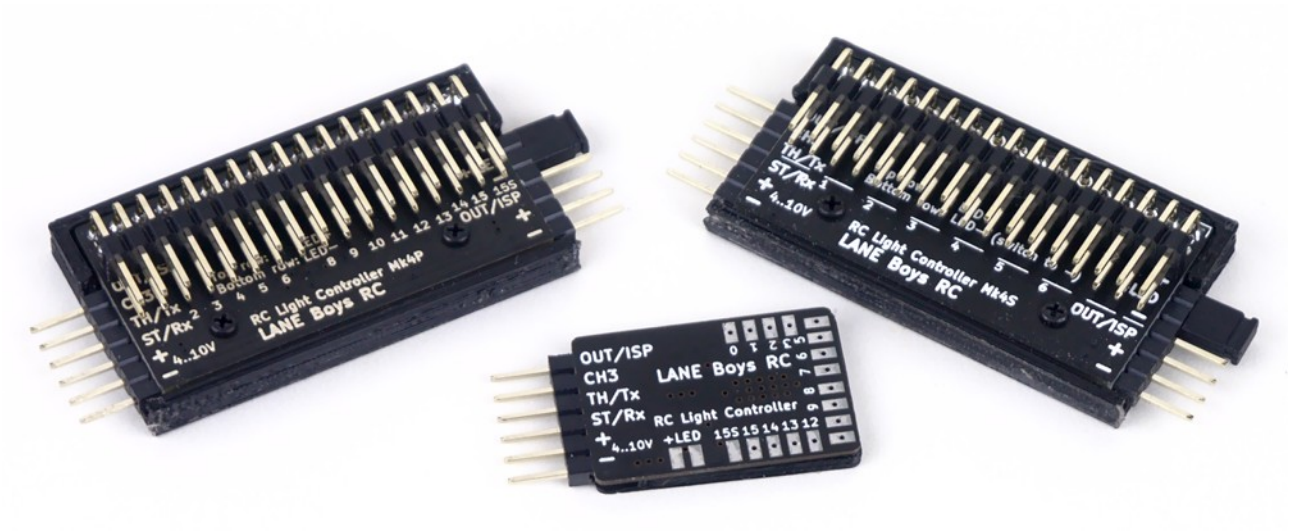
RC Light Controller

Instructions for use

Mk4, Mk4P and Mk4S

Introduction

Thank you for using the LANE Boys RC light controller!



The light controller supports the following features, and more:

- **16 LED outputs**, **constant-current** driven. Up to 20mA per output. Two light controllers can be cascaded for a total of 32 LED outputs.
  (Mk4 and Mk4P only)
- 1 high current non-dimmable **switched output** of up to **2A** to drive a roof light bar
  (Mk4S: 9 non-dimmable switched outputs)
- **Parking**, **Low-beam**, **High-beam** and a **roof light** bar can be **switched on/off manually** from the transmitter
- **Brake** and **Reverse** lights are **automatically** controlled by monitoring the throttle channel. The brake lights now automatically turn on for a short, random time when the throttle goes to neutral.
- **Combined tail and brake light** function in a single LED through controlling the brightness of the LED. (Mk4 and Mk4P only)
- Separate brake light function for a **3rd brake light**
- **Indicators** only come on when you want to. You have to stay in neutral for one seconds, then hold the steering left/right for one second before they engage. This way normal driving does not trigger the indicators.

- **Hazard lights** can be switched on/off from the transmitter
- **Programmable output** designed to drive a **steering wheel** or a figures head, or a **gearbox servo**
- Automatic center and end-point adjustment for steering and throttle channels
- **Light Programs for custom light animations** like police lights, running lights ...
- Simulation of incandescent lights and faulty ground wiring
- Optional Pre-Processor for **easier wiring**
- **Up to 3 AUX channels** can control light functions when using the 5-channel Pre-Processor
- All functions can be **customized using a web browser-based tool**
  https://laneboysrc.github.io/rc-light-controller/
- Hardware and software is **Open Source** under MIT license
  https://github.com/laneboysrc/rc-light-controller/

Light controller variants

There are three different implementations of the Mk4 Light Controller:

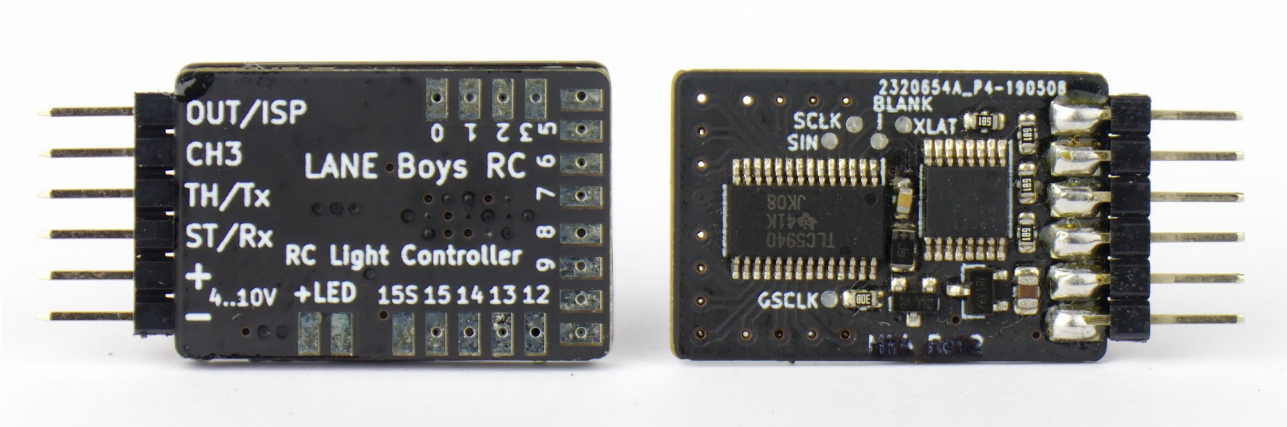
Mk4:	The smallest light controller with solder pads
Mk4P:	Same functions as Mk4, but with a convenient 2.54mm connector for LEDs
Mk4S:	Nine non-dimmable switched light outputs, with a convenient 2.54mm connector for LEDs

The variants are all interchangeable. All of them can be used with the optional Pre-Processor (see below). Any of them can be cascaded if more than 16 light outputs are needed.

For example, you could use an Mk4 or Mk4P to drive indicators, tail/brake and reversing lights of a car, and an additional Mk4S to drive high power light bars, rock lights and headlamps.

Mk4

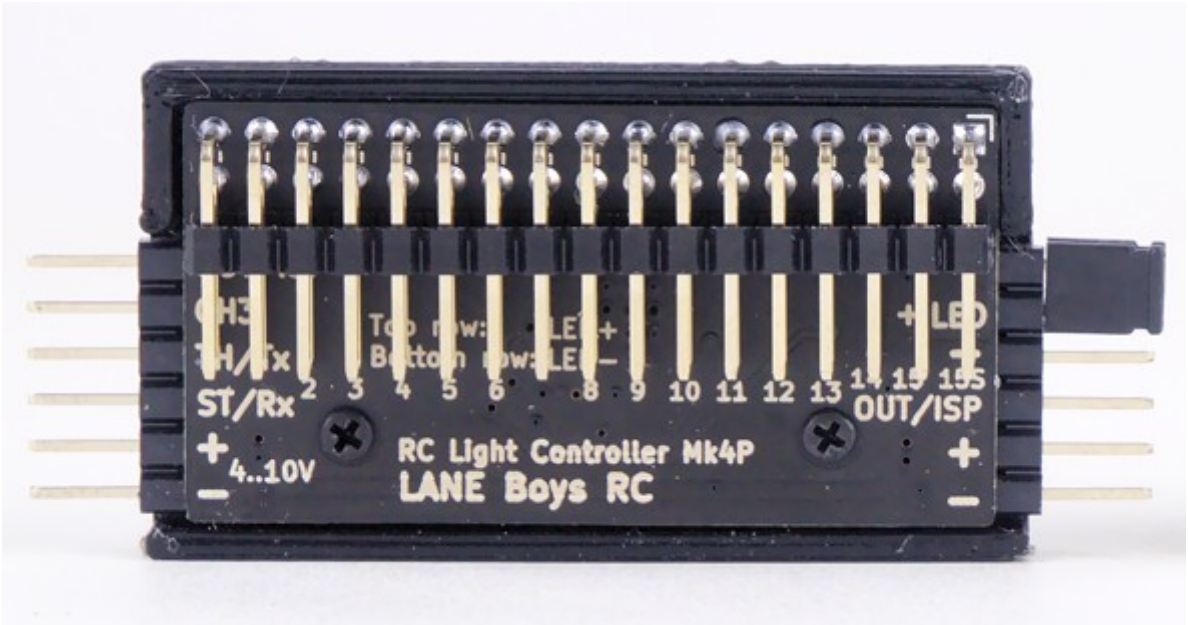
This is the original version of the Mk4 light controller. It has a 6-pin 2.54mm connector for power and servo signals, and solder pads to connect the LEDs.



There are 16 constant-current outputs, each designed to drive a single LED at 20mA; and 1 switching output capable of up to 2A.

Mk4P

The Mk4P is functionally identical to the Mk4, but it has 2.54mm pin headers for easier connections of LEDs. Furthermore, the LEDs can be powered separately, and there are connections to directly hook up a servo or a secondary light controller.



Like the Mk4, the Mk4P has 16 constant-current outputs, each designed to drive a single LED at 20mA; and 1 switching output capable of up to 2A.

Mk4S

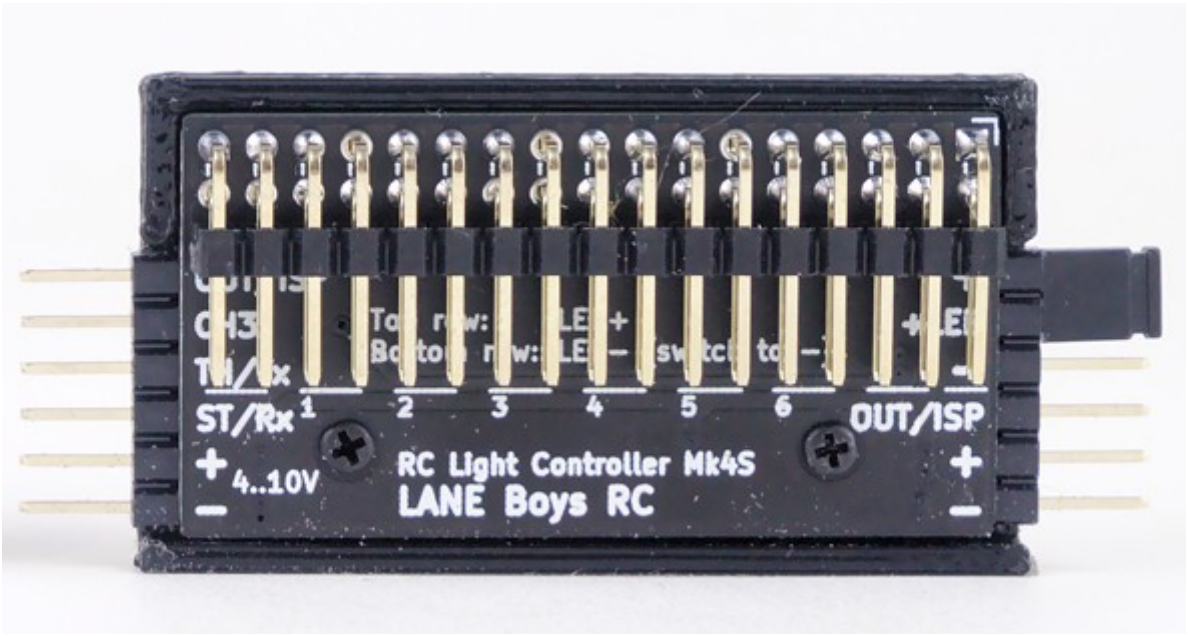
This version of the light controller has 9 non-dimmable switching light outputs, each capable of driving up to 2A.

