```
#include <PS2X_lib.h>
#include <Servo.h>        //copy from servo code
            // a maximum of eight servo objects can be created
Servo myservoA;  // works on motor A
Servo myservoB;  // works on motor B
Servo myservoC;  // works on motor C
PS2X ps2x; // create PS2 Controller Class



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

void setup(){
  Serial.begin(9600);
  myservoA.attach(3);  // attaches the servo on pin 3 to the servo object  //copy from servo code
  myservoB.attach(5);  // attaches the servo on pin 5 to the servo object
  myservoC.attach(10);  // attaches the servo on pin 6 to the servo object

  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(10, OUTPUT);

  error = ps2x.config_gamepad(9,8,7,6, true, true);   //setup pins (in this order) and settings:
GamePad(clock, command, attention, data, Pressures, Rumble)
                                    //Pay attention to order of "command" and "attention" wires



  if(error == 0){
```

```
   Serial.println("Found Controller, configured successful");
   Serial.println("HOLDING L1 or R1 will print out the ANALOG STICK values.");
   Serial.println("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.println("Unknown Controller type");
   break;
  case 1:
   Serial.println("DualShock Controller Found");
   break;
  }
}

void loop(){
  /* You must Read Gamepad to get new values
   Read GamePad and set vibration values
   ps2x.read_gamepad(small motor on/off, larger motor strength 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;
 } else { //DualShock Controller Found
   ps2x.read_gamepad(false, vibrate);      //false unless a command written for vibrate (if "true" large motor spins at 'vibrate' speed)
  vibrate = ps2x.Analog(PSAB_BLUE);       //this will set the large motor vibrate speed based on how hard you press the blue (X) button
 }

/*Large portions below commented out are included in source code, uncomment as needed.
 *Placing code under the "if" commands allow you to configure buttons.
 */

   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)) {
     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);
  }


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

    if(ps2x.Button(PSB_L1))
      Serial.println("L1 pressed");
    if(ps2x.Button(PSB_R1))
      Serial.println("R1 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.ButtonPressed(PSB_RED)) {        //will be TRUE if button is pressed
```

```
 if(ps2x.ButtonReleased(PSB_RED)) {        //will be TRUE if button is released

 if(ps2x.ButtonPressed(PSB_PINK)) {        //will be TRUE if button is pressed

 if(ps2x.ButtonReleased(PSB_PINK)) {        //will be TRUE if button is released
*/
   if(ps2x.ButtonPressed(PSB_RED)){            //will be TRUE if button was JUST pressed
     Serial.println("Circle just pressed");}

   if(ps2x.ButtonPressed(PSB_PINK)){            //will be TRUE if button was JUST released
     Serial.println("Square just pressed");}

   if(ps2x.NewButtonState(PSB_BLUE)) {          //will be TRUE if button was JUST pressed OR released
     Serial.println("X just changed");
   }
//-----------------------------------------------------------------------------------------------------------------------
  int val=ps2x.Analog(PSS_RX); // reads the value of RX on the the PS2 receiver (value between 0 and
255)
  int mapval=map(val,0,255,2000,1000); //reads val, creates a mapval integer between 1000 and 2000
that motor controller reads

  int x=ps2x.Analog(PSS_LX); // reads the value of LX on the the PS2 receiver (value between 0 and 255)
  int y=ps2x.Analog(PSS_LY); // reads the value of LY on the the PS2 receiver (value between 0 and 255)

  if(x==128 && y==127){
    myservoA.writeMicroseconds(mapval);  //stationary rotation
    myservoB.writeMicroseconds(mapval);
    myservoC.writeMicroseconds(mapval);
    delay(10);
```

```
}

int mapx=map(x,0,255,100,-100);
int mapy=map(y,0,255,100,-100); //values from PS2 controller are read from 0-255, read as 100 to -100 value to represent a cartesian plane with a 0 in the center to simplify calculations.

float theta= atan2(mapx,mapy); // arc tangent of x/y

int L=sqrt(mapx*mapx+mapy*mapy);  //Pythagorean theorem

float cosa=L*cos(150*M_PI/180-theta); //150 degrees minus theta
float cosb=L*cos(30*M_PI/180-theta);  //first value is L (hypotenuse)
float cosc=L*cos(270*M_PI/180-theta); //second value (parentheses section) represents adjacent side along x axis

int Fa=map(cosa,-142,142,1000,2000); //-142 and 142 are limits of calculations above (should be)
int Fb=map(cosb,-142,142,1000,2000);
int Fc=map(cosc,-142,142,1000,2000);

if(val==128){                //F values indicate motor rotation, robot navigation
myservoA.writeMicroseconds(Fa);
myservoB.writeMicroseconds(Fb);
myservoC.writeMicroseconds(Fc);
delay(10);
}

//Next four sets of code allow you to read stick values
if(ps2x.Button(PSB_L1)){
```

```
Serial.print(Fa,DEC);
Serial.print(",");
Serial.print(Fb,DEC);
Serial.print(",");
Serial.println(Fc,DEC);
}

if(ps2x.Button(PSB_L2)){
Serial.print(ps2x.Analog(PSS_LX), DEC);
Serial.print(",");
Serial.println(ps2x.Analog(PSS_LY), DEC);
}                              //same values as R2

if(ps2x.Button(PSB_R1)){
Serial.print(ps2x.Analog(PSS_RX), DEC);
Serial.print(",");
Serial.println(ps2x.Analog(PSS_RY), DEC);
}

if(ps2x.Button(PSB_R2)){
Serial.print(x,DEC);
Serial.print(",");
Serial.println(y,DEC);
}                              //same values as L2

delay(50);
}
```