Scout32 Servo Control Code & Variables

Initial Servo Setup / Scout32.ino

```
ledcSetup(3, 50, 16); // PWM 3, 50 hz PWM, 16-bit resolution.
ledcAttachPin(ServoPinL, 3); // Attach PWM 3 to GPIO 12
```

A typical hobby servo uses a 50hz signal, so we assign a hardware PWM pin and set that frequency to 50hz. At 50hz, we send out a pulse every 20ms. The width of this pulse is defined as a fraction of 20ms, represented by a 16 bit integer as a fraction of 65536. A hobby servo accepts values of 1ms-2ms at 50hz.

Defining Servo PWM Values / Appserver.cpp

```
int endpoint = 220  ; // Multiplier for endpoints. 60 on modded, 220 TowerPro 360.
int neutral = 4915; // Servo Neutral Point
```

While our pulse width could be set to any between 0ms and 20ms, represented as 0-65536, a hobby servo interprets values from 1ms to 2ms as valid. A pulse width of around 1.5ms is typically interpreted as neutral. To define our neutral value, we need to find out what 1.5ms would be as our integer value.

1.5ms / 20ms = 0.075
65536 * 0.075 = **4915 -** This is where we get our default neutral value.

Endpoints vary from servo to servo. Premade 360 servos and ESCs typically “play nice” with our expected values and use 1ms as a minimum and 2ms as a maximum. This is not true for modded servos! Because standard servos are designed to quickly match a rotational value, they don’t directly interpret signals as an output speed, and usually have a far narrower window of usable values.

How speed and servo direction are calculated at runtime

Initial speed and trim variables are defaults and may be overridden by sliders on the control interface. Speed multiplies the endpoint by a val of 1 to 8. Trim is calculated as a fraction of the endpoint.

```
speed = val * endpoint;
trim = val * endpoint / 16;
```

When writing to our servo, our final speed is calculated as a 16 bit integer. We take the neutral value, add the trim value, and add the calculated speed value. Both Trim and Speed may be negative, depending on the value taken from the interface.

```
ledcWrite(3,(neutral+trim+speed));  // GPIO pin 12 – Add/Sub speed to neutral value
```

For a TowerPro 360 servo using 220 as our endpoint, a full speed forward command would look like:

speed: 8 * 220 = 1760
trim: 0 * 220 / 16 = 0
4915+0+1760 = **6675 <- Value written to servo.**
6675/65536 = .102
.1019 * 20 = **2.04ms <- Value converted to milliseconds.**

Conclusion

You probably only want to change the endpoint values to the ones I already provided. If it's too sensitive, lower the endpoints. If it's not reaching max speed, raise the endpoints. Anything between 50 and 220 should be a sane value to try.

Do not use a neutral value of zero.