Since all outputs are switching outputs, you must use current-

limiting resistors for all LEDs connected to the Mk4S.

The Mk4S light controller is not dimmable; LEDs can only be on or off.

Eight of the outputs (OUT0 .. OUT7) are wired to 2 LED connections each, and OUT8 is wired to a single LED connection.



The Mk4S light controller only has only 9 outputs: OUT0 .. OUT8. Any configuration for other outputs (OUT9 .. OUT15) will be processed internally but are not accessible.

Light Controller connections

The table below shows how the outputs are configured by default.

All output functions can be fully customized, refer to a later section in this document on how to adapt the configuration to your vehicle.

Name	Function	Description
+	Positive power supply	+4..10V input voltage, usually taken from the receiver (red wire)
-	Negative power supply (GND)	Supply ground (brown or black wire)
ST/Rx	Steering input	Steering servo signal from the receiver, or input from the Pre-Processor
TH/Tx	Throttle input	Throttle (ESC) signal from the receiver. In case the Pre-Processor is used, this pin can also serves as configurable output similar to OUT/ISP

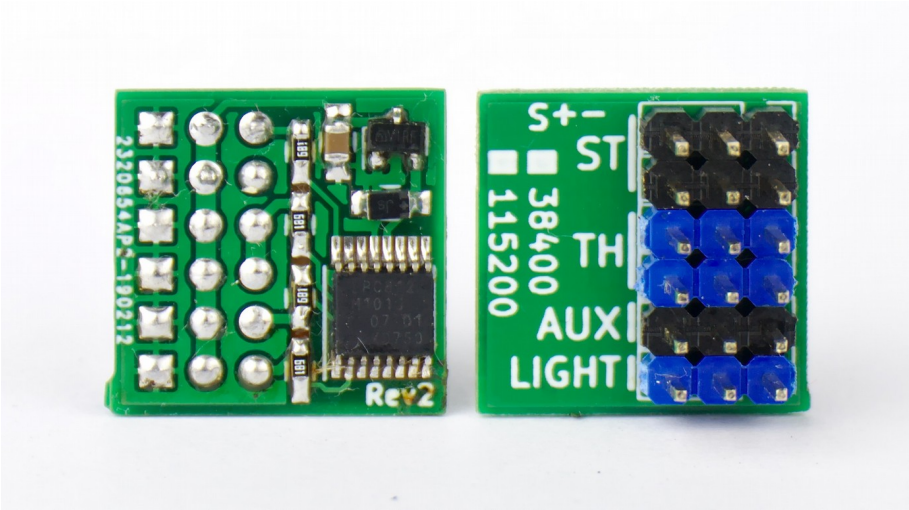
Name	Function	Description
CH3	CH3/AUX input	Channel 3 (AUX) signal from the receiver
OUT/ISP	Output	Output to drive a servo or connect a slave controller. The output function can be configured via the C*onfigurator* tool, see below.
+LED	LED supply	Terminal to connect the (+) Anode of the LEDs.
OUT0	Light output 0	Parking/Position light front left
OUT1	…	Parking/Position light front right
OUT2	…	Main beam front left
OUT3	…	Main beam front right
OUT4	…	High beam front left
OUT5	…	High beam front right
OUT6	…	Indicator front left
OUT7	…	Indicator front right
OUT8	…	Tail/Brake light rear left
OUT9	…	Tail/Brake light rear right
OUT10	…	Reversing light rear left
OUT11	…	Reversing light rear right
OUT12	…	Indicator rear left
OUT13	…	Indicator rear right
OUT14	…	3rd brake light rear
OUT15	Light output 15	Roof light
OUT15S	Switched light output (left of light output 15)	Carries the same signal as light output 15, but instead of current-controlled it is switched towards Ground (-). Up to 2A of current can be switched....

Connecting the light controller to your RC car

The light controller is usually powered from the RC receiver. The following sections describe different methods how to route the servo signal from the RC receiver to the light controller and the other RC components.

Using a Pre-Processor

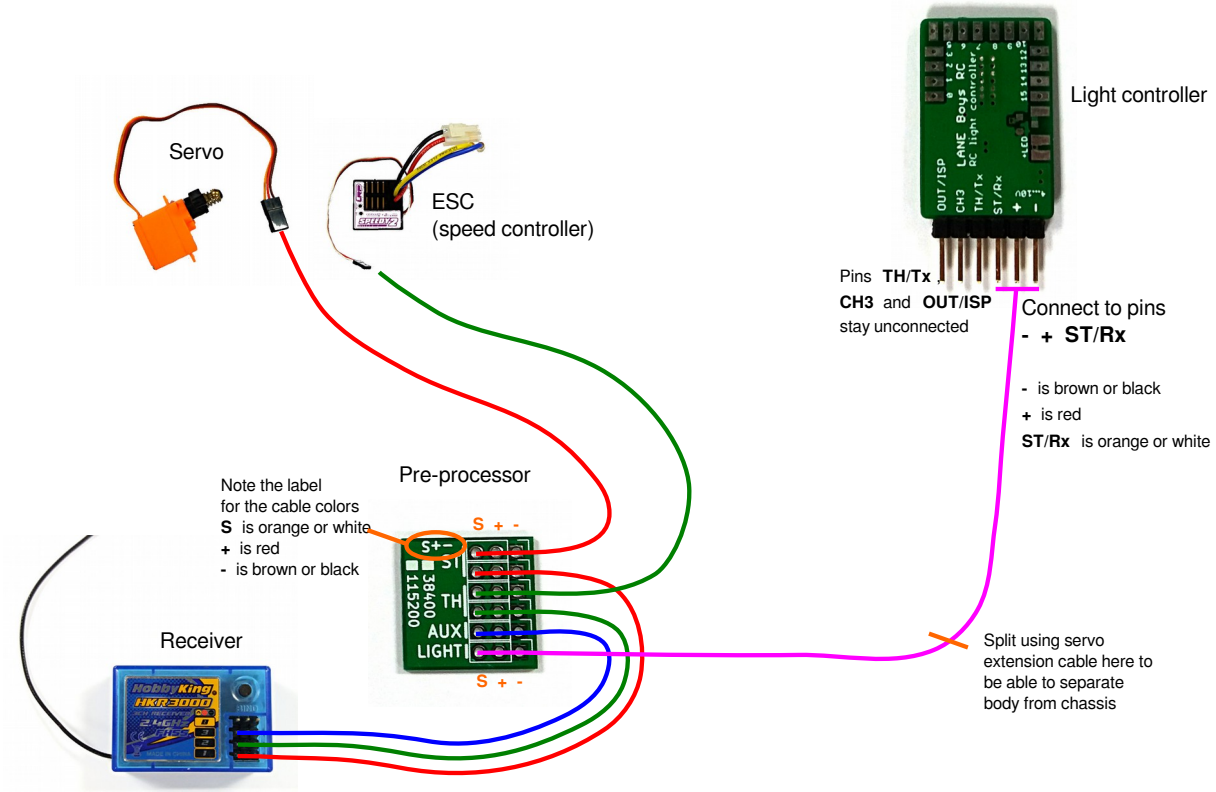
The Light Controller has a small companion circuit board called Pre-Processor.



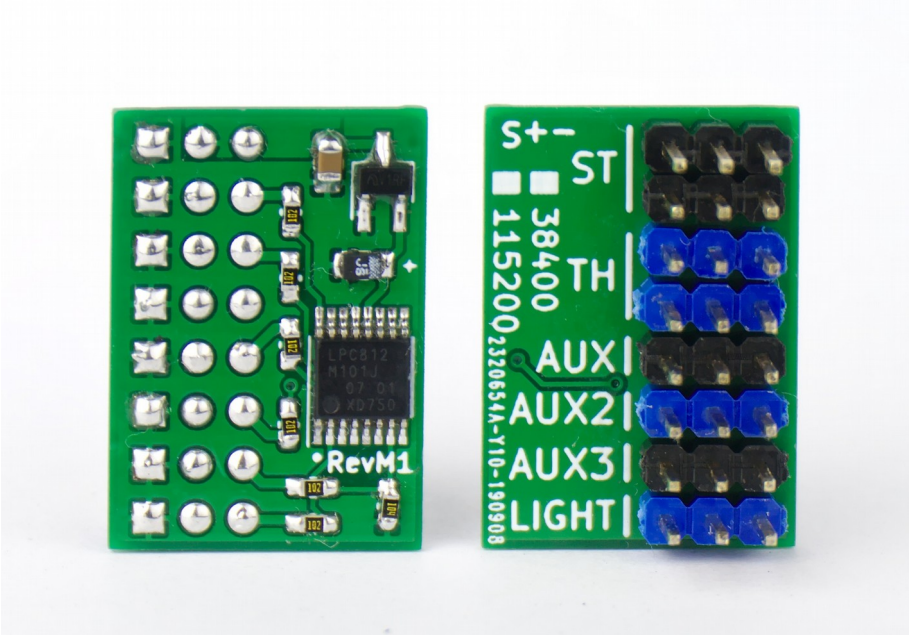
3-channel Pre-Processor

Instead of connecting the servo signals to the light controller directly, connect them to the Pre-Processor instead. The Pre-Processor accumulates all servo signals into a single data line to the light controller. The Pre-Processor also has additional connectors for the steering and throttle signal, saving the hassle of Y-cables.

This simplifies wiring drastically: only a 3-pole servo extension cable is needed between the chassis of the car (where receiver, ESC, Servo and Pre-Processor reside) and the body shell (where the light controller and the LEDs are).



Pre-Processor also comes in a 5-channel variant, which is useful if you have a modern RC car radio that supports more than 3 channels. With the 5-channel Pre-Processor you can use separate AUX channels for manual indicators, or switching the lights, or turning the hazard lights on an off. The function of each AUX channel can be configured separately.

