```
#include <PS2X_lib.h>  //for v1.6
#include <L298N.h>

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 conect the controller,
//or call config_gamepad(pins) again after connecting the controller.
int error = 0;
byte type = 0;
byte vibrate = 0;

#define ENA 3
#define IN1 4
#define IN2 5
#define IN3 6
#define IN4 7
#define ENB 9

L298N motorA(ENA, IN1, IN2);
L298N motorB(ENB, IN3, IN4);

//mapping analog sticks with range -255 - 255
int left_y= map(ps2x.Analog(PSS_LY),0,255,-255,255);
 int right_y= map(ps2x.Analog(PSS_RY),0,255,-255,255);

//for neopixel
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN        8 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 5 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 250 // Time (in milliseconds) to pause between pixels


void setup(){
 pinMode(2, OUTPUT);
 pinMode(1, OUTPUT);


 Serial.begin(57600);
motorA.setSpeed(230);
```

```
motorB.setSpeed(230);
 //CHANGES for v1.6 HERE!!! **************PAY ATTENTION*************

 error = ps2x.config_gamepad(13,11,10,12, true, false);   //setup pins and
settings:  GamePad(clock, command, attention, data, Pressures?, Rumble?) check
for error

 if(error == 0){
   Serial.println("Found Controller, configured successful");
   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("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. ");

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

   type = ps2x.readType();
     switch(type) {
       case 0:
        Serial.println("Unknown Controller type");
       break;
       case 1:
        Serial.println("DualShock Controller Found");
       break;
       case 2:
         Serial.println("GuitarHero Controller Found");
       break;
     }

}

//for neopixel
// These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  // INITIALIZE NeoPixel strip object (REQUIRED)
```

```
void loop(){
   pixels.begin();
   pixels.clear();
   /* You must Read Gamepad to get new values
   Read GamePad 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

   ps2x.read_gamepad();          //read controller

   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))          //will be TRUE as long as button is
pressed
     Serial.println("Up Strum");
    if(ps2x.Button(DOWN_STRUM))
     Serial.println("DOWN Strum");


    if(ps2x.Button(PSB_START))                   //will be TRUE as long as
button is pressed
         Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
         Serial.println("Select is being held");


    if(ps2x.Button(ORANGE_FRET)) // print stick value IF TRUE
    {
        Serial.print("Wammy Bar Position:");
        Serial.println(ps2x.Analog(WHAMMY_BAR), DEC);
```

```
    }
 }

 else { //DualShock Controller

    ps2x.read_gamepad(false, vibrate);          //read controller and set large
motor to spin at 'vibrate' speed

    if(ps2x.Button(PSB_START))                   //will be TRUE as long as
button is pressed
         Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
         Serial.println("Select is being held");


     if(ps2x.Button(PSB_PAD_UP)) {         //will be TRUE as long as button is
pressed
       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_BLUE);        //this will set the large motor
vibrate speed based on
                                              //how hard you press the blue (X)
button

    if (ps2x.NewButtonState())               //will be TRUE if any button
changes state (on to off, or off to on)
    {



        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_GREEN))
```

```
        Serial.println("Triangle pressed");

    }




    if(ps2x.Button(PSB_L1) || ps2x.Button(PSB_R1)) // print stick values if
either is TRUE
    {
        Serial.print("Stick Values:");
        Serial.print(map(ps2x.Analog(PSS_LY),0,255,-255,255), DEC); //Left
stick, Y axis. Other options: LX, RY, RX
        Serial.print(",");
        Serial.print(ps2x.Analog(PSS_LX), DEC);
        Serial.print(",");
