PixelPusher

Search this site

Motion Control with PixelPusher

Motion control with PixelPusher

Jas Strong <jasmine@heroicrobotics.com>

2014-08-30

Introduction

Since firmware version 1.32, PixelPusher has supported motion control, with broad applications in robotics, animatronics, camera control and lighting. The first motion control device to be supported is the radio control modelling servo, which is available in a bewildering variety of shapes and sizes and is suitable for many applications. Electronic speed controllers (ESCs) for continuous-motion motors also accept the same format of signal, on the same connectors.

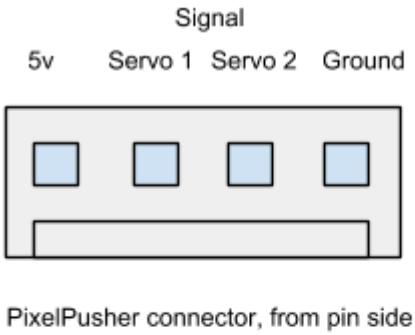
Hardware

In terms of hardware support, PixelPusher is already well adapted to driving the R/C servo. R/C servos usually operate from a 4.8 volt “receiver power” bus, which is close enough to the 5v power usually used by PixelPusher’s LEDs that it can be directly connected without any extra electronics. Interestingly, ESCs often contain a battery eliminator circuit (BEC) that can derive 5v power from a higher voltage battery. In R/C modelling this is usually used to power the radio receiver and the control servos; in our application, it can be used to power the PixelPusher and an attached pocket router.

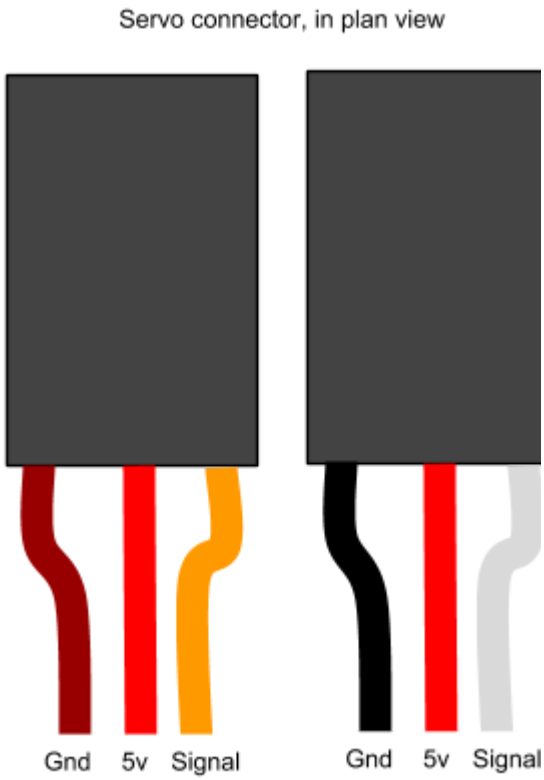
The connector used on standard servos is a three-pin connector with 0.1” spacing. Its pinout is not compatible with the strip connectors on PixelPusher, and an adapter cable must be constructed or the servo cable re-terminated with the appropriate connector. The respective pinouts are given below. Note that servo cables often use different colour schemes; the pinout is always the same.

Each strip port on PixelPusher is capable of driving up to two servos given an appropriate Y-cable.

.