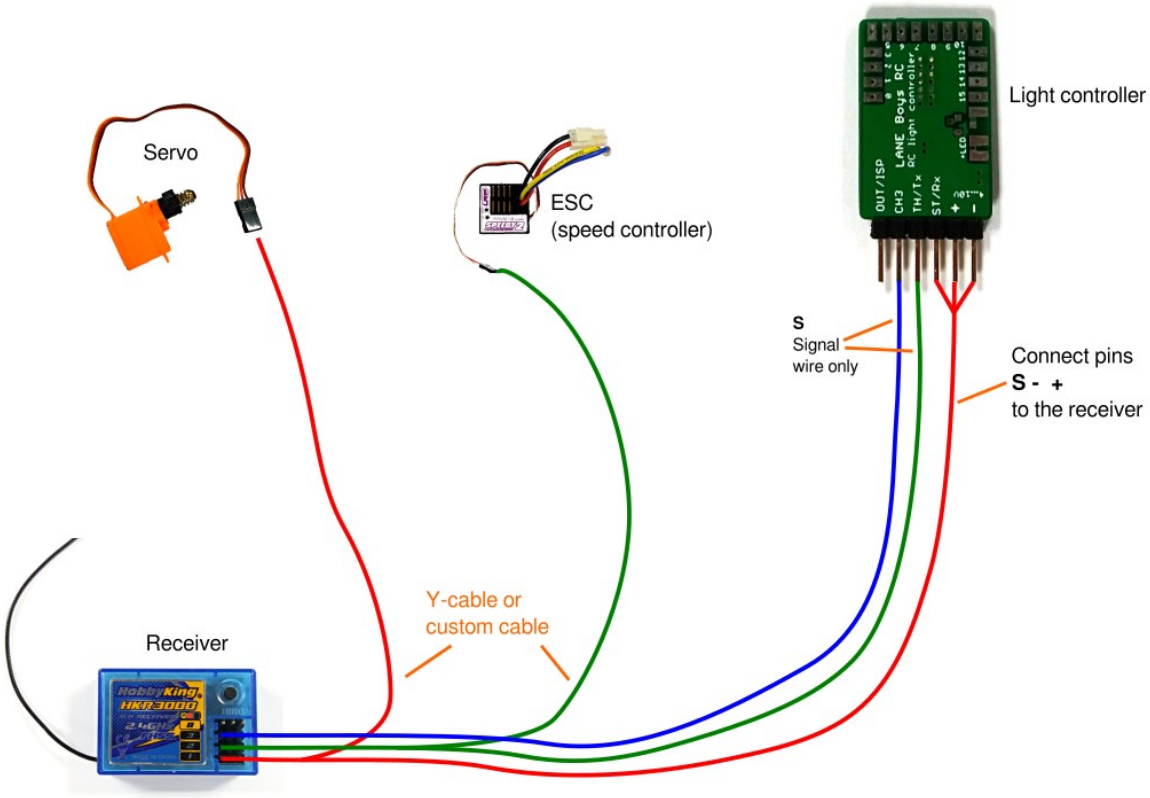


5-channel Pre-Processor

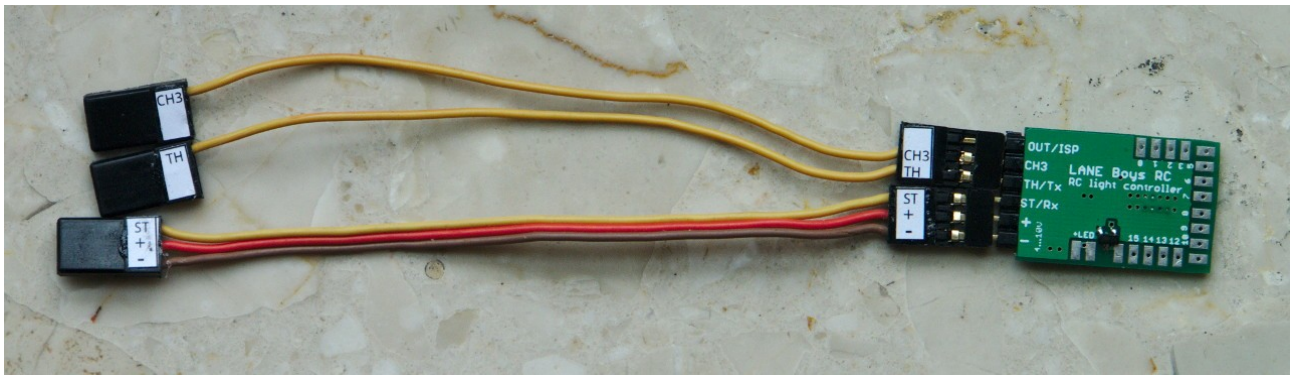
Direct connection to the servo signals

Connect the light controller signals ST/Rx, TH/Tx and CH3 to the steering, throttle and auxiliary channels of your receiver.

For steering and throttle you will need to utilize a Y-cable as shown on the schematics below:



Only one of the channels needs to provide power to the light controller. For the other channels it is sufficient to use just the signal wire (usually orange or white in color). Servo extension leads can be easily reconfigured to achieve this:

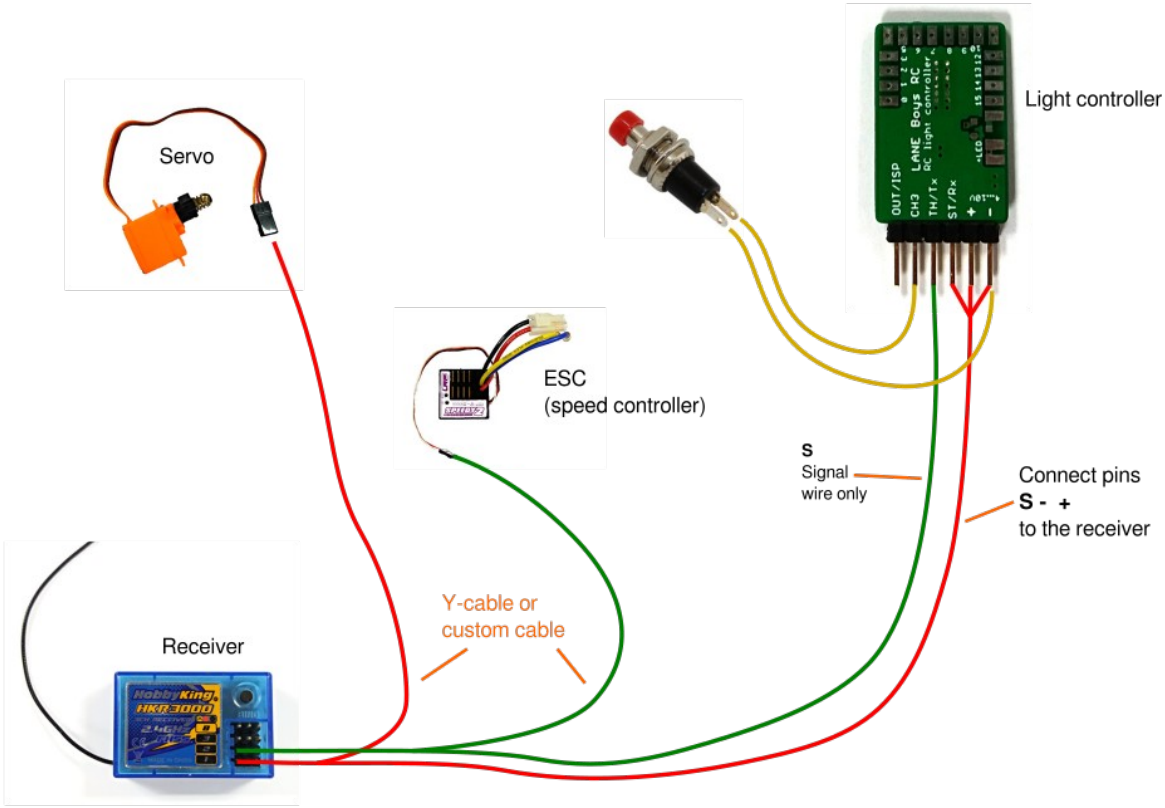


Using a push-button installed in the car to switch functions

If you have a two-channel RC transmitter, or don't have a spare channel to control the light functions, then you can alternatively hook up a push-button to the CH3 input directly to the light controller. You can hide the push-button somewhere in the car and access it when needed.

The push-button needs to connect the **CH3** input of the light controller to **−** (Receiver supply minus; GND) when pushed, as shown in the diagram below.

Note that you must enable the **Push button on the light controller** setting in the ***Configurator* tool** described later in this document to enable this function.



Connecting two light controllers for a total of 32 LED outputs

For large installations, where 16 LED outputs are not sufficient, two light controllers can be connected to control a total number of 32 independent LED outputs.

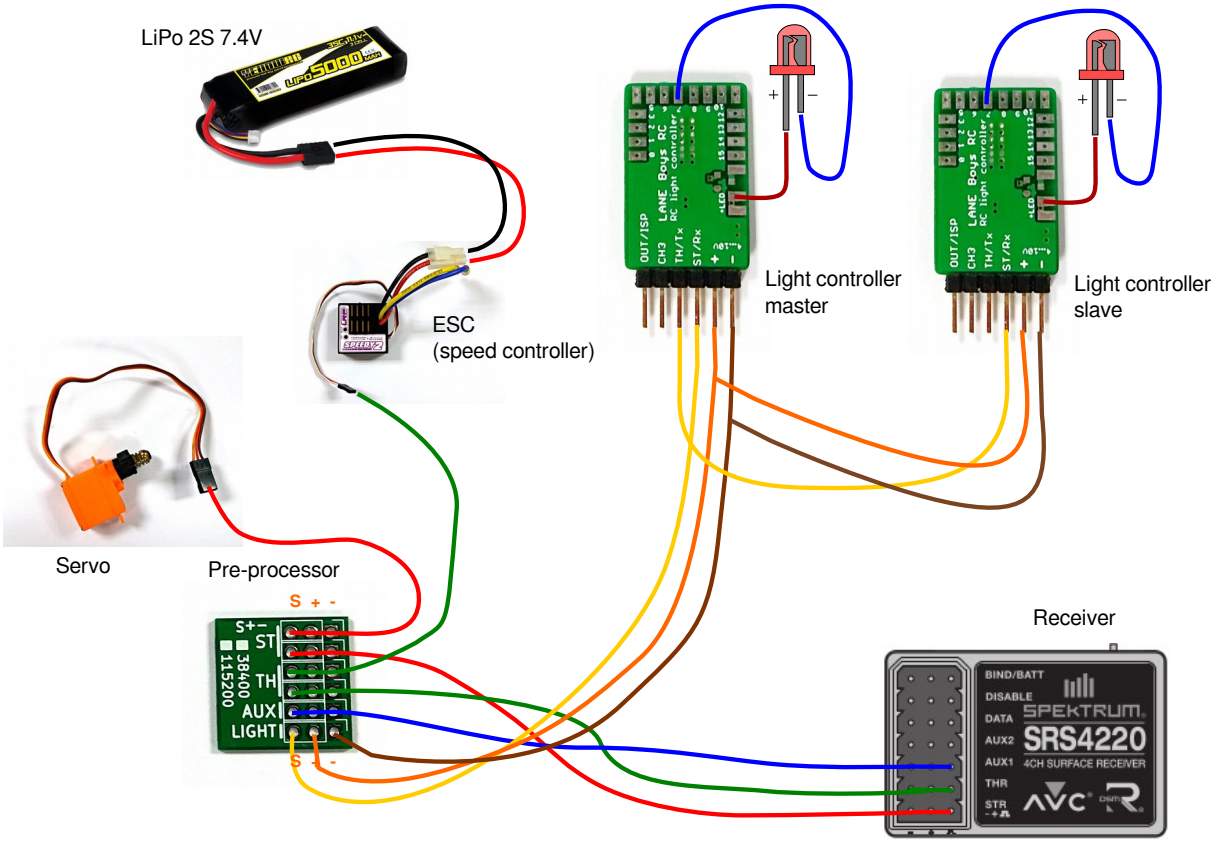
For traditional reasons this is called Master/Slave system. The Slave light controller contains a generic configuration that is independent of the vehicle.

The Master light controller gets configured with the Slave output enabled on OUT/ISP (or TH/Tx) and holds the configuration for all 32 LEDs. This is convenient, as only one light controller needs to be programmed when adjustments to the configuration are necessary.

> ***Any of the light controller variants (Mk4, Mk4P and Mk4S) can be used as either Master or Slave. You can mix-and-match as required.***
>
> ***Using Mk4P and Mk4S as Master can be convenient as they come with a separate 3-pin output connector that accepts a servo extension cable, no custom cable required.***

A simple 3-pole servo extension wire is sufficient as connection between the Master and

Slave light controller. This allows the Slave light controller to be placed close to where the LEDs are, or even in a trailer.



When creating a configuration for Master and Slave, the Baudrate needs to be set to the same value.