        Serial.print(map(ps2x.Analog(PSS_RY),0,255,-255,255), DEC);
        Serial.print(",");
        Serial.println(ps2x.Analog(PSS_RX), DEC);

    }

if (map(ps2x.Analog(PSS_RY),0,255,-255,255) > -1){
  motorB.backward();
motorB.setSpeed(map(ps2x.Analog(PSS_RY),0,255,-255,255));
}
if (map(ps2x.Analog(PSS_RY),0,255,-255,255) < -1){

  motorB.forward();
  motorB.setSpeed((map(ps2x.Analog(PSS_RY),0,255,-255,255)) * -1);
}
if (map(ps2x.Analog(PSS_RY),0,255,-255,255) == -1){

  motorB.stop();
}
if (map(ps2x.Analog(PSS_LY),0,255,-255,255) > -1){
  motorA.backward();
motorA.setSpeed(map(ps2x.Analog(PSS_LY),0,255,-255,255));
}
if (map(ps2x.Analog(PSS_LY),0,255,-255,255) < -1){

  motorA.forward();
  motorA.setSpeed((map(ps2x.Analog(PSS_LY),0,255,-255,255)) * -1);
}
if (map(ps2x.Analog(PSS_LY),0,255,-255,255) == -1){

  motorA.stop();

}
if(ps2x.Button(PSB_RED)){             //will be TRUE if button was JUST pressed
        Serial.println("Circle being held");
```

```
  pixels.setPixelColor(0, pixels.Color(255, 94, 0)); //neopixel color by (R, G,
B) values
  pixels.setPixelColor(1, pixels.Color(255, 94, 0));
  pixels.setPixelColor(2, pixels.Color(255, 94, 0));
  pixels.setPixelColor(3, pixels.Color(255, 94, 0));
  pixels.setPixelColor(4, pixels.Color(255, 94, 0));
  pixels.show(); //show the new color settings on the neopixels

    }
if(ps2x.ButtonPressed(PSB_BLUE)){
  pixels.setPixelColor(0, pixels.Color(58, 74, 220));
  pixels.setPixelColor(1, pixels.Color(58, 74, 220));
  pixels.setPixelColor(2, pixels.Color(58, 74, 220));
  pixels.setPixelColor(3, pixels.Color(58, 74, 220));
  pixels.setPixelColor(4, pixels.Color(58, 74, 220));
  pixels.show();
}
if(ps2x.ButtonPressed(PSB_GREEN)){
  pixels.setPixelColor(0, pixels.Color(68, 220, 58));
  pixels.setPixelColor(1, pixels.Color(68, 220, 58));
  pixels.setPixelColor(2, pixels.Color(68, 220, 58));
  pixels.setPixelColor(3, pixels.Color(68, 220, 58));
  pixels.setPixelColor(4, pixels.Color(68, 220, 58));
  pixels.show();
}
if(ps2x.ButtonPressed(PSB_PINK)){
  pixels.setPixelColor(0, pixels.Color(117, 0, 168));
  pixels.setPixelColor(1, pixels.Color(117, 0, 168));
  pixels.setPixelColor(2, pixels.Color(117, 0, 168));
  pixels.setPixelColor(3, pixels.Color(117, 0, 168));
  pixels.setPixelColor(4, pixels.Color(117, 0, 168));
  pixels.show();
}
 if(ps2x.Button(PSB_START)){
  pixels.clear(); //all neopixels off
  pixels.show();
 }
if(ps2x.Button(PSB_R1)){
  motorB.forward();
  motorA.backward();
}
if(ps2x.Button(PSB_R2)){
  pixels.setPixelColor(0, pixels.Color(255, 255, 255));
  pixels.setPixelColor(1, pixels.Color(0, 0, 0));
  pixels.setPixelColor(2, pixels.Color(0, 0, 0));
  pixels.setPixelColor(3, pixels.Color(0, 0, 0));
  pixels.setPixelColor(4, pixels.Color(0, 0, 0));
  pixels.show();
  delay(50);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.setPixelColor(1, pixels.Color(255, 247, 0));
  pixels.setPixelColor(2, pixels.Color(0, 0, 0));
  pixels.setPixelColor(3, pixels.Color(0, 0, 0));
  pixels.setPixelColor(4, pixels.Color(0, 0, 0));
```

```
  pixels.show();
  delay(50);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.setPixelColor(1, pixels.Color(0, 0, 0));
  pixels.setPixelColor(2, pixels.Color(255, 80, 0));
  pixels.setPixelColor(3, pixels.Color(0, 0, 0));
  pixels.setPixelColor(4, pixels.Color(0, 0, 0));
  pixels.show();
  delay(50);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.setPixelColor(1, pixels.Color(0, 0, 0));
  pixels.setPixelColor(2, pixels.Color(0, 0, 0));
  pixels.setPixelColor(3, pixels.Color(69, 239, 35));
  pixels.setPixelColor(4, pixels.Color(0, 0, 0));
  pixels.show();
  delay(50);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  pixels.setPixelColor(1, pixels.Color(0, 0, 0));
  pixels.setPixelColor(2, pixels.Color(0, 0, 0));
  pixels.setPixelColor(3, pixels.Color(0, 0, 0));
  pixels.setPixelColor(4, pixels.Color(35, 123, 239));
  pixels.show();
  delay(50);
}


 delay(50);

}
}
```