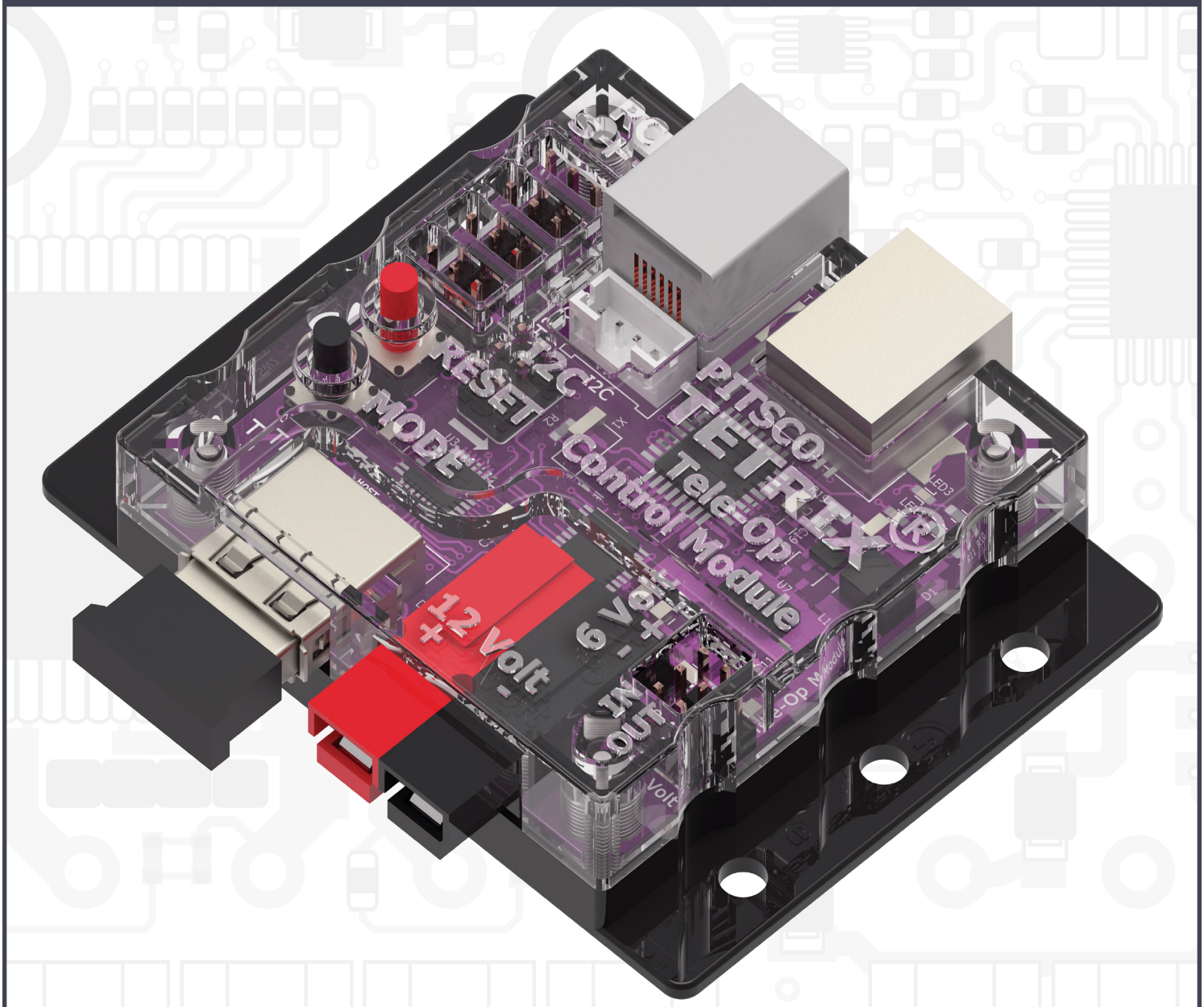
PITSCO

TETRIX® MAX



TETRIX® Tele-Op Control Module Technical Guide

For use with SONY PS4 DUALSHOCK 4 gaming controller

PITSCO EDUCATION

45292

Content advising by Paul Uttley.
SolidWorks® *Composer*™ and *KeyShot*® renderings by Tim Lankford, Brian Eckelberry, and Jason Redd.
Desktop publishing by Todd McGeorge.