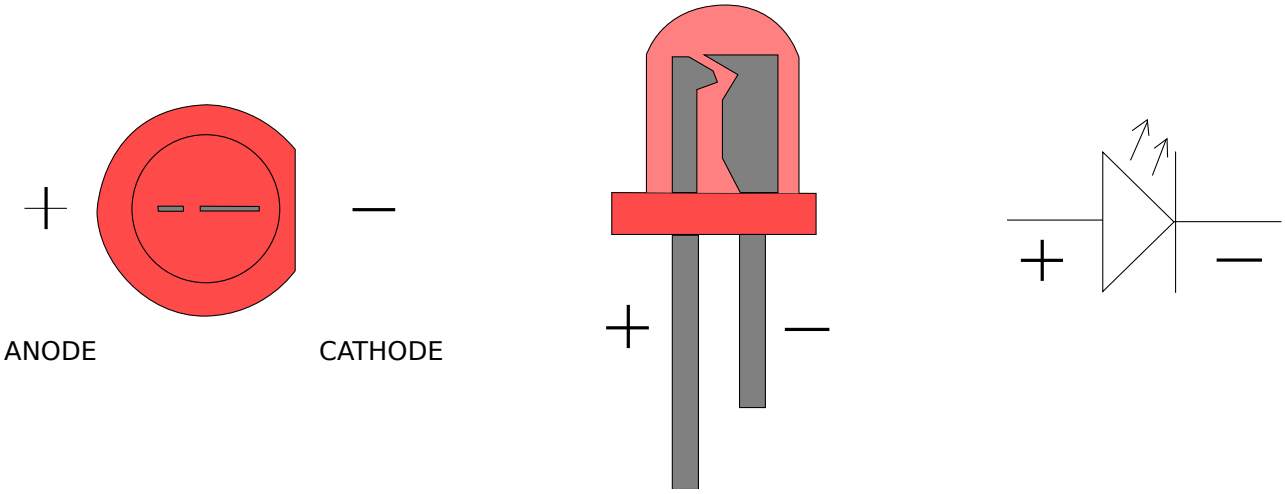
Some users tried to install a second receiver, battery and light controller in the trailer. This is not recommended due to the following reasons:

- The timing reference in the light controller is not precise and drifts quickly. This means that indicators don't light up in sync on the car and the trailer
- Some timings like automatic reversing or brake lights are created randomly. Therefore the car and trailer brake and reversing lights will not be in sync.

None of these problem exists when using the Master/Slave system, as all timings are generated by the Master only.

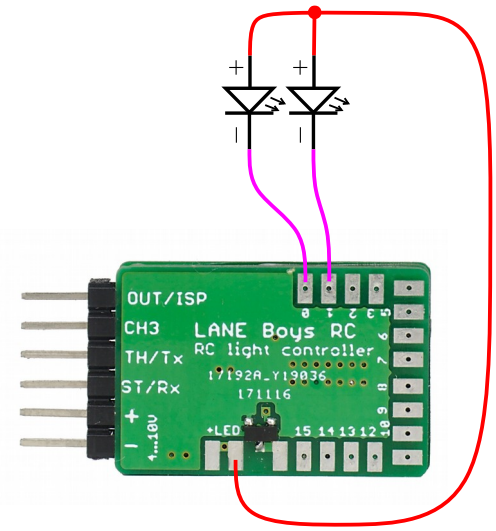
Connecting LEDs

The (+) Anode (usually the long pin) connections of all LEDs are connected together to the terminals **+LED**.



Constant-current controlled LED outputs

Each light output on the Mk4 and Mk4P light controller is designed to drive a single LED. The outputs are driven with a constant current of up to 20mA for optimal LED performance and uniform brightness.



Do not connect a resistor in line with the LED on Mk4 and Mk4P, it is not necessary for constant-current controlled outputs!

The outputs are designed to drive standard 3mm or 5mm (or SMD) LEDs at 20mA, they are not suitable for high-power LEDs.

For more information about LEDs please refer to Appendix A: More information about LED connections on page 23

Switched LED output OUT15S, and Mk4S

Beside the 16 current-regulated light outputs, the Mk4 and Mk4P light controller also have one non-dimmable, high current capable, switched light output. It is designed to drive an off-the-shelf light bar or high power LEDs.

The non-dimmable switched light output is marked **15S** and turns on together with output OUT15. When OUT15 is turned on *at any brightness*, OUT15S is also turned on.

On the Mk4S (Switching) light controller, all outputs are such non-dimmable switched outputs.

Every switched output can sink up to 2 Ampere in current.

> ***For the switched outputs you must use a current-limiting resistor in series with the LED.***
>
> ***Without resistor the LED will burn out immediately.***

Output functions

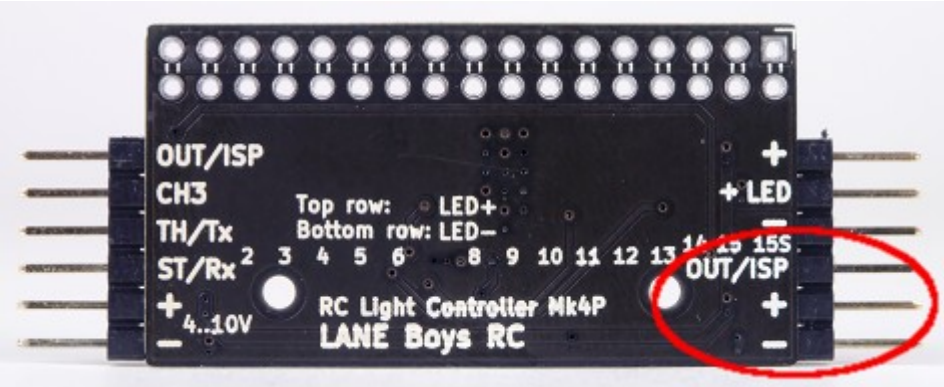
The light controller can provide auxiliary functions on its OUT/ISP pin. When using a Pre-processor, an second auxiliary function can be configured to the TH/Tx pin.

The following auxiliary functions are available:

- Servo output that follows the steering channel
- Gearbox servo output
- Servo output under control of light programs
- Slave light controller output
- Pre-Processor output

These auxiliary functions can be split into two types: **Servo functions** and **UART functions** (serial communication).
The light controller supports at most one Servo function and one UART function at a given time.

The Mk4P and Mk4S light controller have a 6-pin connector on the right side. The lower 3 pins follow the standard servo connector layout (Minus, Plus, Signal). The signal pin is connected to OUT/ISP. This makes it very convenient to connect a servo, or a Slave light controller.



Servo output that follows the steering channel

The light controller can drive a small servo in synchronization with the steering input. This is designed to turn the steering wheel in the cabin of a truck. The center, endpoints and direction can be fully programmed (see below).

> ***For safety reasons do not connect the steering servo of your vehicle to this output!***

The signal wire of the servo must be connected to the configured output pin (OUT/ISP or TH/Tx) on the light controller. Note that some servos, especially analog servos, put a high load on the OUT/ISP pin, preventing the light controller to start. Digital servos usually work without issue.

Gearbox servo output

The servo output can also be configured for controlling a 2-speed or 3-speed gearbox. When the gearbox control is enabled, 1-click and 2-clicks on CH3/AUX change their behavior:

2-speed gearbox: 1-click changes to gear 1, 2-clicks to gear 2.

3-speed gearbox: 1-click changes to the next higher gear, 2-clicks changes to the next lower gear.

Since 1-click and 2-clicks are now taken up by gearbox control, it is no longer possible to switch between the different virtual light switch positions. The lights can only be switched on/off using 3-clicks. If this is not desired, then one can opt to control the gearbox from a light program. With a small light program it is possible to toggle through the gears when performing 6-clicks on CH3/AUX.

Servo output under control of light programs

It is possible to control the servo output from a light program. When the respective configuration option is chosen, values between -100 (left endpoint), 0 (center) and +100 (right endpoint) can be set to move the servo to desired positions. This can be useful for simple mechatronics animations, like random movements of a figures head.

Servo setup

The center point and end points of the servo output can be configured independently of the car's steering.

To set up the servo output, perform **eight** clicks on channel CH3. The left indicators will light up and the steering wheel servo will follow directly the steering input on the transmitter. Move the steering wheel on the transmitter to the position that shall be used for full left on the steering wheel servo.

Note that this may require that you need to turn the steering wheel on the transmitter *right*, if the servo is reversed. Don't worry, this is correct and lets the light controller know which way to turn the steering wheel servo. Hold the transmitter steering in position while clicking channel CH3 once.

Now both left and right indicators will turn on. Turn the steering wheel on the transmitter to the position that the steering wheel servo is centered and click channel CH3 once. Now the right indicators will turn on. Perform the same procedure as for the left end point and click channel CH3 once.

The center and end points should now be correct, the steering wheel should follow in the same way as the car. The settings are stored persistently and are not affected by steering trims and endpoint adjustments.

Note that after changing endpoints or trim on your RC transmitter, you need to restart the light controller so that it learns the new settings.

Slave light controller output

When choosing this option, a secondary Slave light controller can be connected so that a total of 32 independently controlled light outputs are available.

Pre-Processor output

This setting is usually not useful for end-user configuration, it is internally used by the Pre-Processor.

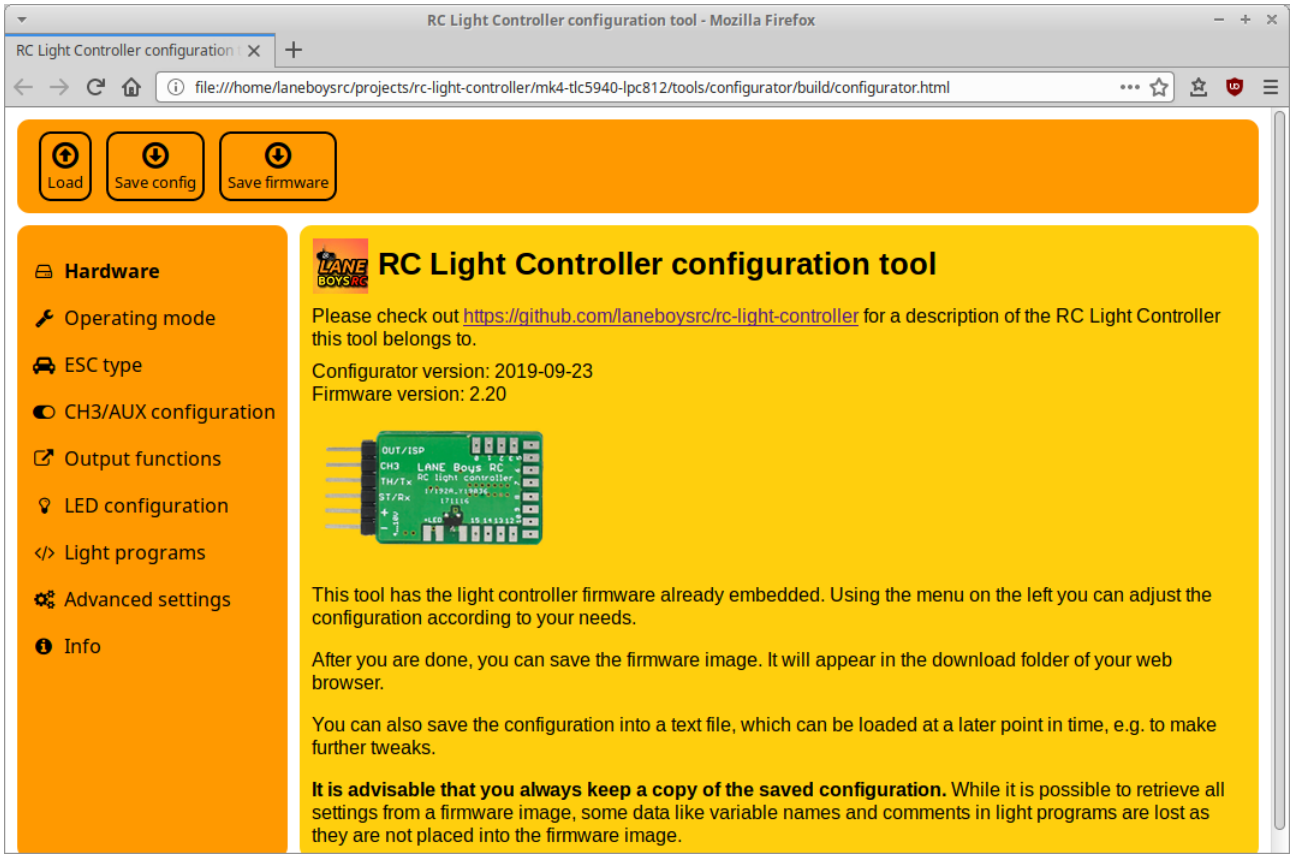
When enabled, the Steering, Throttle and AUX signals on the input of the light controller are output using a serial protocol. The protocol is documented here:
https://github.com/laneboysrc/rc-light-controller/blob/master/doc/preprocessor-protocol.md

Configurator: Custom configurations

Every car is different. To adjust the light controller behavior for your particular vehicle, create a custom configuration by using a tool called ***Configurator***.

