRET))
            Serial.println("Red Fret Pressed");
        if(ps2x.ButtonPressed(YELLOW_FRET))
            Serial.println("Yellow Fret Pressed");
        if(ps2x.ButtonPressed(BLUE_FRET))
            Serial.println("Blue Fret Pressed");
        if(ps2x.ButtonPressed(ORANGE_FRET))

```

```

    Serial.println("Orange Fret Pressed");

    if(ps2x.ButtonPressed(STAR_POWER))
        Serial.println("Star Power Command");

    if(ps2x.Button(UP_STRUM))
        Serial.println("Up Strum");
    if(ps2x.Button(DOWN_STRUM))
        Serial.println("DOWN Strum");

    if(ps2x.Button(PSB_START))
        Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
        Serial.println("Select is being held");

    if(ps2x.Button(ORANGE_FRET)) {
        Serial.print("Wammy Bar Position:");
        Serial.println(ps2x.Analog(WHAMMY_BAR), DEC);
    }
}

else {
    ps2x.read_gamepad(false, vibrate);

    if(ps2x.Button(PSB_START))
        Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
        Serial.println("Select is being held");

    if(ps2x.Button(PSB_PAD_UP)) {
        Serial.print("Up held this hard: ");
        Serial.println(ps2x.Analog(PSAB_PAD_UP), DEC);
    }
    if(ps2x.Button(PSB_PAD_RIGHT)){
        Serial.print("Right held this hard: ");
        Serial.println(ps2x.Analog(PSAB_PAD_RIGHT), DEC);
    }
    if(ps2x.Button(PSB_PAD_LEFT)){
        Serial.print("LEFT held this hard: ");
        Serial.println(ps2x.Analog(PSAB_PAD_LEFT), DEC);
    }
    if(ps2x.Button(PSB_PAD_DOWN)){
        Serial.print("DOWN held this hard: ");
        Serial.println(ps2x.Analog(PSAB_PAD_DOWN), DEC);
    }
}

```

```

vibrate = ps2x.Analog(PSAB_CROSS);
if (ps2x.NewButtonState()) {
  if(ps2x.Button(PSB_L3))
    Serial.println("L3 pressed");
  if(ps2x.Button(PSB_R3))
    Serial.println("R3 pressed");
  if(ps2x.Button(PSB_L2))
    Serial.println("L2 pressed");
  if(ps2x.Button(PSB_R2))
    Serial.println("R2 pressed");
  if(ps2x.Button(PSB_TRIANGLE))
    Serial.println("Triangle pressed");
}

if(ps2x.ButtonPressed(PSB_CIRCLE))
  Serial.println("Circle just pressed");
if(ps2x.ButtonReleased(PSB_SQUARE))
  Serial.println("Square just released");

if(ps2x.NewButtonState(PSB_CROSS)) {
  Serial.println("X just changed");
}
if(ps2x.ButtonReleased(PSB_SQUARE)) {
  Serial.println("Square just released");
}

if(ps2x.Button(PSB_R2)){
  rightforward(200);
  delay(30);
  shutoff();
}
if(ps2x.Button(PSB_L2)){
  leftforward(200);
  delay(30);
  shutoff();
}
if(ps2x.Button(PSB_R1)){
  rightbackward(200);
  delay(30);
  shutoff();
}
if(ps2x.Button(PSB_L1)){
  leftbackward(200);
  delay(30);
}

```

```
    shutoff();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance= duration*0.034/2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);

  if(distance<6){
    shutoff();
    rightbackward(1000);
    delay(30);
    shutoff();
  }
  if(distance<6){
    shutoff();
    leftbackward(1000);
    delay(30);
    shutoff();
  }
}
}
}
```