A downloadable PDF of the most recent version of this guide can be found at
Pitsco.com/TETRIX-Tele-Op-Control-Module#resources.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

V1.0
01/19

TETRIX® Tele-Op Control Module Technical Guide

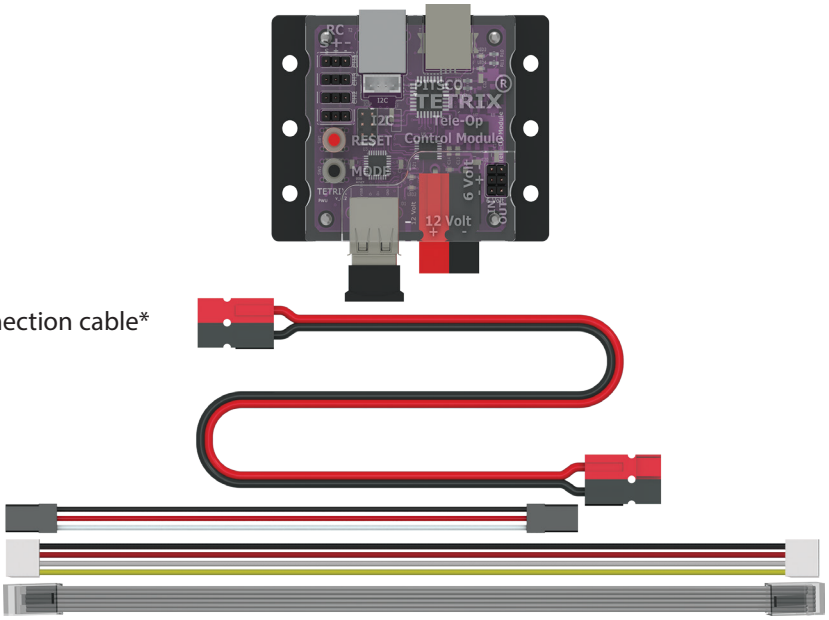
General Description

The TETRIX® Tele-Op Control Module is a peripheral add-on to the programmable TETRIX PRIZM® or PULSE™ Robotics Controllers. It enables tele-op control of a TETRIX robot using a USB/wireless input device such as a gaming controller. Note that the Tele-Op module ships with firmware that supports a **SONY PS4 DUALSHOCK 4** gaming controller by default. The Tele-Op module connects to the PRIZM expansion port or PULSE I2C port to enable the interface of a Bluetooth wireless device, such as a joystick gaming controller, to the TETRIX control system. The module communicates using an I2C protocol for sending and receiving data. The wireless communication is implemented using a Bluetooth dongle plugged into the Tele-Op module's USB host port. A PDF of this guide is available at **Pitsco.com**.

While use of the Tele-Op module with the PULSE controller is not currently supported by the visual programming software TETRIX Ardublockly, please note that the PULSE controller is supported by the Arduino Software (IDE), which includes example sketches.

What's Included

- TETRIX Tele-Op Control Module
- TETRIX MAX Powerpole Extension Cable
- Daisy chain data cable
- Bluetooth dongle
- Four-pin I2C port extension cable*
- Female to female three-pin battery connection cable*



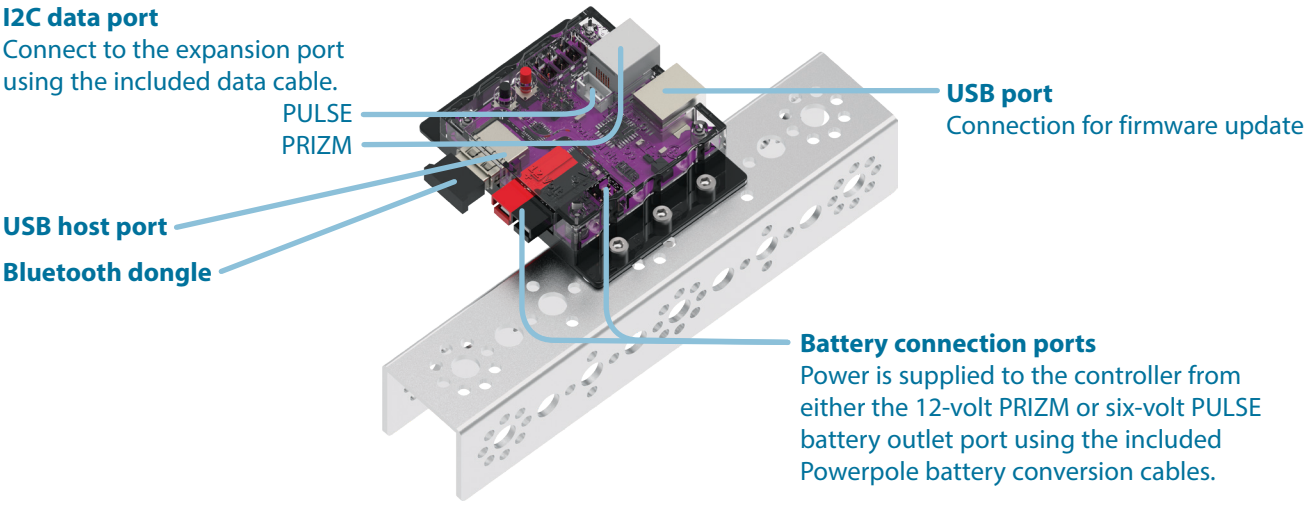
Bluetooth Dongle

It's important to know that not all Bluetooth dongles are created equal. For multiple Bluetooth devices to operate simultaneously within radio range of each other, each dongle must have a unique identifier, which is called a MAC address. Bluetooth guidelines state that manufacturers of dongles are supposed to follow this rule. However, some producers of cheap dongles don't always do this. Instead, they mass produce dongles, all with the same identifier. The Bluetooth dongle that comes with each TETRIX Tele-Op module has been tested to each have a unique identifier. It is recommended that you always use the dongle that ships with the product. Additional replacement dongles are available from Pitsco.

*For use with PULSE controller

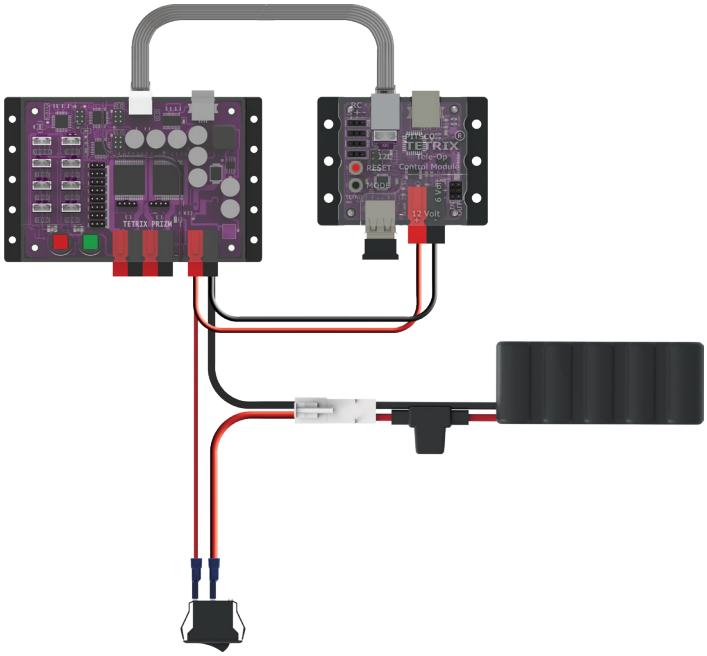
Attaching the Tele-Op Module

When mounting the Tele-Op module on your robot, be sure to place it away from DC or servo motors as they tend to generate frequency noise that could interfere with Bluetooth connectivity. Also, be sure that Tele-Op module is not surrounded by too many metal parts, which could shield the Bluetooth signal.



Connections

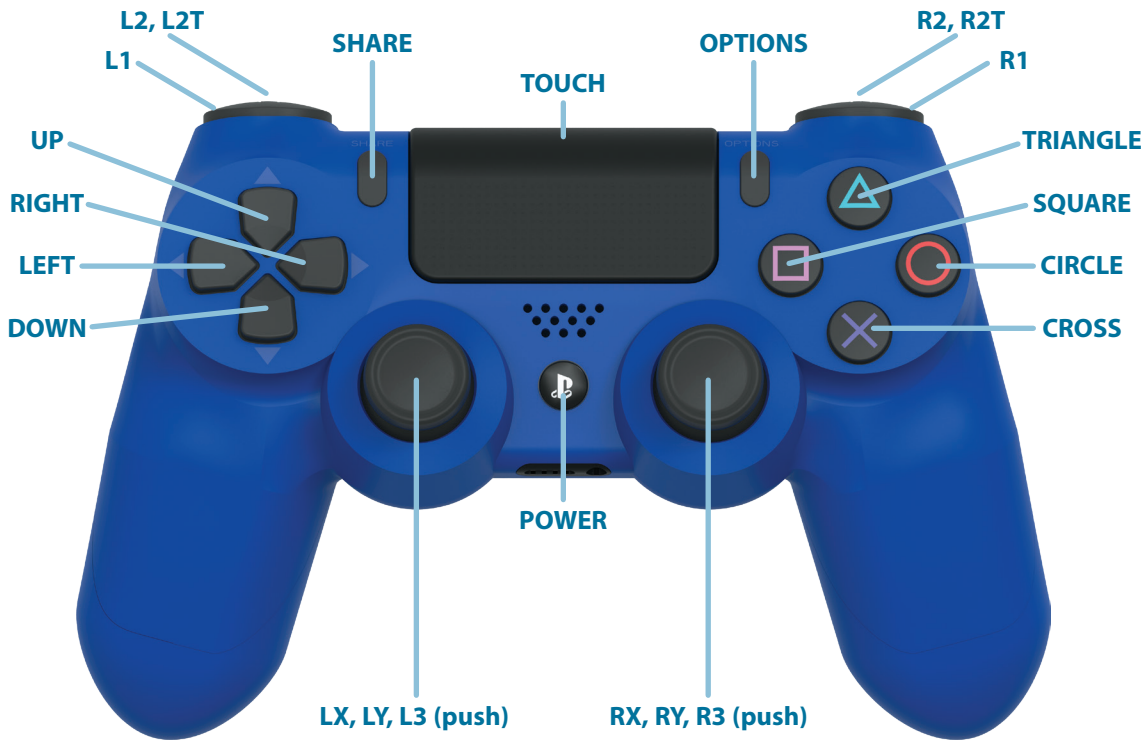
The Tele-Op module connects to the PRIZM battery power expansion terminals using the included Powerpole extension cable. The module's data port connects to the PRIZM expansion port using the included data cable. If your system includes any additional TETRIX DC motor or servo motor expansion modules, the Tele-Op module's data port and power connection can be plugged into the last available data and power ports in the system daisy chain. A USB Bluetooth dongle should be preinstalled in the module's USB host port. If it is not, insert the dongle fully into the USB host port. Secure the module to your TETRIX structure using screws and nuts including in a TETRIX building set. For best operation, place the module as far away as possible from DC motors or servos to avoid electrical noise interference with the Bluetooth signal.



Important Safety Information

Caution: Use only a TETRIX battery pack that is equipped with an in-line safety fuse. Failure to do so could result in damage or injury. Connect the TETRIX battery pack to either the top or bottom red/black power inlet row at the battery connection port. Do **not** connect two battery packs to the PRIZM controller.

SONY PS4 Gaming Controller Diagram



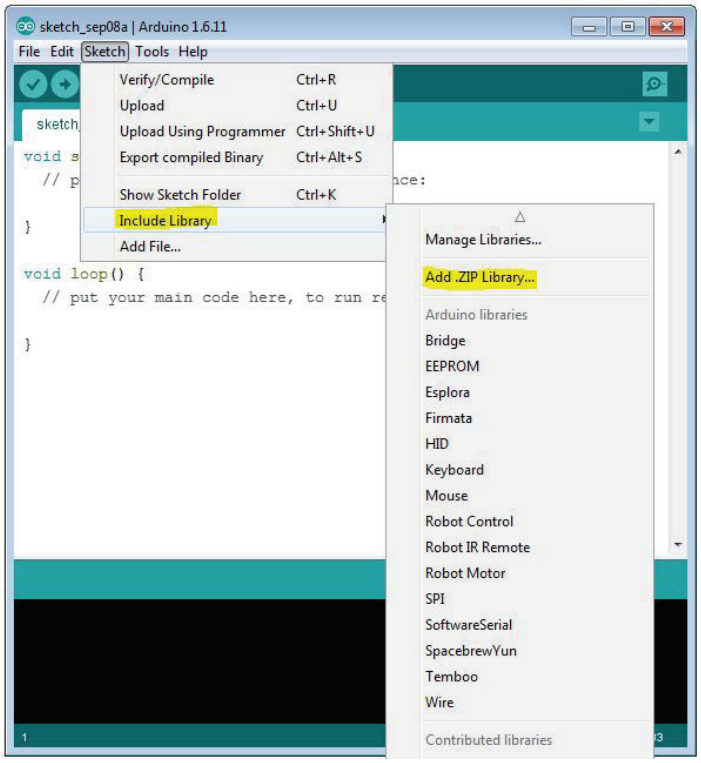
Supported Software and Additional Resources

The TETRIX Tele-Op Control Module is designed to be easily interfaced to the TETRIX PRIZM Robotics Controllers utilizing an Arduino programming library. The library is a simplified set of functions enabling easy and intuitive integration of Tele-Op control operation into PRIZM coding applications created within the Arduino Software (IDE). It is assumed that the Arduino Software (IDE) has already been installed, along with the programming libraries for the PRIZM controller.

- Download the Tele-Op library at **Pitsco.com/TETRIX-Tele-Op-Control-Module#downloads**. The library contains several coding examples that demonstrate each Tele-Op module library function as well as application to PRIZM coding techniques. The code examples provide the easiest way to get up and going with the Tele-Op module. Other devices might be added and supported by the Tele-Op module in the future. Activities that demonstrate functionality with a PS4 gaming controller are available at the website.
- Visit **Pitsco.com/TETRIX-Tele-Op-Control-Module#resources** to view the latest activities, code examples, firmware updates, and list of devices supported by the Tele-Op module.
- Visit **Pitsco.com/TETRIX-PRIZM-Robotics-Controller#downloads** to download the latest software libraries for the TETRIX PRIZM Robotics Controller if you don't have them already installed.

Installing the Tele-Op Module Arduino Programming Library

The TETRIX Tele-Op Arduino Library is distributed as a .zip file available for download at **Pitsco.com/TETRIX-Tele-Op-Control-Module#downloads**. After it's downloaded, you can use the Add .ZIP Library option found under the Sketch menu tab in the Arduino Software (IDE) menu bar. Additional instructions on how to install a library into the Arduino Software (IDE) can be found by visiting the Arduino website at **www.arduino.cc**. For other detailed tips on how to install a library, you can reference the library installation section in the full *TETRIX PRIZM Programming Guide* that is available for download on the Pitsco website.



After the library is installed, close and restart the Arduino Software (IDE). After it's restarted, you can access the coding examples by selecting **Examples** under the File menu and scrolling down to the TETRIX_Tele-Op library.

Pairing the SONY PS4 DUALSHOCK 4 Gaming Controller

To begin using a PS4 controller with the Tele-Op module, you must pair the PS4 controller to the Bluetooth dongle connected to the Tele-Op module. To do this:

1. Be sure the Tele-Op module's Bluetooth dongle is inserted into the USB host port and the PS4 controller's battery is fully charged.
2. Power up the module. After three or four seconds, the green LED on the Tele-Op module should be blinking rapidly.
3. Press and hold down the Share and Power buttons simultaneously on the PS4 controller until the controller's light bar begins to blink white rapidly. This indicates that the PS4 controller is now in discovery mode. Release the buttons.
4. Press the Mode button on the Tele-Op module. The red LED will come on, indicating that the Tele-Op module is attempting to pair with the PS4 controller that is in discovery mode. When they have successfully paired, the Tele-Op module's red LED will turn off and the green LED will be on and solid (not blinking). In addition, the PS4's light bar will change to solid green.

Note: You only need to pair the devices once. After they are paired, simply power up the Tele-Op module and then turn on your PS4 controller by pressing the Power button. The PS4 controller's light bar will turn white and blink slowly while it is linking; it will switch to solid green color as soon as they're linked. It might take up to 10 seconds to link.

Tele-Op Module Library for SONY PS4 DUALSHOCK 4 Gaming Controller

Following is a quick reference of each library function supported by the TETRIX Tele-Op Arduino Library. This command set is to be used in conjunction with a wireless SONY PS4 DUALSHOCK 4 gaming controller. See the diagram earlier in this guide for a map of the PS4 gaming controller buttons and joysticks.

```
ps4.getPS4();
ps4.Connected;
ps4.inRange;
ps4.resetTeleOp();

ps4.Button(L1);
ps4.Button(L2);
ps4.Button(L3);
ps4.Button(R1);
ps4.Button(R2);
ps4.Button(R3);
ps4.Button(L2T);
ps4.Button(R2T);
ps4.Button(UP);
ps4.Button(DOWN);
ps4.Button(RIGHT);
ps4.Button(LEFT);
ps4.Button(TRIANGLE);
ps4.Button(CROSS);
ps4.Button(CIRCLE);
ps4.Button(SQUARE);
ps4.Button(SHARE);
ps4.Button(OPTIONS);
ps4.Button(POWER);
ps4.Button(TOUCH);

ps4.buttons_1;
ps4.buttons_2;

ps4.Touchpad(TOUCHX);
ps4.Touchpad(TOUCHY);

ps4.Angle(PITCH);
ps4.Angle(ROLL);
```

```
ps4.setLED(RED);
ps4.setLED(BLUE);
ps4.setLED(YELLOW);
ps4.setLED(GREEN);

ps4.setRumble(STOP);
ps4.setRumble(SLOW);
ps4.setRumble(FAST);

ps4.setDeadZone(stick, amount);

ps4.Motor(LX);
ps4.Motor(LY);
ps4.Motor(RX);
ps4.Motor(RY);
ps4.Motor(L2T);
ps4.Motor(R2T);
ps4.Motor(PITCH);
ps4.Motor(ROLL);
ps4.Servo(LX);
ps4.Servo(LY);
ps4.Servo(RX);
ps4.Servo(RY);
ps4.Servo(L2T);
ps4.Servo(R2T);
ps4.Servo(PITCH);
ps4.Servo(ROLL);

ps4.Stick(LX);
ps4.Stick(LY);
ps4.Stick(RX);
ps4.Stick(RY);
```

TETRIX Tele-Op Control Module Arduino Library Functions Chart for the SONY DUALSHOCK 4 Gaming Controller

The Tele-Op module ships with firmware for the PS4 gaming controller installed. Please be sure to download and install the latest version of the Tele-Op Arduino libraries at **Pitsco.com/TETRIX-Tele-Op-Control-Module#downloads** for the most up-to-date programming features and functionality.

Description	Function	Coding Example
Get PS4 Controller Data Reads the status of all analog and digital buttons and joysticks from a PS4 game controller connected to the Tele-Op module. Be sure to call this function as frequently as possible in your Arduino code. Each time it is called, the data from the PS4 gaming controller is refreshed.	**ps4.getPS4();** Data Returned: None	void loop() { ps4.getPS4(); }
Check Connection Checks to see if the PS4 gaming controller has been successfully connected to the Tele-Op module.	**ps4.Connected;** Data Returned: 0 = PS4 has disconnected 1 = PS4 has connected	void loop() { ps4.getPS4(); if (ps4.Connected) { // do this if connected } else { // do this if not connected } }
Check Range Checks to see if the PS4 gaming controller Bluetooth signal is in range of the Tele-Op module.	**ps4.inRange;** Data Returned: 0 = PS4 is out of range 1 = PS4 is in range	void loop() { ps4.getPS4(); if (ps4.inRange) { // do this if in range } else { // do this if not in range } }
Reset Tele-Op Module Force resets the module; equivalent to physically pressing the red Stop/Reset button.	**ps4.resetTeleOp();** Data Returned: None	ps4.resetTeleOp(); ***Note:*** *Call this function* ***once*** *in code if there is a need to reset the Tele-Op module.*
Read Digital Button Status Returns the status of a PS4 controller digital button. Digital button function parameters are: L1, L2, L3 R1, R2, R3 UP, DOWN, RIGHT, LEFT TRIANGLE, CROSS, CIRCLE, SQUARE SHARE, OPTIONS, POWER, TOUCH	**ps4.Button(L1);** Data Returned: 0 = Button not pressed 1 = Button pressed	void loop() { ps4.getPS4(); if (ps4.Button(L1)) { // do this if pressed } else { // do this if not pressed } }
Read Analog Button Status Returns the status of a PS4 controller analog trigger button. Analog buttons are the L2 and R2 trigger buttons. Analog button parameters are: L2T, R2T	**ps4.Button(L2T);** *or* **ps4.Button(R2T);** Data Returned: Integer Data Range: 0-255	void loop() { ps4.getPS4(); int x = ps4.Button(L2T); } *The analog value (0-255) of the L2 trigger button is stored in variable* x.

<table>
<tr><th>Description</th><th>Function</th><th>Coding Example</th></tr>
<tr>
<td>Return Button Group Byte Value
Returns the binary byte value of a PS4 button group.

The Tele-Op module can return the digital status of two button groups represented as a binary byte. The digital status of each button in the group is indicated by its corresponding binary digit position in the byte.

Button Group 1 (buttons_1):
L1, L2, R1, R2, CROSS, CIRCLE, SQUARE, TRIANGLE

Button Group 2 (buttons_2):
L3, R3, UP, DOWN, RIGHT, LEFT, SHARE, OPTIONS</td>
<td>ps4.buttons_1;
or
ps4.buttons_2;

Data Returned: Byte

Byte
Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Group 1:
Bit 0: L2
Bit 1: L1
Bit 2: R2
Bit 3: R1
Bit 4: CROSS
Bit 5: CIRCLE
Bit 6: SQUARE
Bit 7: TRIANGLE

Group 2:
Bit 0: UP
Bit 1: DOWN
Bit 2: RIGHT
Bit 3: LEFT
Bit 4: R3
Bit 5: L3
Bit 6: SHARE
Bit 7: OPTIONS</td>
<td>void loop() {
ps4.getPS4();
byte group1 = ps4.buttons_1;
byte group2 = ps4.buttons_2;
}

group1 = the byte value of the buttons_1 group

group2 = the byte value of the buttons_2 group

Tip: Use the Serial Monitor statement with the BIN format parameter to see the binary value of each button group's status.

Serial.println(ps4.buttons_1, BIN);
or
Serial.println(ps4.buttons_2, BIN);</td>
</tr>
<tr>
<td>Read Touch Pad
Returns the x- or y-coordinate of the finger position touched and moved across the surface of the PS4 controller touch pad.</td>
<td>ps4.Touchpad(TOUCHX);
or
ps4.Touchpad(TOUCHY);

Data Returned: Integer</td>
<td>void loop() {
ps4.getPS4();
int x = ps4.Touchpad(TOUCHX);
int y = ps4.Touchpad(TOUCHY);
}

x = the x-coordinate of PS4 touch pad
y = the y-coordinate of PS4 touch pad</td>
</tr>
<tr>
<td>Read Position of Data Mapped to Motor
Returns the position of a PS4 controller's left and right joystick axis, analog trigger buttons, or pitch and roll gyro data mapped to a motor power and direction range of -100 to 100.

Motor function mapping parameters are:
LX, LY, RX, RY, L2T, R2T, PITCH, ROLL</td>
<td>ps4.Motor(LX);

Data Returned: Integer

Range: -100 to 100</td>
<td>void loop() {
ps4.getPS4();
int x = ps4.Motor(LX);
}

x = the position of the left x-axis joystick mapped to a motor power and direction range of -100 to 100

Tip: Replace LX with the parameter mapped to a motor power and direction range that you wish to return.</td>
</tr>
<tr>
<td>Read Position of Data Mapped to Servo
Returns the position of a PS4 controller's left and right joystick axis, analog trigger buttons, or pitch and roll gyro data mapped to a servo motor position range of 0 to 180 degrees.

Servo function mapping parameters are:
LX, LY, RX, RY, L2T, R2T, PITCH, ROLL</td>
<td>ps4.Servo(RY);

Data Returned: Integer

Range: 0-180</td>
<td>void loop() {
ps4.getPS4();
int y = ps4.Servo(RY);
}

y = the position of the right y-axis joystick mapped to a servo motor position range of 0 to 180 degrees

Tip: Replace RY with the parameter mapped to a servo motor position range that you wish to return.</td>
</tr>
</table>

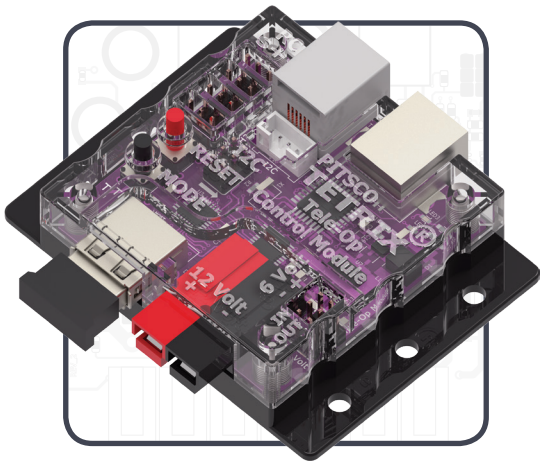
Description	Function	Coding Example
Read Joystick Position Returns the raw analog positional value of a PS4 joystick. By default, the range returned by this function is from 0 to 255 with 128 being the center stick position. The stick function data parameters: LX = Left *x*-axis joystick LY = Left *y*-axis joystick RX = Right *x*-axis joystick RY = Right *y*-axis joystick	**ps4.Stick(LY);** Data Returned: Integer Range: 0-255	void loop() { ps4.getPS4(); int y = ps4.Stick(LY); } *y = the raw analog data value of the right y-axis joystick*
Read Angle Returns the pitch or roll angle of the PS4 controller's internal accelerometer. The angle range returned is from 0 to 360 degrees. The controller will return 180 when controller is held in a flat and level position. This represents the neutral, or center, position.	**ps4.Angle(PITCH);** *or* **ps4.Angle(ROLL);** Data Returned: Integer Range: 0-360	void loop() { ps4.getPS4(); int P = ps4.Angle(PITCH); } *P = the pitch angle of the PS4 controller in degrees*
Set LED State Sets the on, off, and color state of a PS4 controller's light bar. LED data parameters are: RED, BLUE, YELLOW, GREEN, OFF	**ps4.setLED(RED);** Data Returned: None	void loop() { ps4.getPS4(); ps4.setLED(RED); } *Set the PS4 controller LED color to RED.*
Set Rumble Speed Sets the on, off, and speed state of a PS4 controller's internal vibration rumble motors. Rumble motor data parameters are: STOP, SLOW, FAST	**ps4.setRumble(SLOW);** Data Returned: None	void loop() { ps4.getPS4(); ps4.setRumble(SLOW); } *Set the PS4 controller rumble motor to SLOW (low frequency vibration) speed.*
Set Dead Zone Sets the range of PS4 controller joystick axis dead zone. This function is used to create a neutral area, or dead zone, around the center stick position of each joystick's *x*- and *y*-axis. Sometimes when you let go of the left and right joysticks on a gaming controller, they don't always spring back to exact center. The dead zone is a range you can set to be neutral about the center of the joystick's *XY*-axis. If the joystick is within the dead zone range, it is set to be at center. This parameter is used by the **ps4.Motor()** and **ps4.Servo()** functions so that when the joystick is within the dead zone, the value returned by this function is always a zero (motor stop) for DC motors and 90 (center position) for servos.	**ps4.setDeadZone(LEFT, 10);** Data Returned: None	void loop() { ps4.setDeadZone(LEFT, 10); ps4.setDeadZone(RIGHT, 10); } *Set the LEFT and RIGHT joystick x- and y-axis dead zone range to +/- 10.*

General Hardware Specifications

Power:	Use with PRIZM: 12 volts using the TERIX MAX NiMH fuse-protected battery pack Use with PULSE: 6 volts using the TETRIX PRIZM 5-cell NiMH battery pack
LED Indicators:	Blue: Power on Red: Mode indicator (typically used for Bluetooth pairing mode) Green: Module status; blinking = Bluetooth dongle detected; solid = Bluetooth connected
Modular data port:	Data connection to PRIZM controller
I2C data port:	4-pin data port for connection to PULSE controller
USB programming port:	USB port for firmware updates
Battery connection port:	Powerpole type; red/black connectors for connection to PRIZM 12-volt system 3-pin pin style for connection to PULSE 6-volt system
Buttons:	Reset controller: Red button Mode function: Black button; used to pair the Tele-Op module with a Bluetooth device; could also be used as other function depending on device being used
USB host port:	Used for connecting the Bluetooth dongle included with the module. The Bluetooth dongle must support V4.0 + EDR V1.1/2.0/3.0. It is recommended to only use the dongle provided with the Tele-Op module.

TETRIX® Tele-Op Control Module Technical Guide

For use with SONY PS4 DUALSHOCK 4 gaming controller



Call Toll-Free
800•835•0686

Visit Us Online at
Pitsco.com

PITSCO EDUCATION