The *Configurator* runs in a modern web browser. It has been tested with Mozilla Firefox, Google Chrome and Microsoft Edge.

You can access the latest version of the *Configurator* on-line at
https://laneboysrc.github.io/rc-light-controller/



Alternatively, you can download everything needed to program the light controller from here – which is useful if you want to configure and program the light controller without requiring Internet access:

https://github.com/laneboysrc/rc-light-controller/blob/master/mk4-tlc5940-lpc812/mk4-download-me.zip?raw=true

This archive contains the *Configurator* (file name ***configurator.html***) and the programming tool described below.

To learn how the configuration process works, please watch the following video (starting at 14:09):

https://youtu.be/-VyNAVU3-ok?t=849

The *Configurator* has the firmware already embedded, just set the options you want and click on the ***Save firmware image...*** button. A file named ***light_controller.hex*** will be stored in the ***Download folder*** of your web browser.

Flashing the firmware

The light controller is shipped with a default firmware, with the light configuration described in section Light Controller connections on page 4.

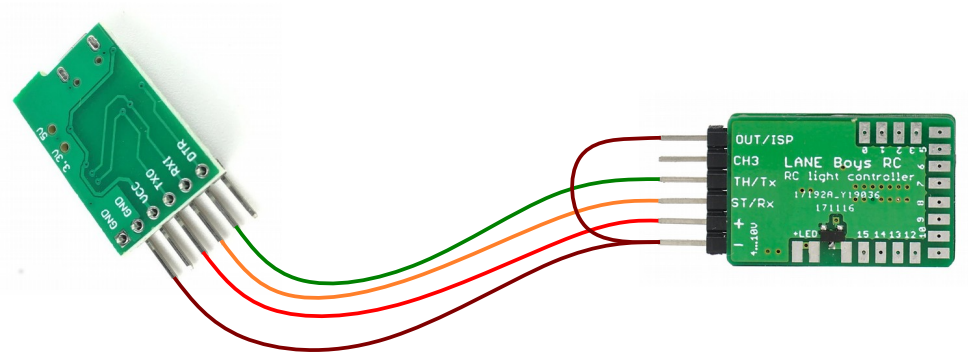
However, most installations will require changes to the configuration, as described in the previous section. After creating such custom configuration, you can use the **LPC81x-ISP**

tool to flash the firmware. The tool can be downloaded here:

https://laneboysrc.github.io/rc-light-controller/lpc81x-isp-windows-64bit.zip

The tool is also included in the **mk4-download-me.zip** archive linked above.

You need a USB-to-serial adapter to connect your PC to the light controller. Wire-up the USB-to-serial adapter as follows:



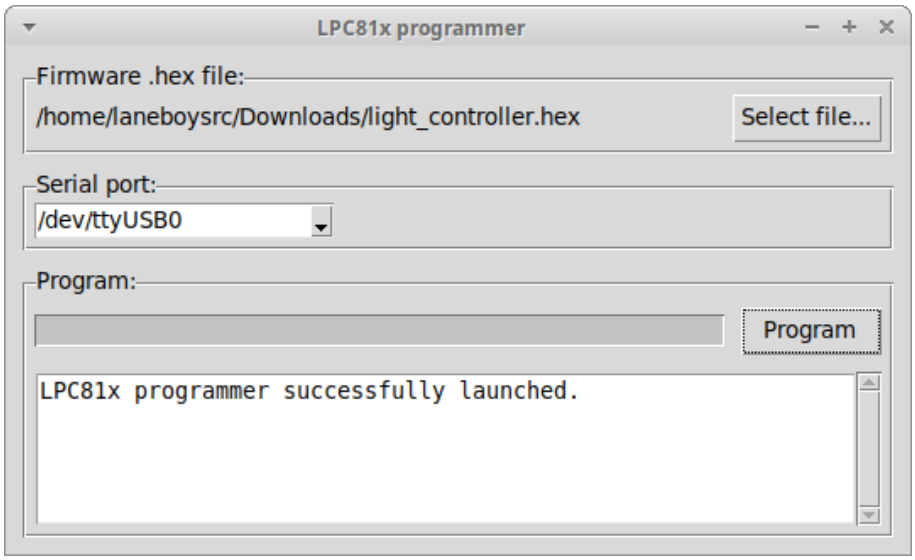
USB-to-Serial		Light Controller
GND (–)	↔	**–** (power supply)
VCC (+)	↔	**+** (power supply)
TX	↔	ST/Rx
RX	↔	TH/Tx
GND	↔	OUT/ISP

Note that the TX (transmit) line from the USB-to-Serial adapter connects to the RX (receive) line on the Light Controller, and vice-versa.

Connect the light controller to the USB-to-serial adapter before plugging the USB-to-serial adapter into the PC.

This is necessary as OUT/ISP pin must be connected to (-) GND while power-on to allow the light controller to enter the programming function.

Start the **LPC81x-ISP tool**, click on the ***Select file…*** button and load the ***light_controller.hex*** file that you saved in the download folder earlier.



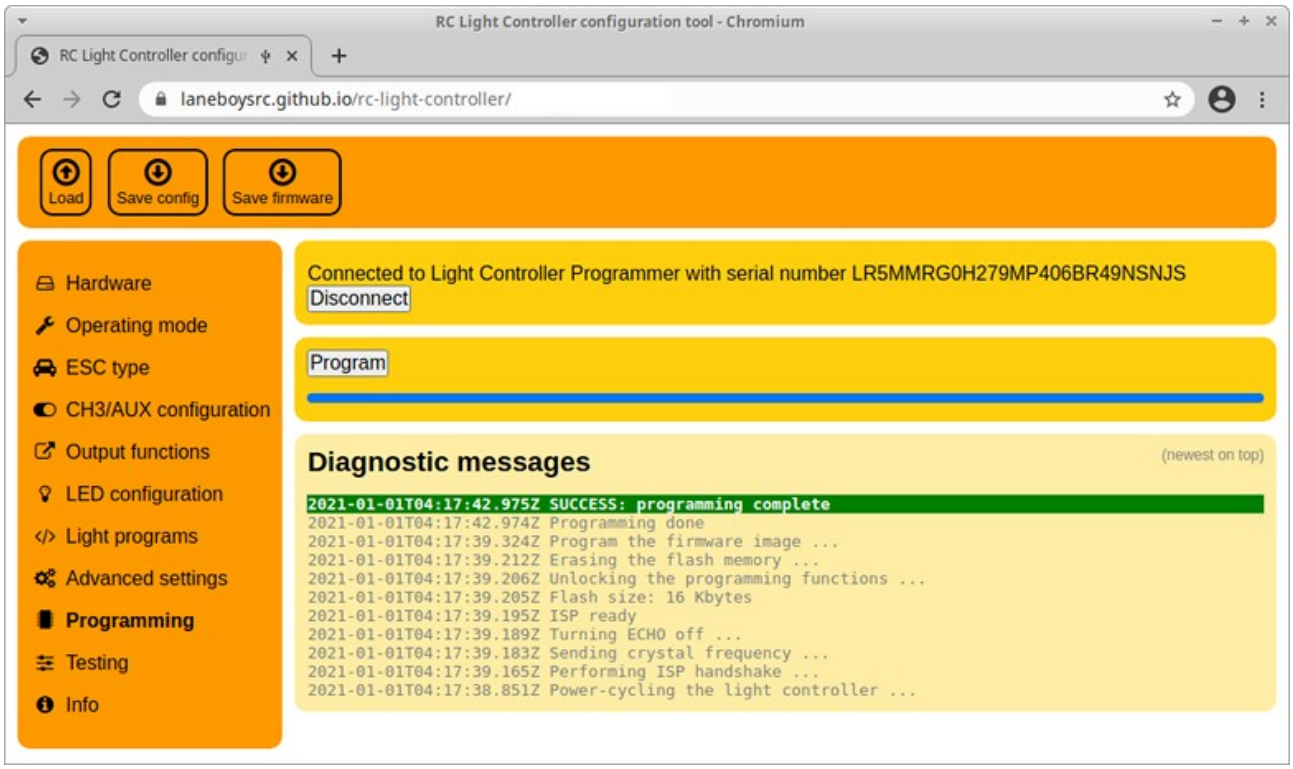
Click the ***Program*** button to start the flashing of the new firmware containing your custom configuration.

Easier firmware flashing with the WebUSB programmer

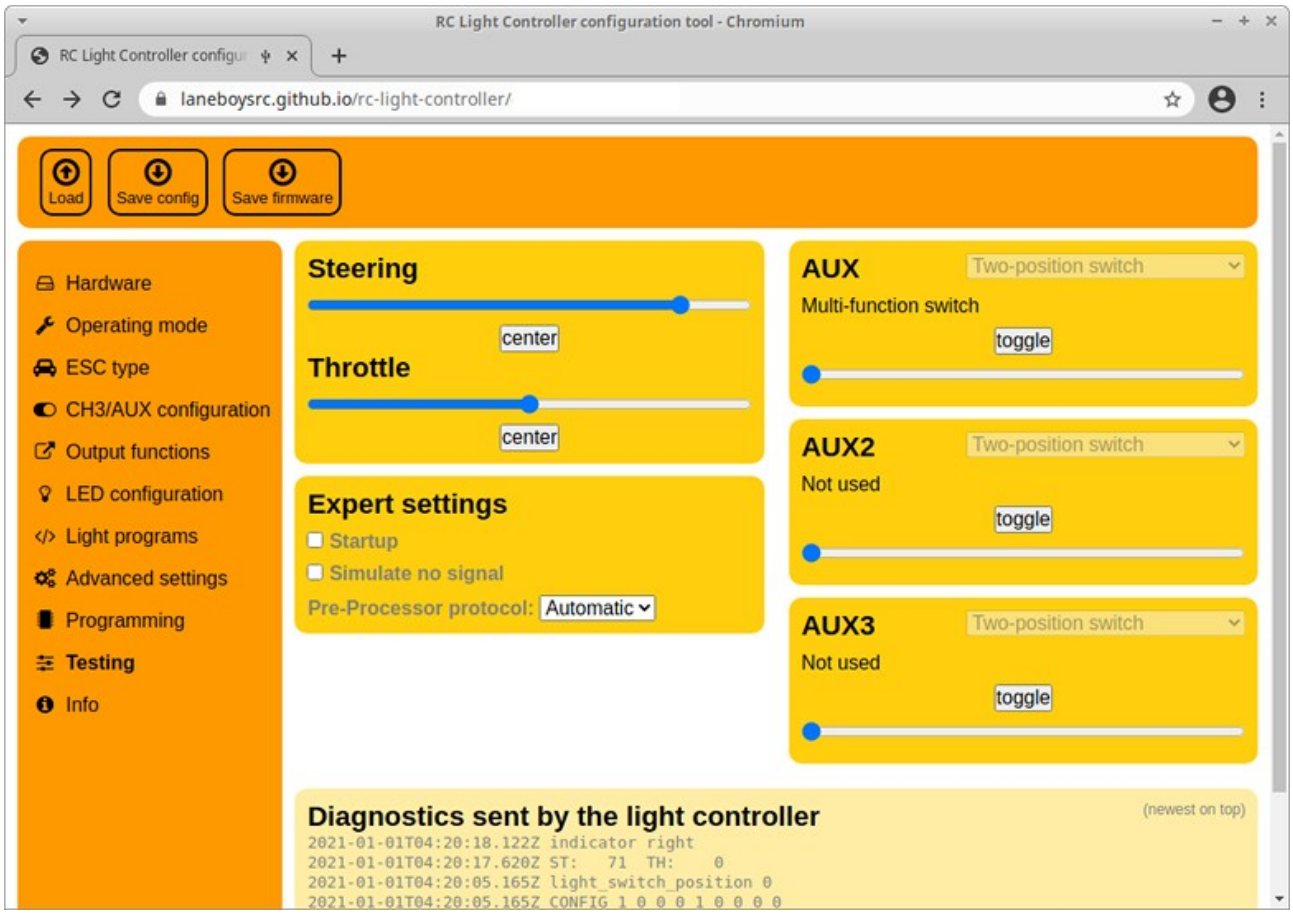
As alternative to the USB-to-serial adapter, we offer a custom programmer hardware that allows flashing and testing the light controller firmware direct from within the *Configurator* tool.