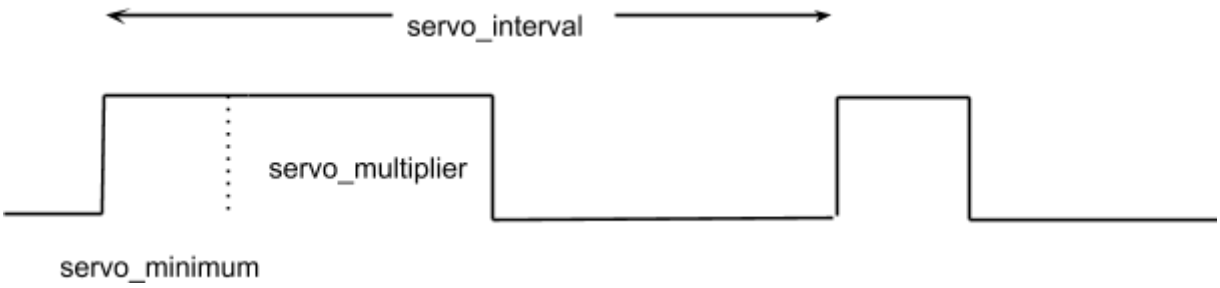


PixelPusher connector, from pin side



Configuration

PixelPusher must be configured for servo drive. The configuration is handled in the same way as other strip configurations; in your pixel.rc file, “servo” is the strip type, and there are some subsidiary options that control the signal synthesizer in order that the pulses may be tuned to your specific servos. How these variables affect the pulses is shown below.



.

strip*n*=servo:

```
strip1=servo
strip2=servo
```

Sets an output port to servo mode. Note that latency on this PixelPusher may increase if servos are enabled. Any and all ports may be configured to servo mode; using more servo ports does not require more system resources over those required for a single servo.

servo_interval:

```
servo_interval=20000
```

This sets the time interval in microseconds between servo pulses. Note that the interval is internally stored in milliseconds.

servo_minimum:

```
servo_minimum=1000
```

This sets the minimum duration of a servo pulse, in microseconds. As per specification, 1000 microseconds is the shortest pulse, but your servos may need different settings.

servo_multiplier:

```
servo_multiplier=4
```

This sets how long the pulse will be for a given position output. The unit is microseconds per pixel value tick. Since pixel values range from 0 to 255, a servo_multiplier of 4 will give a pulse that ranges from 1000 to 2022 microseconds, for a servo_minimum of 1000. These are the default values.

flag*n*=10:

```
flag1=10
flag2=10
```

The hex value 0x10 in the strip flags sets the SFLAG_NOTIDEMPOTENT property for that strip. This is intended for use with motion control devices such as stepper motors where a repeated write of the same value does not result in the same physical state in the device. (If you are using servos, this isn't needed.)

data_watchdog_time:

```
data_watchdog_time=300
```

Since servos may control mechanisms, it is desirable to ensure that they home to a known position in event that the software driving them crashes or the computer running it shuts down. The data watchdog allows PixelPusher to restart itself in the event that no data are received in the specified number of seconds. On restart, PixelPusher will reset the servos.

Software

In terms of software, the library implements two features that allow a client application to determine whether a given strip is actually a motion controller, and if so, what kind of motion controller it is. Firstly, there are the strip flags SFLAG_MOTION and SFLAG_NOTIDEMPOTENT. The SFLAG_MOTION flag is automatically set when a servo port is configured. The SFLAG_NOTIDEMPOTENT flag should be set by the user when an ESC is connected to a servo port (see pixel.rc options above). The library provides methods to query a Strip object to determine whether it is a motion device or idempotent. `Strip.isMotion()` returns `true` for a motion control device. `Strip.isNotIdempotent()` returns `true` for a non-idempotent motion control device.

The second feature that has been added to facilitate motion control is a method to determine when a Strip has actually been sent to the pusher. The method `Strip.getPushedAt()` returns the time in nanoseconds in the Java epoch when the data was actually sent to the PixelPusher. If the strip has not yet been pushed since it was last updated, it returns zero. By subtracting the value returned by `strip.getPushedAt()` from the value returned by `System.nanoTime()` you can determine how long the servo has been moving.

Mapping

For a servo port, the first pixel of the Strip controls the first servo and the second pixel controls the second servo. The first component actually controls each, but since colour

ordering may vary with configuration, it is best to write code that sets all three, as shown in the Processing example below:

```
int pan=127;
int tilt=127;
colorMode(RGB,255);
List<Strip> strips = registry.getStrips();
for (Strip strip : strips) {
      if (strip.isMotion()) {
         // this is a motion controller.
         strip.setPixel(color(pan, pan, pan), 1);
         strip.setPixel(color(tilt, tilt, tilt), 0);
      }
}
```

This will work regardless of order settings.

Example code

Examples will be added to the next release of the library. For now, you can find example sketches here and here.

Comments

You do not have permission to add comments.