The WebUSB programmer is especially useful for Apple Mac users, as the LCP812_ISP tool is only available as Python source code, requiring in-depth computer skills to get working.
The WebUSB programmer, however, works, for example, with the Google Chrome browser. Note that it does not work in *Safari*, as Apple did not implement the WebUSB standard.

This hardware is custom made for the light controller and therefore more expensive than an off-the-shelf USB-to-serial adapter.



A web browser with the Configurator running after a successful programming operation



The Testing tab simulates a RC system including Pre-Processor, so that the light configuration can be tested without disconnecting from the computer.

Attention: the Testing function only works for configurations with Pre-Processor operation. If you intend to use direct servo connections to the light controller, you can temporarily switch to Pre-Processor usage for testing and then change back to Servo inputs when done.

The WebUSB programmer requires a web browser that supports WebUSB technology. An up-to-date list of compatible browsers can be found here:

https://caniuse.com/?search=webusb

Operating the Light Controller

After power-on, the front main beam LEDs will turn on for about two seconds. During this time ensure that steering and throttle controls on the transmitter are kept in neutral position.

If if the front indicators light up instead of the main beam LEDs, this means that none of the input channels ST, TH and CH3 receive a proper servo signals from the receiver. Turn off the power and check all connections.

After initialization, all lights turn off and the system is ready to use.

It is advisable to let the light controller know the endpoints of steering and throttle before driving off. To do this, simply move the steering fully left, then fully right. Hold the car in the

air safely and pull the throttle full forward, then full backward – ensuring that the drive train does not get damaged.

Brake and reversing light

When driving, the brake and reverse lights will now come on according to the throttle input. The indicators can be engaged by keeping both steering and throttle neutral for one second, then turning the steering into the direction the indicators should show. Once they are engaged, you can start driving off with the indicators operating. They will turn off after a short delay when the car goes straight, or immediately when the car turns in the other direction.

Switching the main lights

Several functions of the light controller can be operated manually through the auxiliary channel CH3 (AUX). Since there are quite a few functions, the concept of multiple “clicks” is employed. Similar to the operation of a computer mouse, one can switch the CH3 (AUX) switch on the transmitter repeatedly within a short time to invoke different functions.

One click: Turn on more lights at each click: Parking, Main beam, High beam, Roof lights

Two clicks: Turn the lights down; reverse of one click

Three clicks: toggle all lights on and off

Four clicks: toggle hazard lights (all indicators flash) on and off

Here is a video showing the operation: https://youtu.be/-VyNAVU3-ok

The light controller supports various switch types on the transmitter side. Please refer to Appendix F: Determining the AUX switch type for details.

Steering and Throttle channel reversing

The direction of the steering and throttle channels can be programmed to match the car. To do this, perform **seven** clicks on channel CH3. The front and rear indicators on one side as well as the front main beam lights will turn on.

Move the steering wheel on the transmitter into the direction of the indicators that light up (i.e. if the indicators on the left side of the car light up, turn the steering wheel left). When successful the indicators will turn off.

Now engage the throttle forward. When successful the main lights will turn off, programming channel reversing has finished and the light controller will resume normal operation.

The steering and throttle direction are stored persistently so this configuration has to be carried out only once after installing the light controller.

Light Programs

Light Programs are simple scripts that allow full customization of light controller functions and light outputs without the need of developing a custom firmware.

Detailed documentation of Light Programs can be found at https://github.com/laneboysrc/rc-light-controller/blob/master/mk4-tlc5940-lpc812/doc/light-programs.md

Light programs are useful to create running lights, police flashers etc. A collection of light programs can be found at https://github.com/laneboysrc/rc-light-controller/tree/master/mk4-tlc5940-lpc812/configurations

Technical data

Parameter	Mk4	Mk4P	Mk4S
Operating voltage	**4V .. 10V**		
Constant current outputs 20mA per LED output	**16**	**16**	–
Dimming	**63 steps** DC, no PWM	**63 steps** DC, no PWM	(no dimming)
Switched output 2A max per output	**1**	**1**	**9** 8 outputs with two 2-pin connectors each, 1 output with one 2-pin connector
LED configuration	**Common Anode** Common Plus pole; Minus pole goes to individual outputs of the light controller		
Dimensions	36 × 19 × 3mm	62 × 27.5 × 9.5mm	62 × 27.5 × 9.5mm
Dimensions (without connectors and case)	27 × 18 × 2.5mm	44 × 23 × 3mm	44 × 23 × 3mm

Pre-Processor: 17 × 17 mm
5-channel Pre-Processor: 24 × 17 mm

Have fun with RC! Werner

laneboysrc@gmail.com

https://laneboysrc.blogspot.com/

https://www.youtube.com/user/laneboysrc

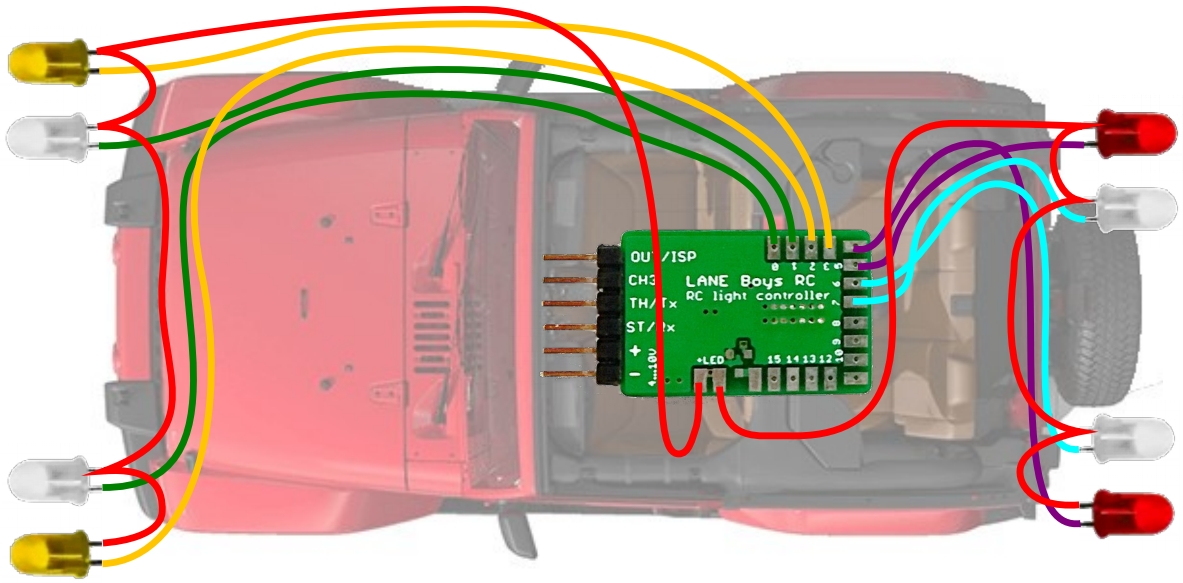
https://www.flickr.com/photos/78037110@N03/albums/

https://github.com/laneboysrc/rc-light-controller/

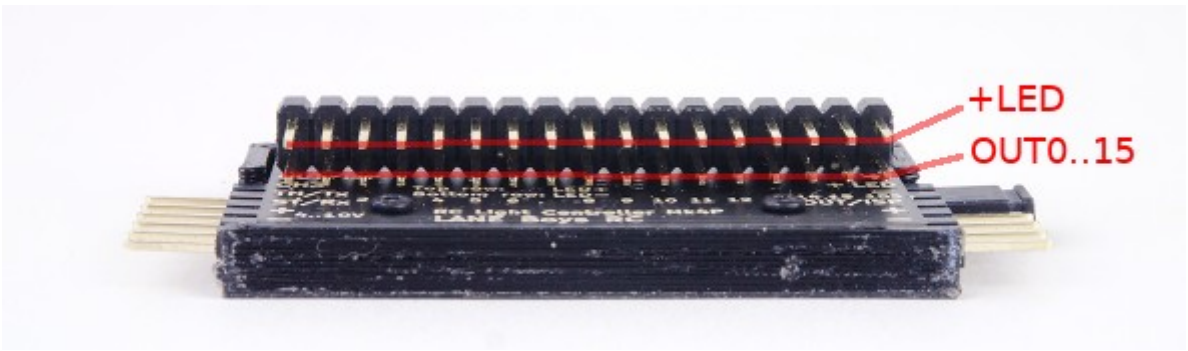
Appendix A: More information about LED connections

How to connect the + of 16 LEDs to only two +LED pins on the light controller?

The Mk4 light controller is very small and only offers two +LED connections. The design idea is that the + (Anode) of all LEDs are tied together right at the LEDs, and then only a single wire goes from the various LED clusters in the car to the light controller.



Alternatively, you can use the **Mk4P pin-header version of the light controller**, where a separate +LED output is available for each LEDs.



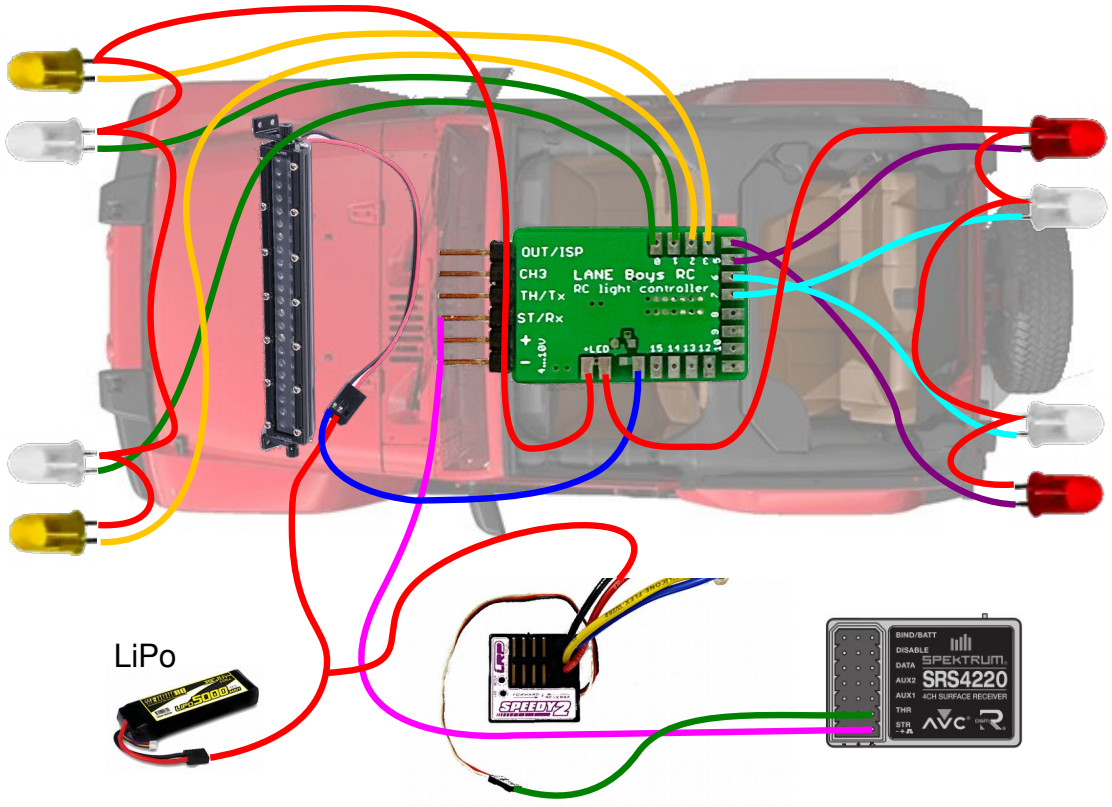
Using a separate power source for the LEDs

Usually the light controller is powered from the BEC that feeds the receiver and servos.

Alternatively, it is possible to drive the LEDs from a separate power source, while the light controller is still powered form the BEC. This could be useful, for example, if the vehicle is

powered from a 3S LiPo (about 12V) and a 12V capable light bar is used.

In the following schematic the light bar is powered directly from the car LiPo, while the light controller and other LEDs are powered from the BEC of the ESC:



The power source can be either a separate battery, the driving battery of the car, or a separate BEC (DC/DC converter).

This method can be applied to any light output (current-controlled or switched).

> ***When used for current-controlled outputs care must be taken that the <u>total</u> power dissipation of the light controller does not exceed 2.5W***

The power dissipation for current-controlled light outputs can be calculated as follows:

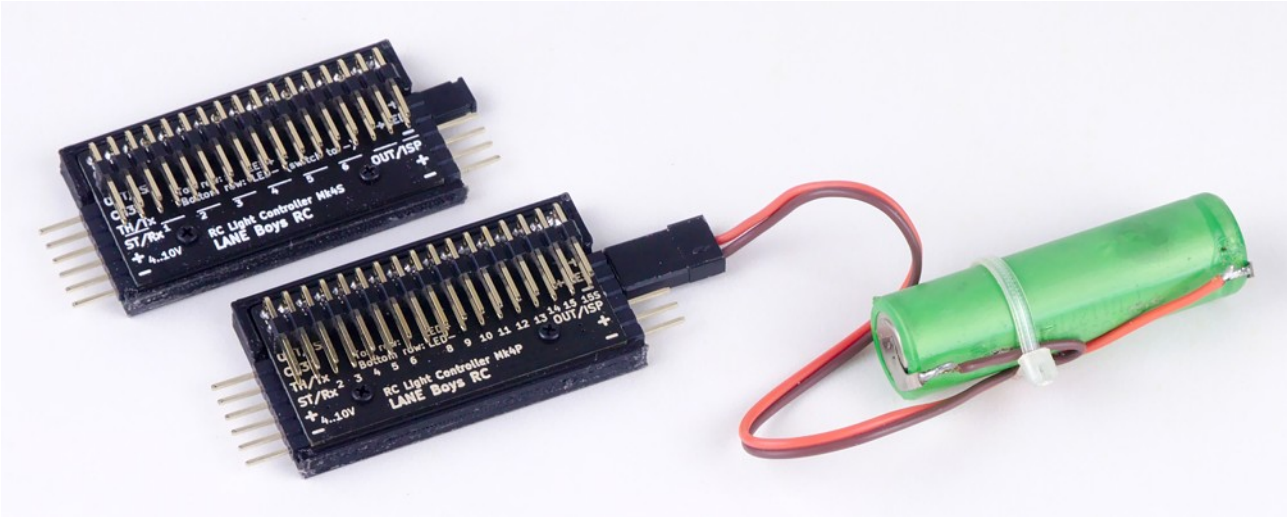
(Supply-voltage minus LED-forward-voltage) times LED-current

Example: **8.4V** supply voltage, **3V** LED forward voltage, **20mA** LED current
(8.4V – 3V) × 0.02A = **0.108W** (per LED output!)

On the Mk4 light controller the +LED pins on the circuit board are directly connected to the + input power supply. Do not use +LED when you want to power the LEDs from an external power source.

The Mk4P and Mk4S light controller have a separate connector for choosing whether the +LED connection is powered from the BEC/receiver or from a separate power source.

The supplied jumper can be used when powering +LED from the receiver, or instead a servo connector or JST connector can be used to supply the +LED from a separate battery.

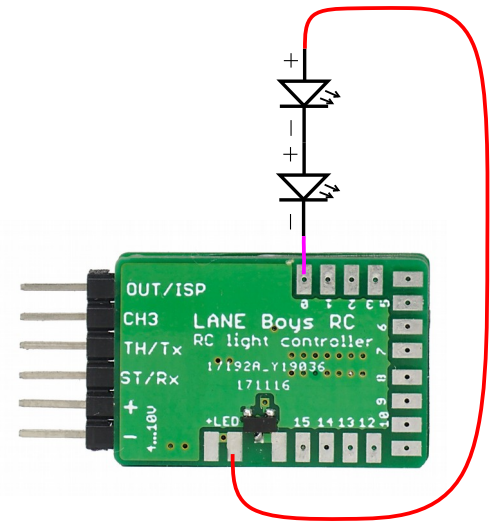


*The light controller on the top has the jumper installed to connect the **+LED** lines to the **+** from the receiver.*

*The light controller on the bottom has the jumper removed, and instead an external battery connected to **+LED** and **-** pins.*

Wiring LEDs in series

Depending on the power supply voltage and the LED's forward voltage, it may be possible to drive two LEDs in series on a single light output:



The LED forward voltage depends on the color of the LED.

- **Red** and **Amber** LEDs have a forward voltage of about **1.8V**
- **Green** LEDs about **2.2V**

- **White** LEDs about **3.2V**
- **Blue** LEDs may require up to **3.6V**

> ***Always consolidate the datasheet of your LEDs for exact values.***

The light controller itself requires at least 0.8 V to be able to control the current.
Usually the light controller is powered from the BEC in the car, which outputs 5V or 6V.

Example: **Two red LEDs are wired in series**,
...so the voltage drop across them is 1.8V times two = **3.6V**.
The BEC voltage is **5V**.
5V – 3.6V = 1.4V, which is more than the 0.8V that the light controller requires for proper operation, so **everything works fine**.

Example 2: **A white and a blue LED wired in series.**
The voltage drop across both LEDs is 3.2V + 3.6V = **6.8V**.
The BEC voltage is **6V**, so it is **not high enough to drive those LEDs at full brightness**.

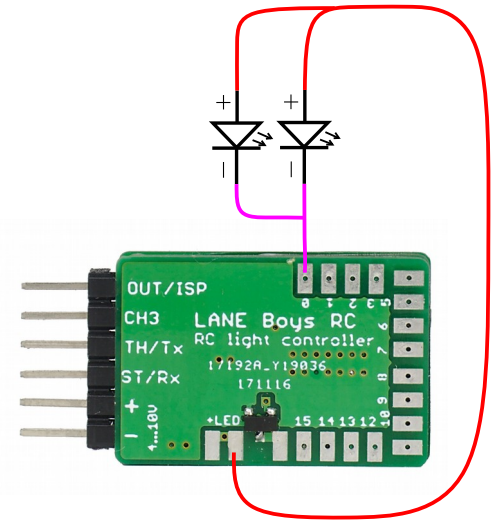
However, it is possible to power the light controller from the 2S LiPo car battery.
A fully charged LiPo is **8.4V**.
8.4V – 6.8V = 1.6V, which is more than the 0.8V required by the light controller.
As the LiPo discharges and voltage drops below 6.8V + 0.8V = **7.6V**, the LEDs will be not driven properly anymore and may become dim.

> ***When powering the light controller directly from a 2S LiPo, precautions have to be taken to ensure that the installation is safe against short circuits, as the energy stored in a battery can cause fire.***
>
> ***Furthermore, the power dissipated in the light controller increases with the input voltage. The light controller may become too hot and malfunction.***

Wiring LEDs in parallel

In general, it is not recommended to wire LEDs in parallel. LEDs are non-linear devices – a small change in forward voltage can cause a large change in the current flowing through the LEDs. If two LEDs wired in parallel have a different forward voltage, than one of the LEDs may draw more current than the other LED.

That said, in practice it is usually OK to wire two LEDs in parallel, as modern LEDs are manufactured to tight tolerances.



Only LEDs of the same manufacturer, type and color can be wired in parallel.

When two LEDs are wired in parallel, half of the current that the light controller regulates runs through each LED.

One potential use-case is to save an output for white parking lights or reversing lights. Wiring white LEDs in series (see previous section) may not possible as the total voltage required could be more than the supply voltage. However, since parking lights or reversing lights don't need to be very bright, wiring two LEDs together can free up one output on the light controller for another purpose.

Still, it is advisable to wire each LED to a dedicated output on the light controller. Parallel connections should only be used as last resort.

Appendix B: Resistor calculation for LED and light bars

LEDs are active components with a non-linear relationship between voltage and current. Even a small increase in voltage may cause a large increase in the current flowing through the LED.

Furthermore, the temperature of the LED influences this relationship, hence current may increase when the LED heats up, which causes further current increase – and so on, until the LED burns out.

In order to prevent LED damage, we therefore need a current limiting resistor before we can connect an LED to a voltage source.

Outputs OUT0 to OUT15 on the Mk4 and Mk4P light controller are current-regulated and do not need a current limiting resistor.

However, LEDs connected to OUT15S on Mk4 and Mk4P, or any output on Mk4S (switching version) needs a current limiting resistor.

Resistor value for a single LED

The formula is very straight forward:
Resistor = (Supply-voltage minus LED-forward-voltage) divided by LED-current

The forward voltage of an LED is specified in its datasheet. If you don't have a datasheet available for the particular LED you are using, refer to the following guide:

- **Red** and **Amber** LEDs have a forward voltage of about **1.8V**
- **Green** LEDs about **2.2V**
- **White** LEDs about **3.2V**
- **Blue** LEDs may require up to **3.6V**

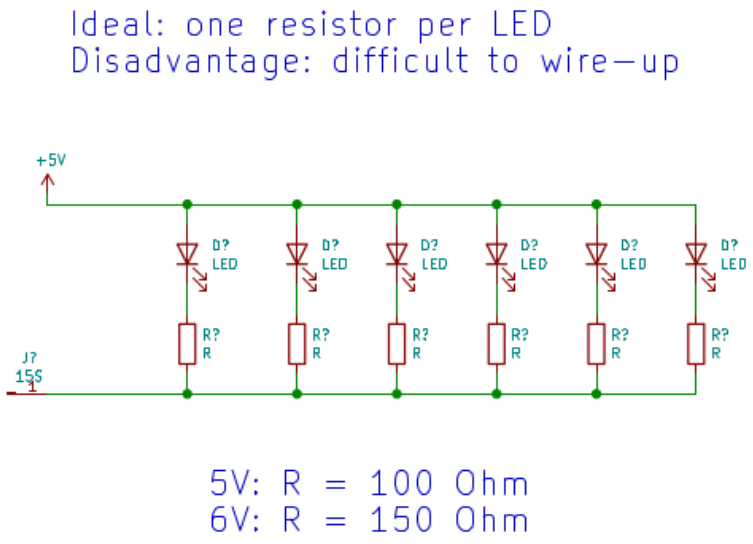
Note that this table applies to normal 3mm, 5mm or small SMD LEDs. Power LEDs may have very different voltage and current requirements.

Example: Supply voltage: **6V**. Amber LED with **2V**. Desired current: **20mA**

(6V – 2V) / 0.02A = 200 Ohm → next larger standard value would be **220 Ohm**

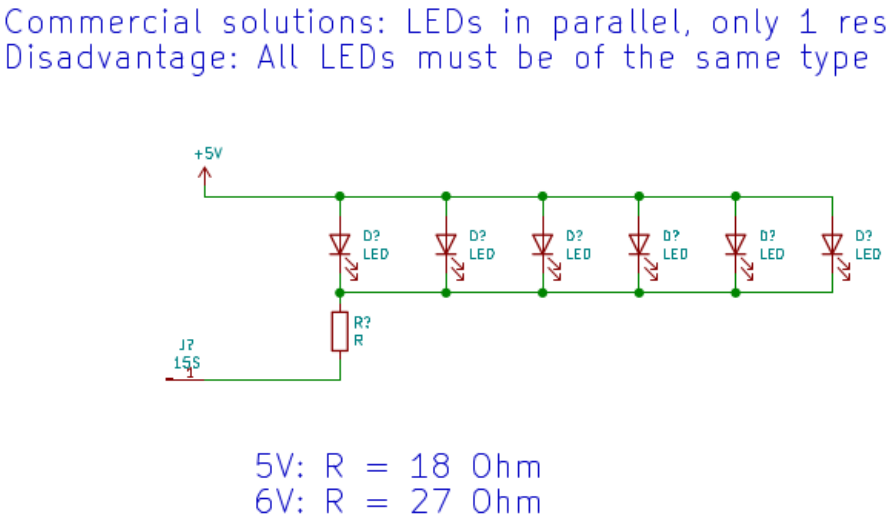
Resistor value for a light bar using multiple LEDs

Ideally each LED in a light bar would have its own current limiting resistor:



In this configuration, the formula for a single LED, described above, applies.

In practice, most commercial light bars have all LEDs wired in parallel and utilize a single current limiting resistor for the whole light bar. This “works” because modern LEDs have relatively tight tolerances, so the current through all LEDs is *roughly* the same.



The formula for calculating the current limiting resistor is similar as for a single LED, except that the current now is the total current of all LEDs:
Resistor = (Supply-voltage minus LED-forward-voltage) divided by (LED-current times number-of-LEDs)

Example: Supply voltage: 6V. White LEDs with a forward voltage of 3V. Desired current per LED: **20mA**. Number of LEDs: **6**.

(6V – 3V) / (0.02A × 6) = 25 Ohm → next standard value = **27 Ohm**

Important: consider the power dissipated in the resistor!

Especially for light bar use, a lot of power is potentially dissipated in the current limiting resistor, causing it to get hot and possibly fail.

The required power capability of the resistor can be calculated as follows:
Power = (Supply-voltage minus LED-forward-voltage) times LED-current times number-of-LEDs

Example: Supply voltage: 6V. White LEDs with a forward voltage of 3V. Desired current per LED: **20mA**. Number of LEDs: **6**.

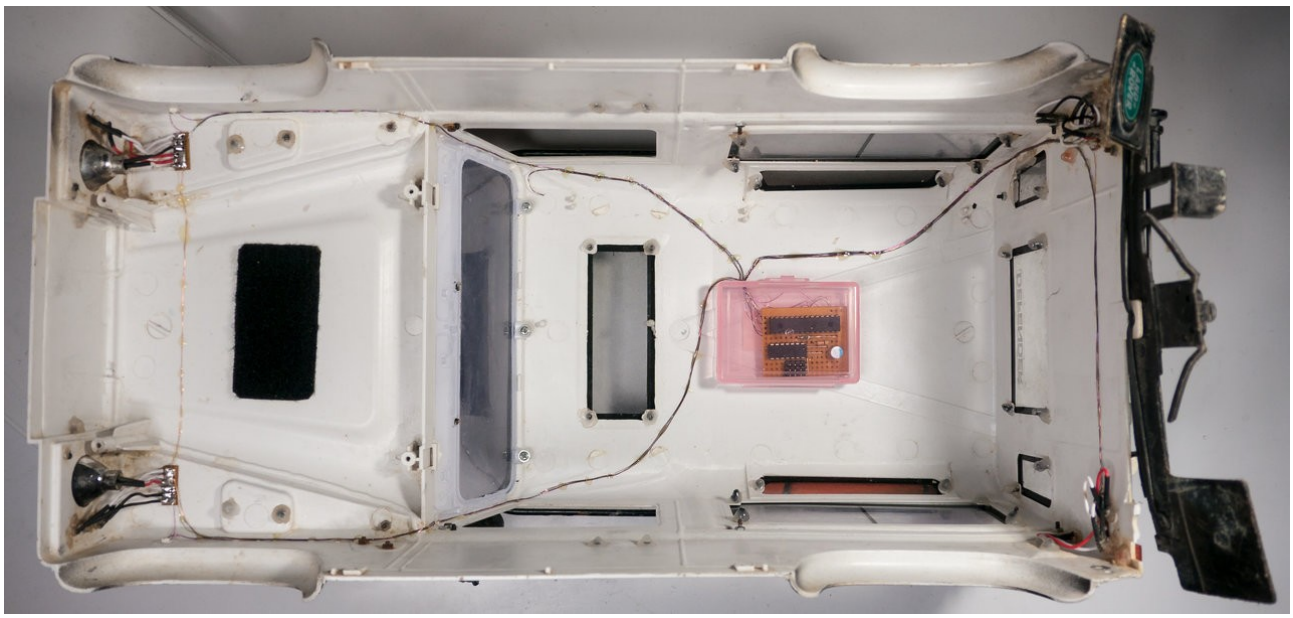
(6V – 3V) × 0.02A × 6 = **0.36W**

In this example we therefore need at least a **half-Watt power resistor**; a typical quarter-Watt resistor would burn out quickly.

Appendix C: Waterproofing

The light controller is **not** waterproof.

To protect the light controller from moisture, place it in a waterproof box and seal all wire exits.



Appendix D: LPC81x-ISP and alternatives

The micro-controller used in the light controller was chosen because it has a built-in bootloader that allows flashing of firmware using an inexpensive USB-to-serial adapter.

The tools to program the firmware are usually part of a large Software Development Kit (SDK) for the micro-controller, which is tedious to install.

The LPC81x-ISP tool was created to make it easier for light controller users to flash the firmware.

The LPC81x-ISP tool is Open Source. The project repository is at: https://github.com/laneboysrc/LPC81x-ISP-tool

The program is written in a programming language called Python (https://www.python.org).

The tool can run on Windows, Linux and Mac OS. However, a executable file is only available for Windows.

Some users reported that their virus scanner identified the tool as potentially malicious. If you don't trust the executable file, you can download the Python source code, install Python and run the tool this way.

The tool can also run on a Raspberry Pi – and because the Raspberry Pi has native serial ports no USB-to-Serial adapter is required.

You can also program the light controller firmware with any other tool that can program the NXP LPC812 micro-controller via UART, for example Flash Magic (https://www.flashmagictool.com/)

Furthermore, we offer a WebUSB programmer that works directly in the Web Browser. With this hardware it is possible to flash the light controller directly from within the *Configurator*. Please refer to section Easier firmware flashing with the WebUSB programmer on page 17 for more details.

Appendix E: Solving programming issues

When ordering the light controller from LANE Boys RC, customers receive a USB-to-Serial adapter and a custom programming cable. Before the items are shipped out, they are tested for correct functionality.

Still, every now and then users report issues with programming the light controller using the LPC81x-ISP tool described in the previous section.

In most cases the Micro-USB cable (not supplied) turned out to be faulty.

In some cases, the USB-to-Serial adapter was not recognized by the operating system. The USB-to-Serial adapter we ship uses the well-established CH340 chip. All modern operating systems come with drivers for this chip and should work without any intervention.

On Windows some restrictive administrative settings may prevent the driver from auto-installing. In these rare circumstances manual driver installation may be required.

Please refer to https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all for detailed instructions.

Some computers may also have built-in serial ports. The LPC81x-ISP tool may by default choose the built-in port rather than the USB-to-Serial adapter. Please select the correct COM port in the drop down box of the LPC81x-ISP tool. You can find the correct COM port in the device manager as outlined in the link above.

Some users are still using a 32-bit version of Windows. A compatible version of the LPC81x-ISP tool that works on 32-bit Windows can be downloaded here:

https://laneboysrc.github.io/rc-light-controller/lpc81x-isp-windows-32bit.zip

Appendix F: Determining the AUX switch type

Several light functions can be controlled through an AUX channel connected to a switch on the transmitter. The main functions are operated through a multi-function switch as explained in section Switching the main lights on page 20.

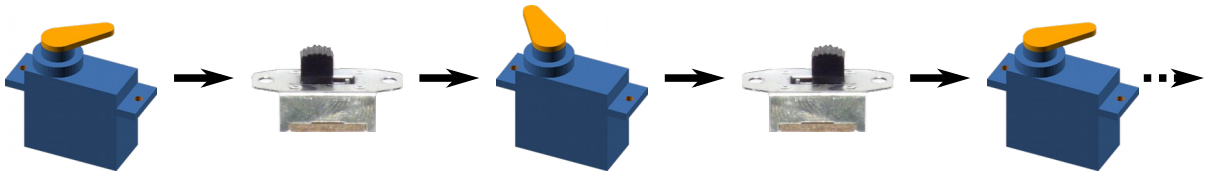
The following switch types are supported

- Two-position switch
- Two-position switch with up/down buttons
- Momentary push button
- Push button on the light controller (see section Using a push-button installed in the car to switch functions on page 8)

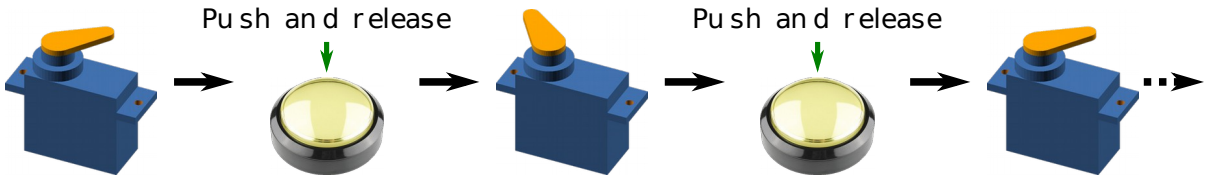
You can determine the switch type of the AUX channel on your transmitter as follows:

Connect a servo to the AUX channel on the receiver. Operate the AUX switch (or push buttons) on the transmitter and match the behavior of the servo with the description below:

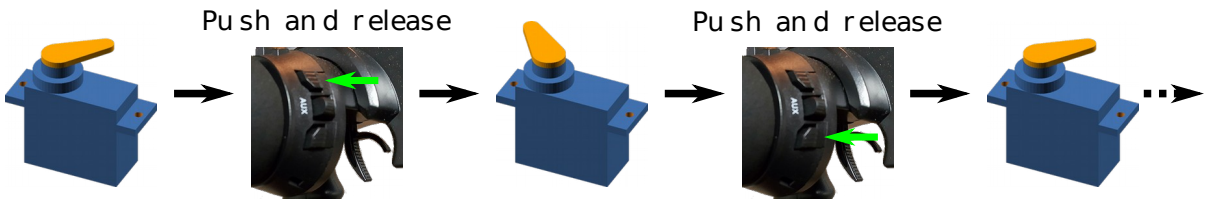
Two position switch:



Push button implementing a Two position switch:



Two-position switch with up/down buttons



Momentary push button: