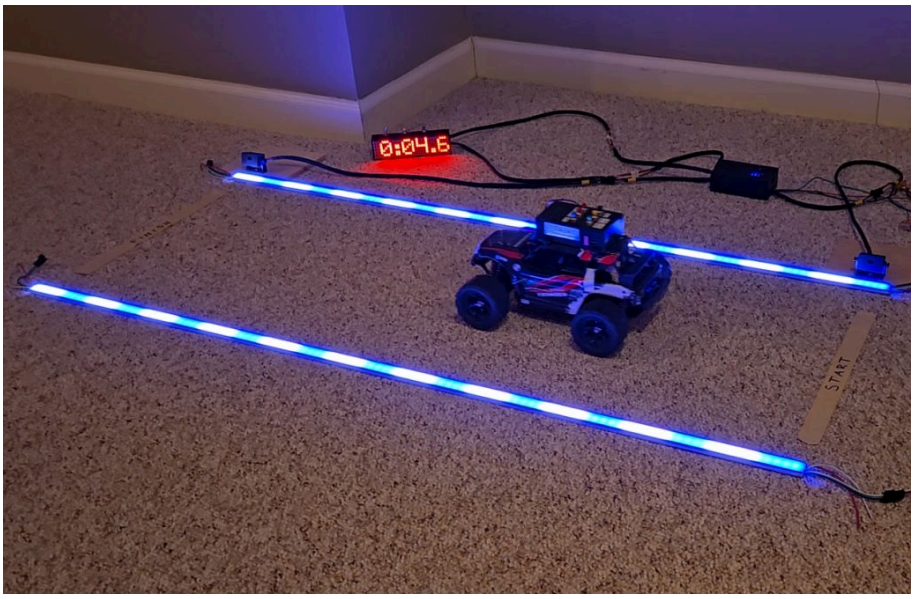
Robot Car Race Timing System



System Setup and User Guide

For Firmware Version: 0.25

Doc Version 0.25.0
December, 2024

Installation, Setup and Initial Configuration

Important Notes about this Document

This document was created as a downloadable version of most of the content found in the Github Wiki (https://github.com/Resinchem/Robot-Car-Timer/wiki). However, while the Wiki is a living document that will be kept current with future enhancements of the firmware, this document will likely remain static. For this reason, the Wiki should always be checked for the latest information and instructions.

This guide was current as of release v0.23 of the firmware and both the version and release date of this document can be found on the title page. If there are later releases of the firmware, you can also check the Wiki pages to see if they have been updated since this document was created.

The Wiki and version release notes found on Github should always be considered the most current and definitive source of documentation for the firmware.

Examples, screen shots and general use described in this document is based on the system build as covered in both the YouTube video and related blog article for the project (links in the appendix). If your build is different or uses different components, then portions of this manual may be incorrect for your specific system.

This document only applies to the full version of the firmware. Older releases also contained a non-WiFi version that worked without WiFi. Info on that version is briefly covered in the appendix but is no longer being supported and isn't described in the main sections of this guide.

Always follow general safety procedures for working with small electronics and assure all components used are properly rated for the voltages and current required.

This timing system is ***not*** certified by NIST or any other agency. It should be used for entertainment purposes only and not as a timer for any sort of "official or sanctioned" event.

This document and the referenced firmware are provided "as is" with no warranties, express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose. The user assumes all risk and responsibility for its use, including potential errors, omissions, or interruptions.

Initial Firmware Installation

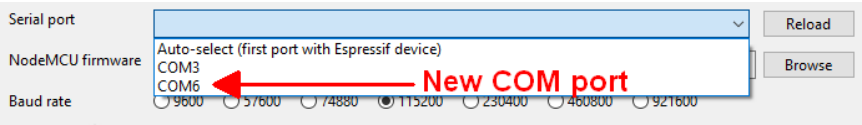
The very first time you install the firmware onto a new ESP32, you must use a USB cable and connect the ESP32 to a computer to flash the firmware. After the initial flash, future updates or firmware versions can be installed wirelessly over-the-air.

Beyond a computer, USB data cable and an ESP32, you will need two additional items:

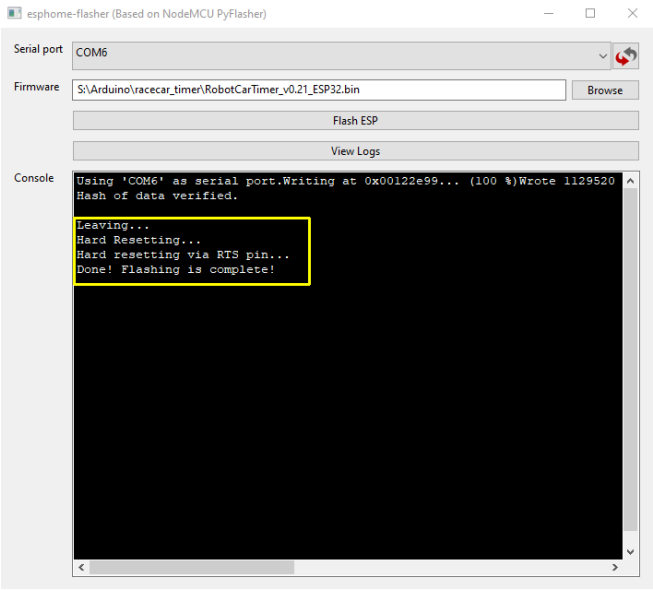
A local downloaded copy of the latest firmware .bin file.
This can always be found at: https://github.com/Resinchem/Robot-Car-Timer/releases/latest. Remember that you need the .bin file, which is found under the release assets and follows a naming convention of RobotCarTimer_vx.xx_ESP32.bin, where x.xx indicates the release number. You cannot flash or install the source code (.ino) files directly onto the ESP32. The source code must first be compiled into a .bin file. Download and save the .bin file locally.

A desktop flashing utility
There are numerous utilities for flashing firmware to an ESPx board. Two of the most common are NodeMCU PyFlasher (Windows & Mac) and ESPHome Flasher (Windows, Mac & Linux). Simply navigate to the latest release section and download the file(s) for your operating system. In most cases, these utilities work without installation. I'll be showing ESPHome Flasher here (no… you do not need to have or use ESPHome for this flasher to work). Most utilities work in a very similar manner.

Once you have these two items, you are ready to flash the firmware. Connect the ESP32 to your computer using the USB data cable. Make sure the cable supports data use and isn't just a charging cable. A new COM port should appear on your device. Note this COM port number.



Then launch the flashing utility



ESPHome Flasher

For this utility, you only need to select the COM port where the ESP32 is connected and then browse to and select the .bin firmware file. Then simply click 'Flash ESP'. Watch the output or console window for information. If the flash is successful, you should see a message similar to the above. You can now close the flashing utility, unplug the ESP32 from the USB port and place or power it from the controller board. Alternatively, you can power the device temporarily via the USB port until onboarding is completed.

If you have issues with the initial flash, see the appendix for some troubleshooting information.

Now complete the onboarding and initial setup steps below.

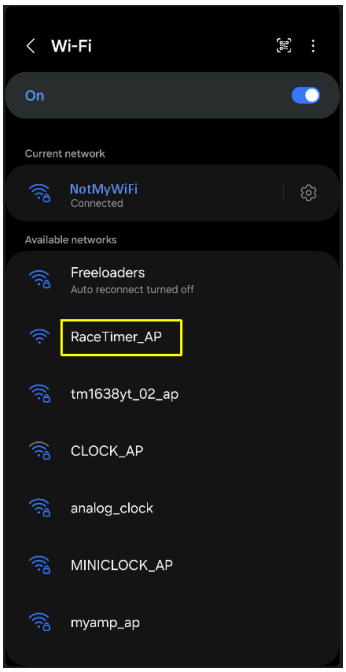
Note: This will not apply when installing the firmware for the first time on a new ESP32, but if you are using this method to upgrade or overwrite the existing firmware, some utilities will automatically erase the flash on the board. This could overwrite any existing configuration files resulting in the need to complete the initial onboarding steps that follow again.

Onboarding and Initial Setup

If the controller has never been set up, or if the controller is reset (see info below on resetting a controller), you must onboard the controller to your WiFi and specify a few starting settings. I recommend doing this before installing the ESP32 into the controller assembly.

You will need a mobile phone or other device (e.g. tablet, laptop) that can join an ad-hoc WiFi network. Two independent devices will be needed if you want to join the controller to a WiFi hotspot being broadcast by one of these devices.

I'll be using an Android phone (Samsung Galaxy S24) for the examples below, but the process should be similar on other Android or Apple phones/devices. Power on the controller and open the WiFi settings on your device. You should be in the near vicinity of the controller for the onboarding process.

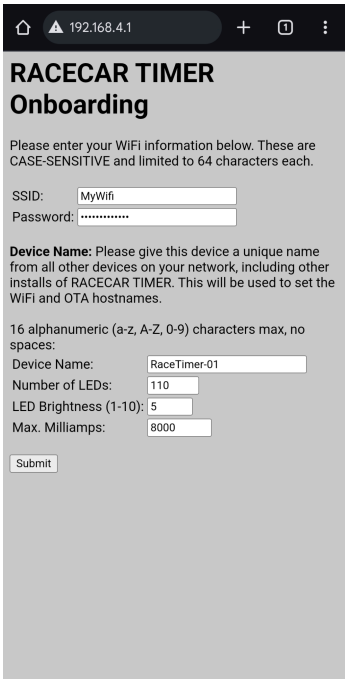


You should see a WiFi access point called "RaceTimer_AP". If you do not immediately see this access point, wait a few minutes and make sure you are within 10' of the controller. Restart the controller again if the AP does not appear.

Join this AP with your device. You may be warned that the AP does not have Internet connectivity. If prompted, select to remain connected to this AP.

You may also be prompted to "log in" to the new WiFi. If so, it should open your browser to the onboarding page. If not, simply open a browser and go to the IP address:

http://192.168.4.1



Enter in the WiFi name (SSID) and password you wish to connect to for the controller. See the special note below if you want to join a hotspot being broadcast by this or another device. Then enter the specific information for your installation:

Device Name: Each controller on a given network **must** have a unique name. This identifies the device on the WiFi, is used for OTA (over-the-air) updates and is also reflected on the web settings page so you can assure you are working with the correct controller in a multi-controller situation. You can use something like Timer-01, Timer-02, etc. but the device name should only use letters, numbers and the hyphen (-) and must be 16 characters or less.

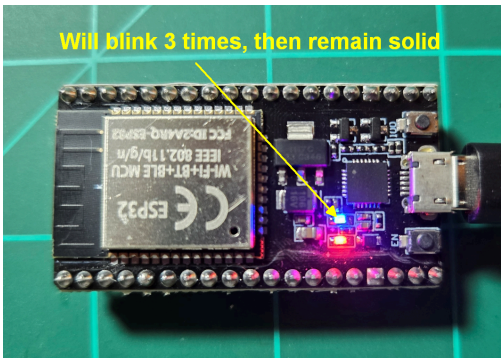
Number of LEDs: Enter the exact number of LEDs ***from one side only*** in your install. If you have identical matching LED strips on each side of the track, the LED signal is split and sent to both strips simultaneously. For this reason, you should only enter the number of LEDs on one side only. If the two sides have a different number of LEDs, use the larger of the two numbers. There is an upper limit of 500 LEDs for this project. If you are not using LED strips, or wish to disable their use, simply enter zero (0) for the number of LEDs.

LED Brightness: Set a starting LED brightness value from 0 - 10. This can be changed via the settings page after onboarding, but it is recommended that you start with a value of 5 or less. The LEDs will likely be brighter than you expect. Note that entering a value of zero effectively turns the LEDs off. This is different from disabling the LEDs by using zero for the number of LEDs. When brightness is set to zero, all calls to the LEDs still occur but with a brightness of zero (off). When disabled, calls to the LEDs are simply skipped.

Max Milliamps: To prevent the LEDs from pulling too much current, enter a maximum milliamp value here. This should not be higher than around 80% of the rated capacity for the power supply. For example, with a 10A power supply, a max value of around 8000 should be used here. There is a minimum value of 2000 (2A… small power supplies should only be used when disabling the LED functionality). This is also a maximum permissible value of 15000 milliamps (15A).

Once you have entered all values, click the **Submit** button.

The controller will automatically reboot and you will receive a message that the device is attempting to connect to the WiFi you specified. Watch the ESP32 controller at this point.



When first powered on, the red LED on the ESP32 will light solid. If the controller successfully joins the WiFi, the blue LED will blink three times and then remain solid. The timer will also show 'Ready' when the boot sequence completes.

If connected to the LEDs at this point, a successful boot/wifi join will be indicated by the boot pattern/LED test on the LEDs (see the Boot Sequence below for more details).

If the ESP32 does not appear to join the WiFi (blue light doesn't blink/remain solid), then use your mobile device to check and see if the **RaceTimer_AP** hotspot is still being broadcast. If this hotspot is still available, it means that the controller failed to join the WiFi and has reverted to broadcasting the hotspot. Join this hotspot again and repeat the onboarding steps above, assuring that the WiFi SSID and password are correct (these are CASE-SENSITIVE).

If you still cannot get a successful WiFi connection, check the WiFi signal strength. Note that the ESP32 boards have an onboard antenna, but it isn't necessarily the best and may struggle to connect to a very weak WiFi signal. It may be necessary to find a way to increase the WiFi signal strength if this is the case.

Normal Boot Sequence Display

Once installed, it may be difficult to observe the LED indicators on the ESP32, so if you have installed LED strip(s) and configured them for use in the initial onboarding (or updated from the settings page), then the LED strip(s) will also used to show certain steps in the boot up process. The normal sequence that should be seen for a routine boot up (after onboarding) should be as follows:

- Power Supplied: Red LED on the ESP32 should illuminate.
- LED strips will briefly cycle through red, green and blue to test the number of LEDs, brightness and milliamp settings.
- The IP address of the controller on the timer display, one segment at a time.

Example IP: 192.168.1.229

- OTA Uploads enabled: The LED strips will briefly turn alternating red and green and the timer display will show “Upload”. This is <u>not</u> an indication that an upload is occurring or that the system is ready to accept an upload. It is only used to show that OTA updates are enabled.
- The LEDs and display will briefly turn off as an attempt to connect to WiFi is made.
- When WiFi is joined, the blue LED on the ESP32 will blink three times then remain on. The timer display will show “Ready” and the LED strips will be set to the ‘Ready’ color (yellow by default, unless changed in the web settings).

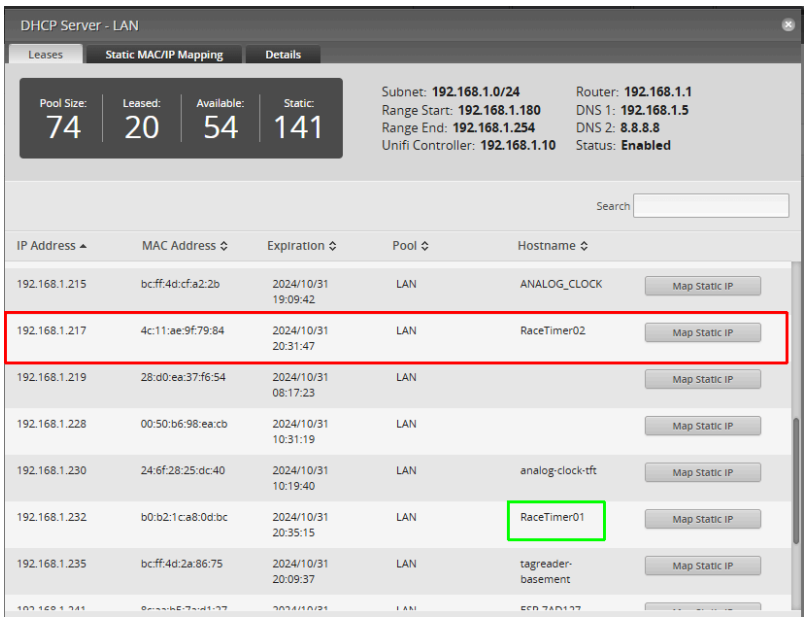
You can use these indicators to determine if, and where, any boot problems may be occurring. Once you see the ‘Ready’ indication on the timer, the system is “live” and ready to start timing.

Note: If the LEDs immediately switch to active race colors and the timer starts, it is likely that something is being detected by the start line sensor. Move any objects outside the field-of-view and maximum specified distance for the start line sensor or change the maximum distance value via the settings page.

Obtaining the Device IP address

Once your controller has successfully joined your WiFi, you will need to know the IP address of the controller to reach the web settings page or to otherwise interact with the controller for options like rebooting, flashing updated firmware, etc. As shown in the section above, the IP address is briefly shown on the timer display during the boot process.

You can also normally find the IP address assigned by visiting your router web interface. Your router's interface may look different from mine (Ubiquiti/Unifi):



If your router recognizes the hostname, then you should be able to easily find your device and its IP address by just searching for the device name you gave it. As you can see above, this is my second controller (named RaceTimer02) and has an IP address of 192.168.1.217. There is another system, RaceTimer01, that I initialized to my WiFi earlier… and it has an IP address of 192.168.1.232.

Now with the IP address, I can launch a browser on any device (phone, tablet, computer) that is on the same WiFi network to see the settings page and to make changes.

Using a Mobile Hotspot

If you wish to use a mobile hotspot for WiFi connectivity (note that at the current time, a WiFi connection is required for many features, including using the settings page, OTA updates, etc.), then the setup can be a little bit more tricky and will likely require two mobile devices for initial setup.

Two devices may be required to onboard the controller because a device may not be able to broadcast a mobile hotspot and also join a wifi network at the same time (I was able to do this using a Samsung S24, but unsure if this will work with older Android or Apple devices). For onboarding, the hotspot must be broadcasting and you must also be able to join the temporary hotspot being broadcast by the controller. This may be easier with two devices, or necessary if your phone cannot broadcast a hotspot and join the controller's hotspot at the same time.

Device 1: This is the device that will be broadcasting the mobile hotspot/WiFi network for the controller. Start this device and begin broadcasting the hotspot.

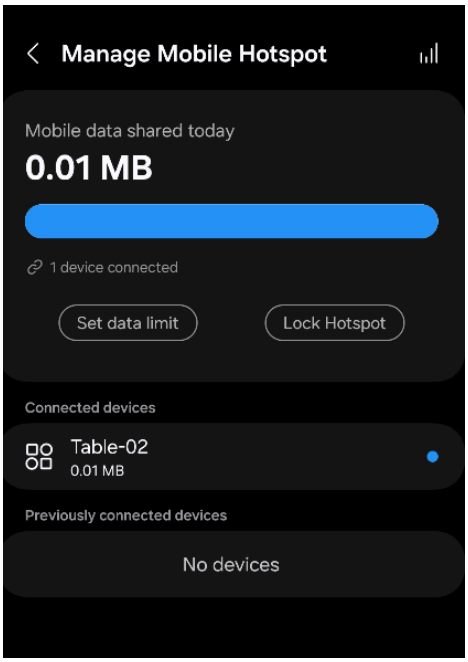
Device 2: This is the device that will be used to join the controller's hotspot (RaceTimer_AP) and complete the onboarding. Follow the normal onboarding process above using this second device.

Once the controller has rebooted and successfully joined the hotspot/WiFi (indicated by the solid blue LED on the ESP32), you no longer need the second device or can also join this second device to the mobile hotspot/WiFi to access the controller's settings with this device (you can use the first device to access the settings as well).

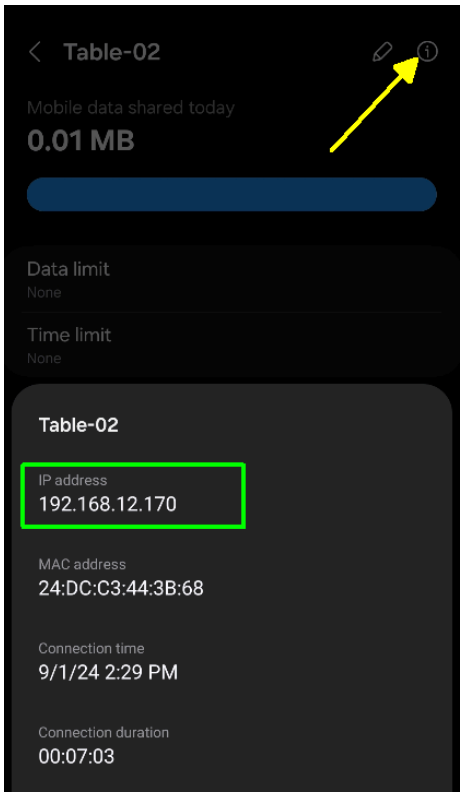
Getting the IP Address when using a Hotspot

You will still need to know the controller's IP address to access the settings page. However, since the controller is using the phone's hotspot, the IP address won't be in your router. Instead, you'll need to get the IP address from the phone itself. The method of doing this will vary greatly depending on the version of Android and/or if it is an Apple device. I'll show the process for the latest Android/Samsung phone here.

Under the mobile hotspot settings, you can see the connected devices



In the example above, you can see that I only have one device currently connected to the mobile hotspot… and that is Table-02, my second controller.



In my case, I could tap on the device and then on the “info” button at the top right to display the IP address for the controller.

The technique for getting the IP address will vary based on phone type and operating system, so you may need to do a Google search on “mobile hotspot connected device ip address” and you should find info on how to find the IP address of any connected devices.

Once you have the IP address, you can use the browser of any device (same or different from the one broadcasting the hotspot) that is connected to the same hotspot/WiFi to access the web settings page by just entering the current IP address of the controller (e.g. 192.168.12.170 in the above example).

Special Note on performing Firmware Updates

While you can use a mobile hotspot for changing the settings and even restarting or resetting the controller, the one thing that you should NOT do when tethered to a mobile hotspot is a firmware update.

First off, you will need to have access to the new firmware file (.bin) somewhere on the local network for uploading. When connected to a mobile hotspot, this means that the .bin file would need to be copied to the local phone and be accessible. You also do not want any “interruptions” during the upgrade process. For this reason, it is recommended that you join/rejoin the controller to your normal WiFi network when doing firmware upgrades. More info on changing wifi networks and performing updates is provided below.

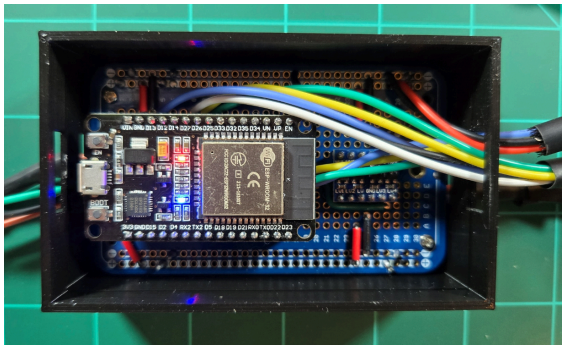
Installing the Hardware

Before looking at the settings and making changes, it is advisable to set up the physical devices and complete all the connections in a configuration approximating the final use. I will be covering the connections and components as covered in my YouTube video and the related blog article. If you build your own and make changes, then update these instructions accordingly.

Primary Components

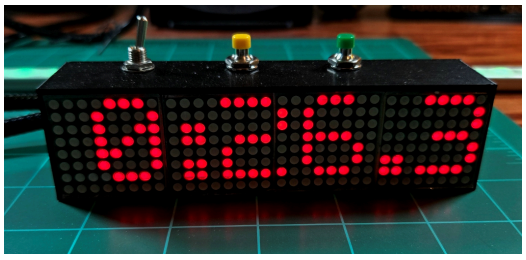
The system consists of these major components (shown in optional 3D printed enclosures):

Controller



The primary controller contains the ESP32 and has a number of labeled external connections, which will be covered below.

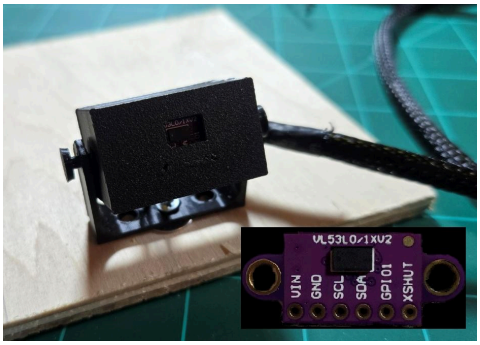
Timer Display



The timer display not only shows the active race time, but is also used to display system messages, like ‘Ready’ when the system is ready to start timing or ‘Upload’ when the system is in OTA update mode.

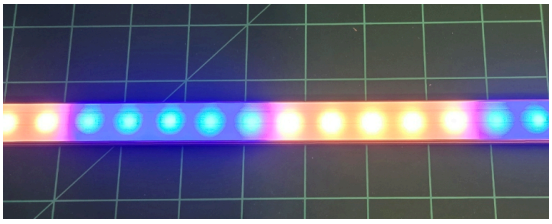
The timer also houses the toggle for setting the timing mode and buttons for resetting the timer and a start/stop button when using manual timing mode (see the General Use section for more details on timing modes and using the buttons).

Start and Finish Line Sensors



Two VL53L0X sensors are used… one for the starting line and one for the finish line. The sensors themselves are interchangeable (either sensor may be connected to either sensor lead from the controller). I have mine in a 3D printed enclosure above that allows easy adjustments to both the horizontal and vertical for precise track alignment (more on this below). If not using an enclosure, you should find a way to mount the controllers so that they do not move or shift during use.

LED Strip(s) - Optional

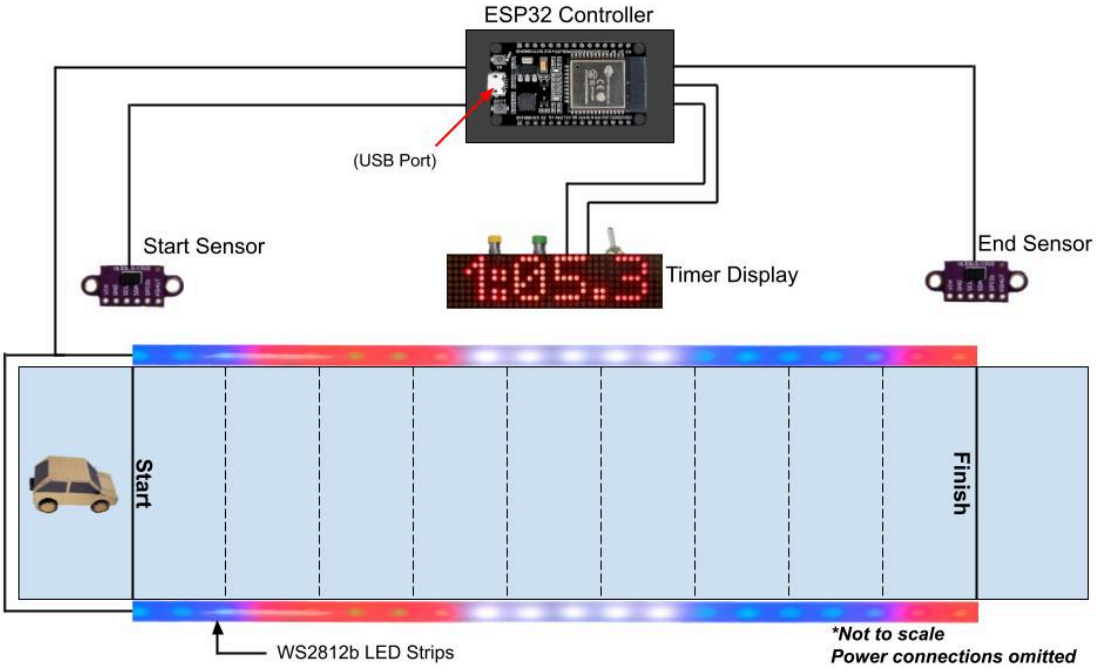


The system is designed for up to two LED strips that would normally be installed in parallel with the race track/course, one strip on each side. LED strip use is optional and the system can function without LEDs. However, if omitting LEDs, you should set the number of LEDs to zero in the settings. This will prevent the system from making unnecessary calls to the LEDs if they do not exist.

The firmware is currently designed for use with 5V WS2812B LED strips and the firmware has a hard limit of up to 500 LEDs (per strip). You should also assure your power supply can adequately provide the necessary amps to power the ***total*** number of LEDs (not just the count from one strip or one side).

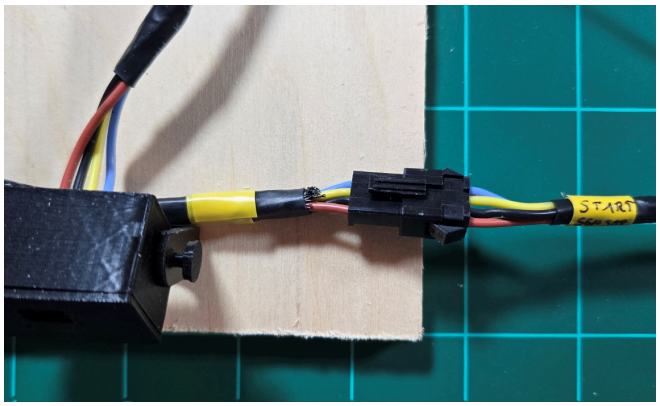
It would be possible to use other types of LEDs, but this has not been tested and it may require changes to the firmware to support other LED types.

Connecting Components



This diagram shows the primary connections between components. However, all components are wired with different styles and orientation of JST connectors and color coded, making it so that components only connect in one way and eliminating the possibility of connecting the wrong component to the wrong cable or reversing the polarity on any connections. This does not include power connections, which are covered below.

Sensor Connections



The sensors connect to the controller using 4-pin black 2.54mm JST connectors. The sensors themselves are interchangeable, but the start line sensor is connected to the cable coming from the controller on the end closest to the ESP32 USB connection while the cable connection for

the finish line sensor is opposite the USB port. If your particular install requires that the two sensor cables coming from the cable need to be swapped (e.g. finish line sensor cable extends from the USB end of the controller), this can be done by simply swapping the GPIO pin numbers for the two I2C buses in the source code:

```
// ======================================================
//  Change or update these values for your build/options
// ======================================================
//Pin Defintions - update if your build is different
#define BUS1_SDA 21    //I2C Bus 1 Data (ToF start sensor)
#define BUS1_SCL 22    //I2C Bus 1 Clock (ToF start sensor)
#define BUS2_SDA 17    //I2C Bus 2 Data (ToF end sensor)
#define BUS2_SCL 19    //I2C Bus 2 Clock (ToF end sensor)
```

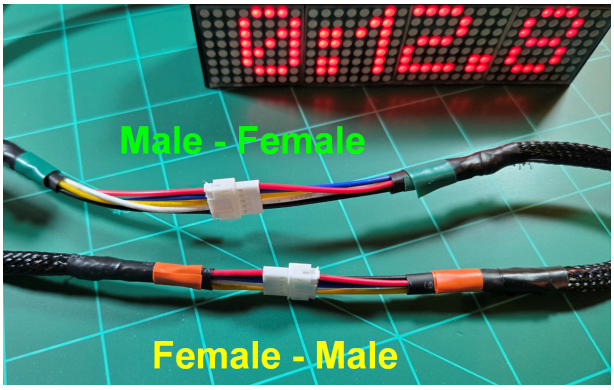
Original values with start line sensor cable on USB end

```
// ======================================================
//  Change or update these values for your build/options
// ======================================================
//Pin Defintions - update if your build is different
#define BUS1_SDA 17    //I2C Bus 1 Data (ToF start sensor)
#define BUS1_SCL 19    //I2C Bus 1 Clock (ToF start sensor)
#define BUS2_SDA 21    //I2C Bus 2 Data (ToF end sensor)
#define BUS2_SCL 22    //I2C Bus 2 Clock (ToF end sensor)
```

Swapped values with finish line sensor cable on USB end

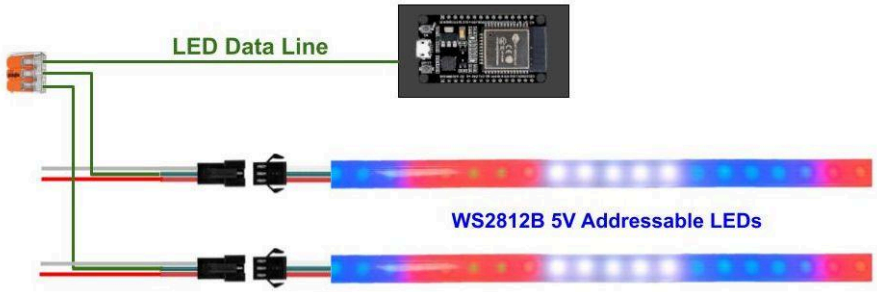
Naturally, you will need to compile and flash your changes to the controller. More information on making code and firmware changes can be found in the appendix.

Timer Display Connections



There are two connections from the timer display to the controller. Both use smaller 5-wire white 2.0 mm JST connectors. The polarity is reversed on the two different connectors, so you cannot inadvertently connect the wrong cables together.

LED Strip Connection(s)



There is a single LED data line from the controller. If using two LED strips, the data line from the controller is split and run in parallel to the center pin of a 3-pin male JST connector. The other two wires/JST pins are for power, which is covered next. If only using one LED strip, the LED data line from the controller can run directly to the JST connector

Do note that the firmware is currently written for WS2812b LED strips. Other LED types are supported, but may require modification of the firmware source code to indicate the type of LEDs in use:

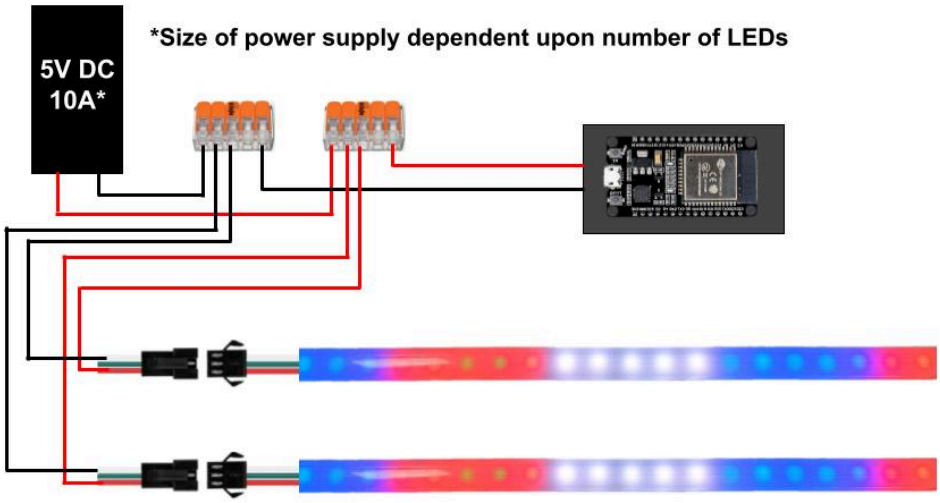
```
FastLED.addLeds<WS2812B, LED_DATA_PIN, GRB>(LEDs, numLEDs);
```

You would need to substitute the WS2812 with a supported chipset for your LEDs:

Chipset	Supported	Wires	Color Bits	Data Rate	PWM Rate	Chipset Power Draw
APA102/DOTSTAR	√	4	8	~24Mbps	20khz	0.9ma@5v
WS2811	√	3	8	800kbps	400Hz	5mw / 1ma@5v
WS2812B/NEOPIXEL	√	3	8	800kbps	400Hz	5mw / 1ma@5v
TM1809/TM1812	√	3	8	800kbps	400Hz	7.2mw / 0.6ma@12v
TM1803	√	3	8	400kbps	400Hz	7.2mw / 0.6ma@12v
TM1804	√	3	8	800kbps	400Hz	7.2mw / 0.6ma@12v
WS2801	√	4	8	1Mbps	2.5kHz	60mw / 5ma@12v
UCS1903	√	3	8	400kbps	unknown	?
UCS2903	√	3	8	800kbps	unknown	?
LPD8806	√	4	7	1-20Mbps	4kHz	?
P9813	√	4	8	1-15Mbps	4.5kHz	?
SM16716	√	4	8	?	?	?
TM1829	X	3	8	1.6Mbps/800kbps	7kHz	6ma@12v
TLS3001	X	?	12	?	?	?
TLC5940	X	4	12	?	?	?
TLC5947	X	4	12	?	?	?
LPD1886	X	3	12	?	?	?

(See current list of FastLED supported chips here)

Power Connections

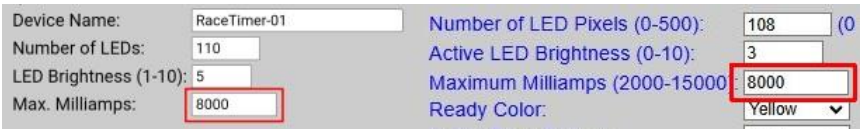


5V power is run in parallel to the controller and LED strips. Connect the (+) and GND wires from the power supply to Wago lever nuts, wire nuts or other connector type. Then run individual 5V/GND wires to the controller and each LED strip in use. Power for the timer display is taken from the controller board's 5V input.

Power is run in parallel due to the potential high current draw of the LED strips. Attempting to draw power for anything more than just a few LEDs through the controller board itself could result in damage (or destruction) of the ESP32 which isn't designed for large current.

Selecting the proper size power supply is covered in the build details, but recall that each individual LED pixel for standard WS2812b LEDs can draw up to 60 mA. This means if your installation uses 200 LEDs (e.g. 100 on each side), the maximum current draw for 100% bright white light could theoretically draw up to 12A!.

This is also why both the initial onboarding form and the web settings page have a setting for maximum milliamps. This will attempt to limit the current draw of the LEDs by limiting the maximum brightness.

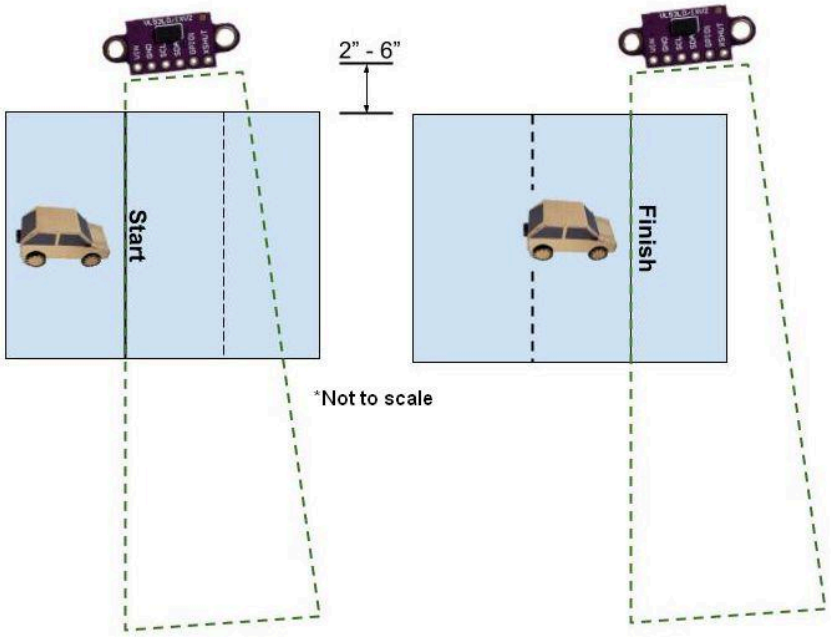


This setting will attempt to limit the total current draw of the LEDs to not exceed this value. For safety, it is recommended that you set the max milliamps equal to about 80% of the rated peak amps for the power supply.

Placing and Aligning the Sensors

After completing the initial onboarding and making all the physical wiring connections, the last step is placement of the sensors. Proper placement of the starting and finishing line sensors play a key role in how well the system operates. Note that the best results will be realized when the sensors are securely mounted, but allow small horizontal and vertical adjustments to fine tune the position.

Step 1 - Horizontal alignment



The VL53L0X has a field-of-view of 25°, so you have to think of the detection zone as a cone shape extending from the front of the sensor. You want to horizontally place each sensor so that it aligns as closely as possible The sensor should ideally be placed around 2-6" from the edge of the track. The VL53L0x has a minimum sensor range of 2" and may not reliably measure objects at less than this distance. On the flip side, best results will be obtained when the opposite side of the track (or the end of the zone you wish to monitor) is within 2-3 feet of the sensor. So you want the sensor close, but at least a couple of inches away from the start of the track or detection zone.

The easiest way to align the sensors is to set up and start the system, placing it in automatic timing mode (toggle on timer display). Place the sensors in an approximate starting position, assuring nothing is within the field of view less than 400 mm (~16") from the sensor. This is the default starting distance value for the sensor, but we'll fine tune that in step 3.

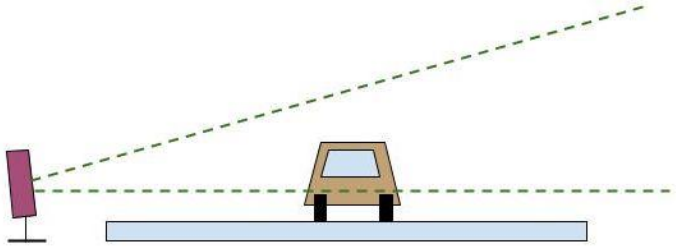
Then slowly move an object towards the start line until the timer starts. Adjust the horizontal position until the timer starts just as the object touches the start line. Repeat for the finish line,

but this time you want the timer running and position the finish line sensor so that that time stops as soon as the object touches the finish line.

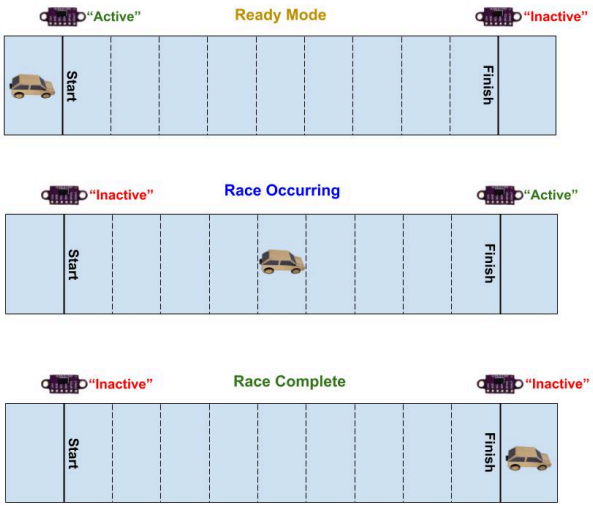
If you find that the time is starting or stopping immediately without an object in view, then it is possible that the sensor is detecting the track itself, so you will need to also make vertical adjustments.

Step 2 - Vertical Alignment

Remember that the sensor has a “cone” of detection, so this means it also detects objects above and below the sensor in addition to left and right.



The sensor will likely need to be slightly angled upward to avoid measuring the distance to the actual track itself. Continue to make horizontal and vertical adjustments to the sensors until properly aligned with the start/finish lines. Do note that the sensors are only “active” and measuring distances when the system is in the proper mode.



Before a race, when the system is in “Ready” mode, only the starting line sensor is active. Any objects, movement or motion within the detection zone of the finish line sensor are ignored.

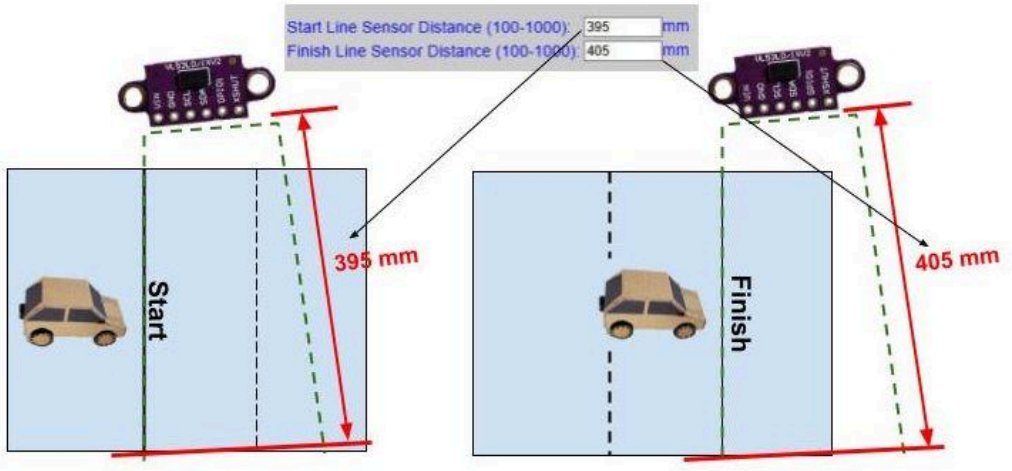
Once the race begins (the starting sensor is tripped and the timer starts), the finish line sensor becomes active and the starting sensor becomes inactive. This means that during an active

race, you can freely move around within the starting sensor detection zone and not impact the timing.

When a car or object crosses the finish line, the timer is stopped and both sensors become inactive. No objects or motion are detected by either sensor until the system is reset via the timer display button and the system is placed back in 'Ready' mode, once again activating the starting line sensor. These concepts are important to remember when testing and setting up your sensors. The start line sensor is only active in 'Ready' mode and the finish line sensor is only active when an active race is occurring and the timer is running. See more info on the race modes and use of the timer display buttons in the General Use section below.

Step 3 - Fine Tuning Detection Distances

The VL53L0x has a range of approximately 1.2 M (~3.9 feet), meaning it could easily detect an object beyond the track or racing surface. But we can limit the maximum distance that will cause a trigger to start or stop the timer.



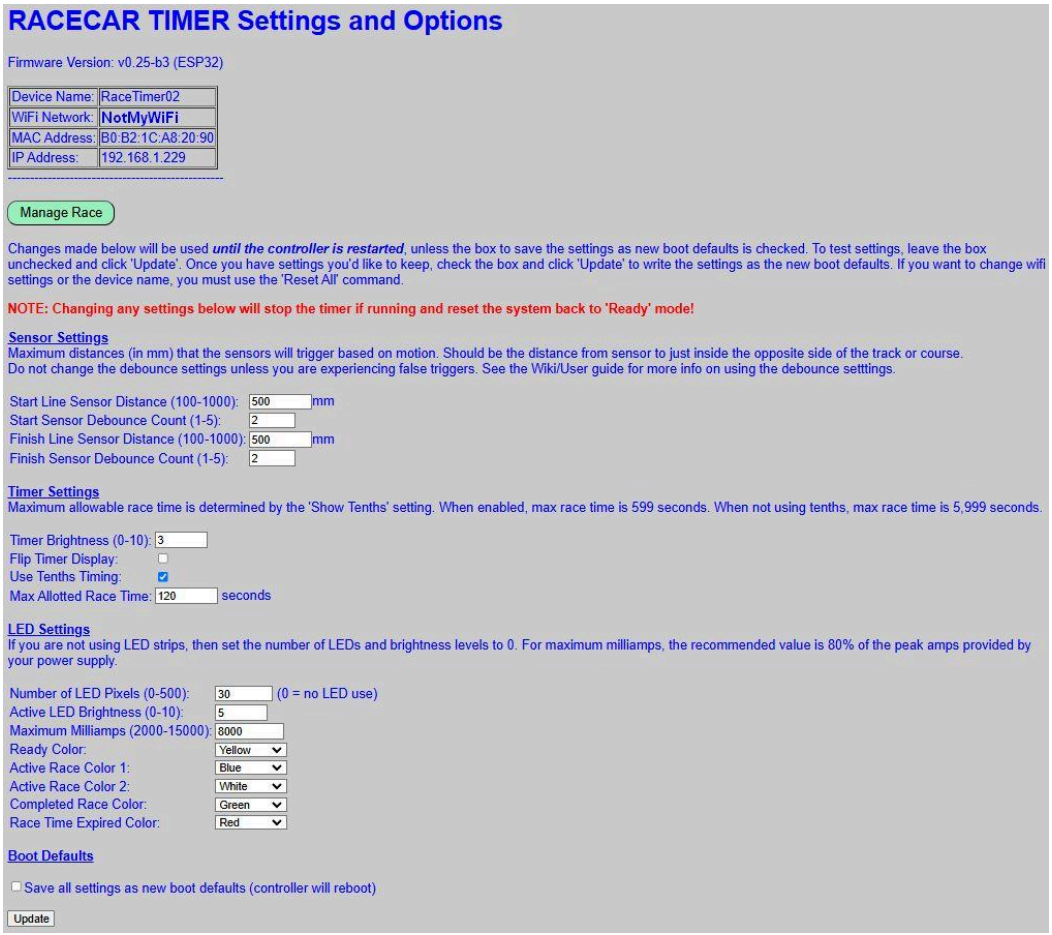
Recall that the VL53L0x is actually measuring distance and not detecting motion. We can set an upper limit on each sensor (via the Setting page) so that any object detected beyond this range is simply ignored. You can either measure the distances and enter them in the settings, or use an object just beyond the race surface and adjust the distances accordingly.

If you get occasional false triggers, you may need to adjust the debounce settings for the sensors. This is covered below under the Web Settings page section that follows.

The Web Settings Page

To access the web settings, simply enter the controller's IP address in a browser on the same network, the main settings page should be displayed. See the section above if you do not know the IP address of the controller.

***Note**:* The controller can be slow to respond when it is running in active race mode as the code has to wait for the current actions to complete before responding to other requests. It is recommended that you stop any active timer functions when working with the settings page or performing other operations like firmware updates. Ideally, the system should be in 'Race Complete' or 'Timeout' modes. While the ESP32 can do a lot, it is a bit limited in terms of processing power, so it still may take a few seconds for the settings page to be displayed.



RACECAR TIMER Settings and Options

Firmware Version: v0.25-b3 (ESP32)

Device Name:	RaceTimer02
WiFi Network:	NotMyWiFi
MAC Address:	B0:B2:1C:A8:20:90
IP Address:	192.168.1.229

Manage Race

Changes made below will be used ***until the controller is restarted***, unless the box to save the settings as new boot defaults is checked. To test settings, leave the box unchecked and click 'Update'. Once you have settings you'd like to keep, check the box and click 'Update' to write the settings as the new boot defaults. If you want to change wifi settings or the device name, you must use the 'Reset All' command.

NOTE: Changing any settings below will stop the timer if running and reset the system back to 'Ready' mode!

Sensor Settings
Maximum distances (in mm) that the sensors will trigger based on motion. Should be the distance from sensor to just inside the opposite side of the track or course.
Do not change the debounce settings unless you are experiencing false triggers. See the Wiki/User guide for more info on using the debounce setttings.

Start Line Sensor Distance (100-1000): 500 mm
Start Sensor Debounce Count (1-5): 2
Finish Line Sensor Distance (100-1000): 500 mm
Finish Sensor Debounce Count (1-5): 2

Timer Settings
Maximum allowable race time is determined by the 'Show Tenths' setting. When enabled, max race time is 599 seconds. When not using tenths, max race time is 5,999 seconds.

Timer Brightness (0-10): 3
Flip Timer Display:
Use Tenths Timing:
Max Allotted Race Time: 120 seconds

LED Settings
If you are not using LED strips, then set the number of LEDs and brightness levels to 0. For maximum milliamps, the recommended value is 80% of the peak amps provided by your power supply.

Number of LED Pixels (0-500): 30 (0 = no LED use)
Active LED Brightness (0-10): 5
Maximum Milliamps (2000-15000): 8000
Ready Color: Yellow
Active Race Color 1: Blue
Active Race Color 2: White
Completed Race Color: Green
Race Time Expired Color: Red

Boot Defaults

Save all settings as new boot defaults (controller will reboot)

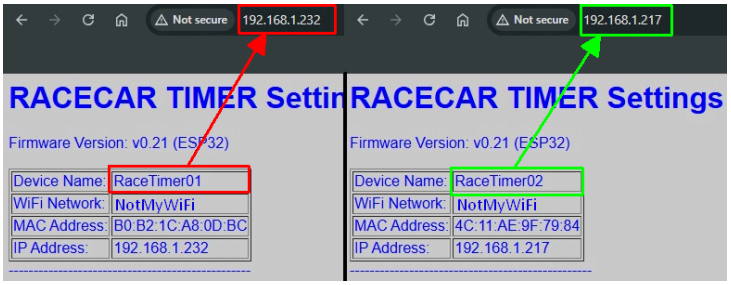
Update

Note that regardless of the current mode of the system, making any changes to the settings will automatically put the system back into 'Ready' mode. If an active race is occuring, it will be stopped and canceled.

In addition, all changes made via the settings page are “temporary” by default and will be retained until the next time the controller is restarted/rebooted, at which point the ‘boot defaults’ will be loaded from the saved configuration file. Info on making the changes permanent as new defaults will be covered below. Temporary changes can be convenient while fine tuning the system and making a lot of adjustments. Then when everything is how you want it, you can write those settings to the permanent/boot configuration file. Making your changes permanent and saving as the new boot defaults is covered below.

Each section of the Web Settings page will be described in detail.

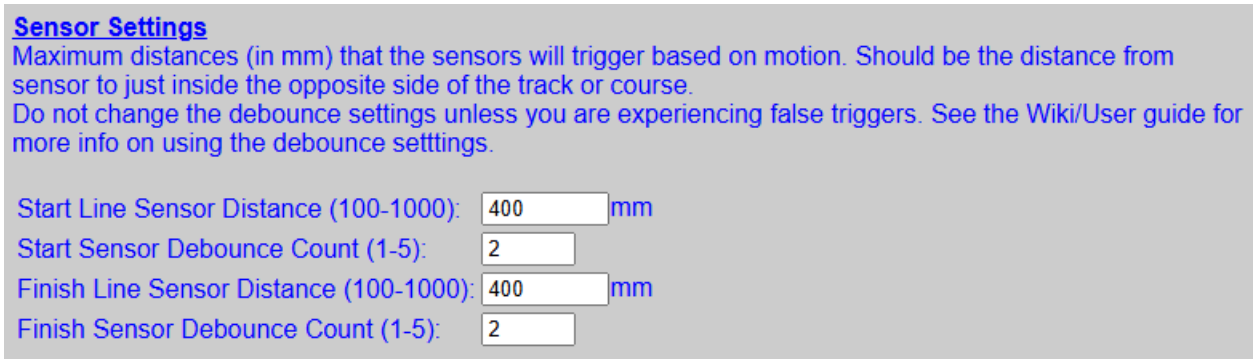
General Information



The top section provides general information, including the current firmware version and the device name you assigned when onboarding. This can be convenient when more than one controller is present. Each controller has its own settings based on IP address, so the device name allows you to be sure which controller you are accessing. The general info also contains the controller MAC and IP addresses.

Do note that the device name and the WiFi network information cannot be changed via the web settings after the controller has been initially onboarded. You must do a full reset of the controller and onboard it again to change either the device name or WiFi settings. Info on resetting the controller is covered below.

Sensor Settings



The use of the sensor distances is described above, under the section on placing and aligning the sensors. In short, you want to set distances here (in mm) from the sensor to the opposite side of the track or racing course. The valid range for both sensors is 100-1000 mm. This is just inside the published range of the VL53L0x sensors.

False Triggers and Debounce Settings

Long cable lengths, using small gauge wire or just general environmental factors can introduce noise into the sensor readings. This noise could result in a very brief distance reading that is less than the sensor's distance setting. When this occurs, the timer may appear to randomly start or stop without any object moving within the detection range of the sensor.

For this reason, a small debounce factor has been introduced starting with version 0.23 of the firmware. This debounce setting specifies how many consecutive readings in row must be less than the specified distance before the system will trigger. Note that each cycle/reading only takes a few milliseconds, so small debounce values will not impact the overall timing. Larger values, however, could add up to a true delay in the timing. For this reason, a maximum value of 5 cycles is enforced.

In most cases you should not need to change these values. However, if you are still receiving false triggers on either the start line or end line (and have assured no objects are within the distance and field-of-view of the sensor), you can try increasing this value by one until the false triggers are resolved.

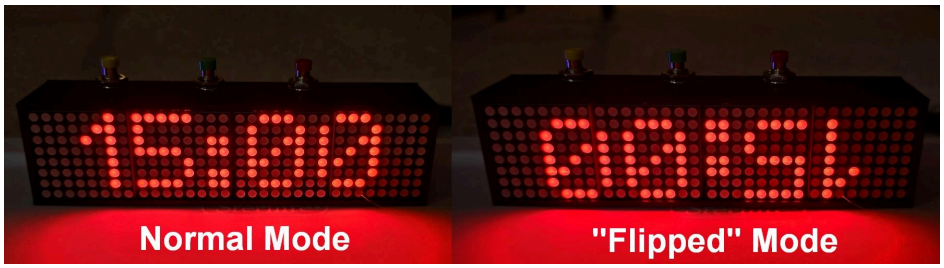
See the appendix for more details on sensor issues and potential solutions.

Timer Settings

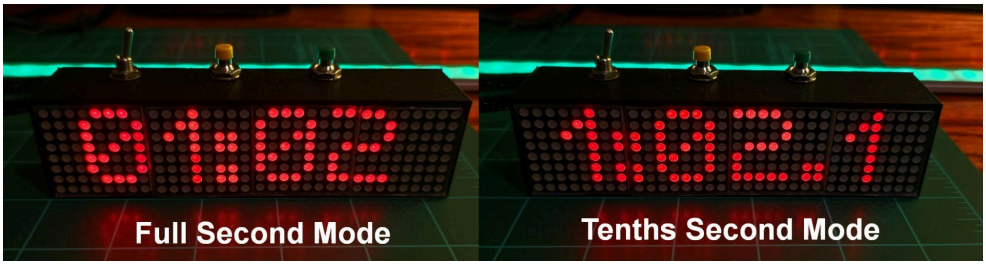
The control the timer display and related timing options:

- **Timer Brightness**: Specify a value from 0 - 10 to set the brightness level of timer's LEDs. Zero is the lowest level, but the display will still be on at this level. There is no "off" setting for the timer display.

- **Flip Timer Display**: Checking this box will flip or invert the display. This can be used if you want to mount the display with the buttons/toggle on the bottom instead of the top.

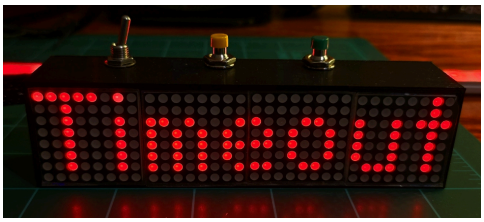


- **Use Tenths Timing**: The timing and timer display can operate in full second mode or in tenths of a second mode.



Both of the above photos show 1 minute and 2 seconds, but the tenths of a second mode also includes the 0.1. Note that when showing tenths, the maximum time that can be displayed is 9:59.9 (9 minutes, 59.9 seconds), but when in full second mode, the timer can display up to 99:59 (99 minutes, 59 seconds). This has a direct impact on the next setting option.

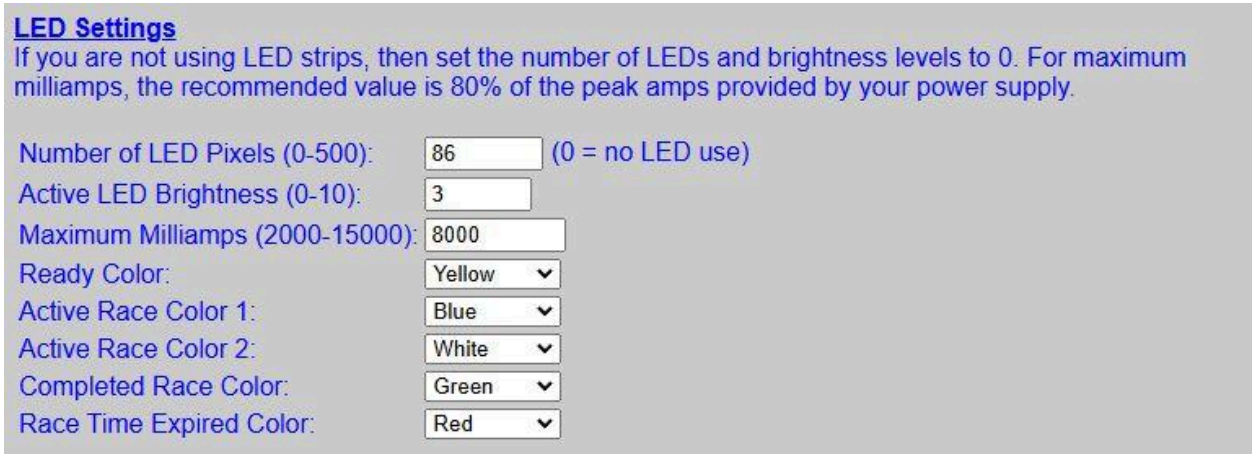
- **Max Allotted Race Time**: You can specify a maximum race time and if exceeded, the timer will stop and show “Timeout”. If LED strips are used, they will also change to the specified timeout color.



The time is specified in seconds (e.g. 300 seconds = 5 minutes). The maximum permissible value is dependent upon the timing mode (full or tenths of a second). When showing full seconds only, you can have a maximum race time of up to 5,999 seconds (99 minutes, 59 seconds). When showing tenths of a second, the maximum race time is limited to 599 seconds (9 minutes, 59 seconds). Regardless of any timeout settings, the timer will always timeout and stop when the maximum displayed value is reached. If you need a race time longer then 10 minutes, then you must use full second mode.

LED Settings

These settings control the LED strip lighting along the track. Note that LED strip lighting is optional and can be turned off via these settings.



- **Number of LED Pixels**: You should specify the number of LEDs in your install. If you are using two strips in parallel, you should specify the number of LEDs ***from one strip only!*** The LED data signal is split and sent to both strips simultaneously, so from the controller standpoint, the number of LEDs are the count from one strip only. If your two strips have different numbers of LEDs, you should enter in the number from the longer strip.

  If you are not using LED strips, or wish to disable them, enter zero (0) for the number of LEDs. In effect, this means the firmware will skip any commands related to LEDs. When a zero is entered for the LED count, all other LED settings are simply ignored and their settings are meaningless.

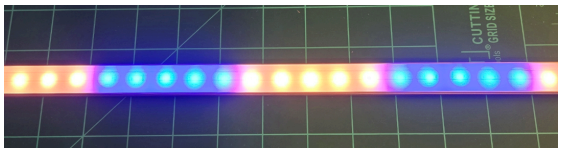
  If using LEDs, then there is a minimum of 1 and a maximum of 500 LEDs (remember this is the number on just ONE strip if using two parallel strips).

- **Active LED Brightness**: This is a value from 0 - 10 and controls how bright the LEDs are when turned on. Using zero effectively turns the LEDs off. The difference between using a zero for brightness and zero for the number of LEDs, is that when a zero is specified for the brightness but there is at least one or more LEDs entered for the number, the controller will still send signals to the LEDs, but they simply won't light up.

  Also note that the higher the brightness, the more current needed by the LEDs and therefore a bigger power supply could be needed. In most situations except possibly outdoors or other bright light situations, an LED brightness of 5 or less should suffice.

- **LED Colors:** You can set the desired LED color for the four various stages of a race as follows. The default values that are initially used are shown in parenthesis, but you can use the settings page to change any of them:

    - *Ready Stage* (yellow): This when the system is armed and ready to begin timing the next race. The timer will also show “Ready”.
    - *Active Race* (blue & white): This is when an active race is occurring and the timer is running. This is the only stage that allows you to specify two different colors for the LEDs, that will alternate:



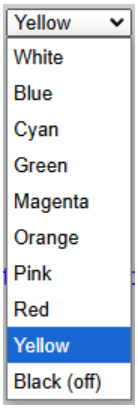
Example showing orange and blue for the two active race colors.

If you wish to only use a single color for the active race, then set both active race segments to the same color.

    - *Race Complete* (green): This is when a car successfully completes the race and the timer stops. When this stage is reached, the system will “freeze” until reset with the timer button, at which point the system and LEDs will return to the Ready stage.

    - Race Time Expired (red): When the maximum allocated race time is reached and the race still has not been completed, the timer will show “Timeout” and the LEDs will turn the color specified here. Like the race complete stage, the system will “freeze” until reset back to ready mode.

Notes on Color Codes: Currently, only solid colors as listed in the dropdown box are available.

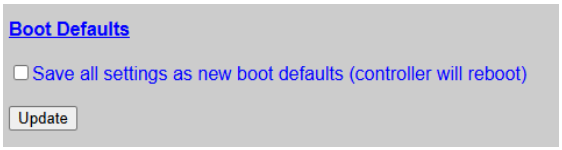


It is simply not possible to run effects or patterns at this time as doing so is processor-intensive and could impact timing accuracy, especially when using tenths of a second timing. Other solid colors can be added if desired by editing the source code.

Saving Settings and Updating Boot Defaults

Making any changes to the settings will **immediately stop any active race if ongoing** and return the system to ready mode. It is recommended that the timer be stopped (end of race, time expired or via the timer's reset button) before making any changes to the settings.

There are two save modes for the settings… temporary and 'permanent'... controlled by a checkbox just above the 'Update' button.

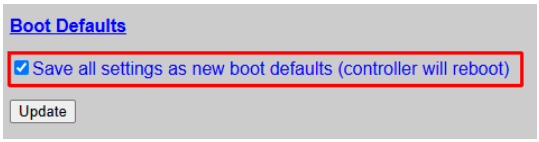


All the settings for the system are saved in a config file, written to the flash storage of the ESP32. Each time the controller starts up, this config file is read and the settings applied. These settings are considered the "boot defaults". However, you can test and try out different settings (LED colors, sensor distances, etc.) without permanently writing these changes to the configuration file.

Temporary Changes: If you want to experiment or are in the process of setting up the system and may be making numerous changes, you can apply the changes without saving them by clicking the 'Update' button **without** checking the box to update the boot settings. Changes will be applied immediately upon clicking the 'Update' button and the web page will show the currently applied settings:

As stated on the page, any temporary settings will remain in effect until the controller is powered off or restarted, at which point the boot defaults will be reloaded. You can return to the settings and continue to make changes until you have the settings you wish to save as new defaults.

'Permanent' (Boot Default) Changes: If you have changed settings and want to save the current values as the new boot defaults, then check the box immediately above the 'Update' button:

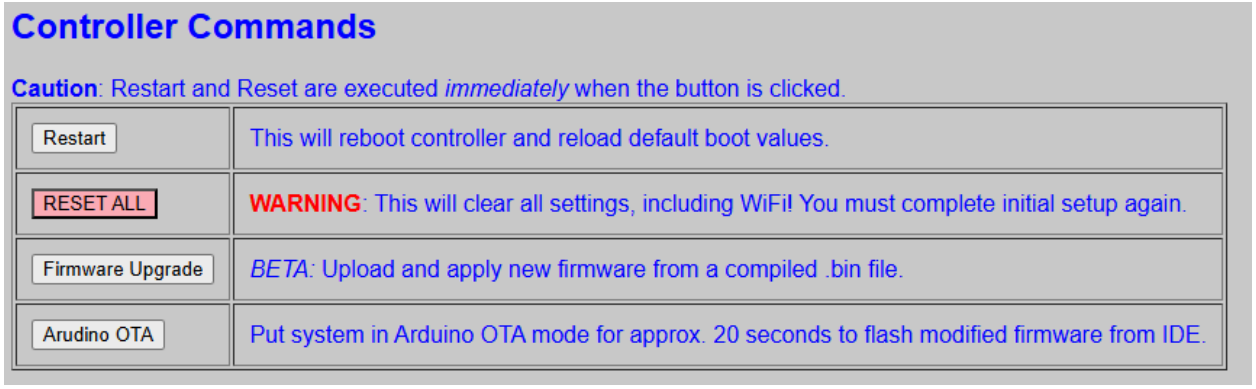


All current settings will be written to the config file, overwriting previous values. The current new settings will be shown and the controller will reboot, loading the new saved settings values.

Once the controller finishes the reboot (the timer display will show "Ready"), you can return to the settings page and make other changes if desired. But the newly saved boot defaults will now be "permanent" and used every time the controller starts or restarts until the boot default settings are overwritten again.

Controller Commands

The Settings page also has a few commands that can be used with the system.



Restarting the Controller

You can reboot or restart the controller by simply clicking this button. Restarting the controller will reload any boot defaults, so if you have made temporary settings changes, note that these will be lost with a controller restart. The restart will begin immediately upon clicking the button and there is no confirmation prompt. A message will be shown on the web page confirming that the controller is restarting. After the restart completes (shown by 'Ready' on the timer display), you can return to the settings page and make changes if desired. Otherwise, the system is ready for use.

Resetting the Controller

Clicking RESET ALL will wipe all settings, ***including Wifi and the device name!*** After this step, you must complete the onboarding step again, including providing your WiFi information and specifying the other onboarding info. All other boot defaults will revert back to the original values. See the section above on Onboarding and Initial setup for these steps. ***Note:*** The reset process executes as soon as you click the button and there is no confirmation prompt, so be sure this is what you want to do. However, resetting the controller is the only way to change WiFi information or the device name as these settings are not available via the settings page.

Firmware Upgrade

You can apply new compiled versions of the firmware (.bin) directly to the controller right from the web page over-the-air. The device being used must be on the same WiFi network and the new .bin file must be locally available to that device. This is currently considered a beta feature, but testing has been successful so far. More info on the various methods of applying firmware updates can be found in the appendix.

Arduino OTA

This will place the controller into Arduino OTA mode for approximately 20 seconds so that modified or custom firmware can be sent wireless to the controller using the Arduino OTA functionality. See Uploading Modified Code via Arduino OTA in the appendix for more info on using Arduino OTA for flashing firmware.

URL Commands

The bottom of the settings page lists a few URL commands that can be issued directly to the controller. For the most part, these are leftover commands that existed before the settings page was added.

You may also issue the following commands directly via your browser to make changes:

Restart/Reboot the Controller:	http://192.168.1.232/restart
Reset Device - Remove all settings (you must onboard again):	http://192.168.1.232/reset
Arduino OTA Update - put device into OTA update mode:	http://192.168.1.232/otaupdate
LED Brightness*:	http://192.168.1.232/leds?brightness=x (where x = 1 to 10)
Timer Brightness*:	http://192.168.1.232/timer?brightness=x (where x = 1 to 10)

**Changes to brightness values using this method are temporary and will reset when the controller is restarted.*

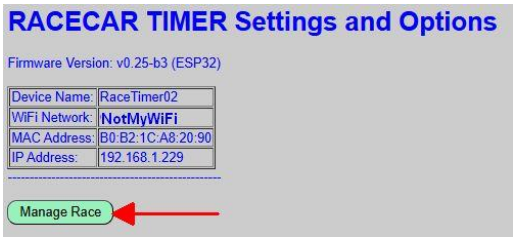
Your settings page will actually show the IP address to use for your controller as opposed to the IP address shown in the above example. Since you can now restart/reset the controller, launch Arduino OTA updates and change the brightness levels from the web settings page, there really isn't much use for these commands… but they do still work.

Note that the Arduino OTA option is used to put the controller into Arduino OTA update mode when flashing modified or custom firmware using Arduino over-the-air updates. This mode is indicated by alternating red and green LEDs (if installed) and a message of 'Upload' on the timer display.

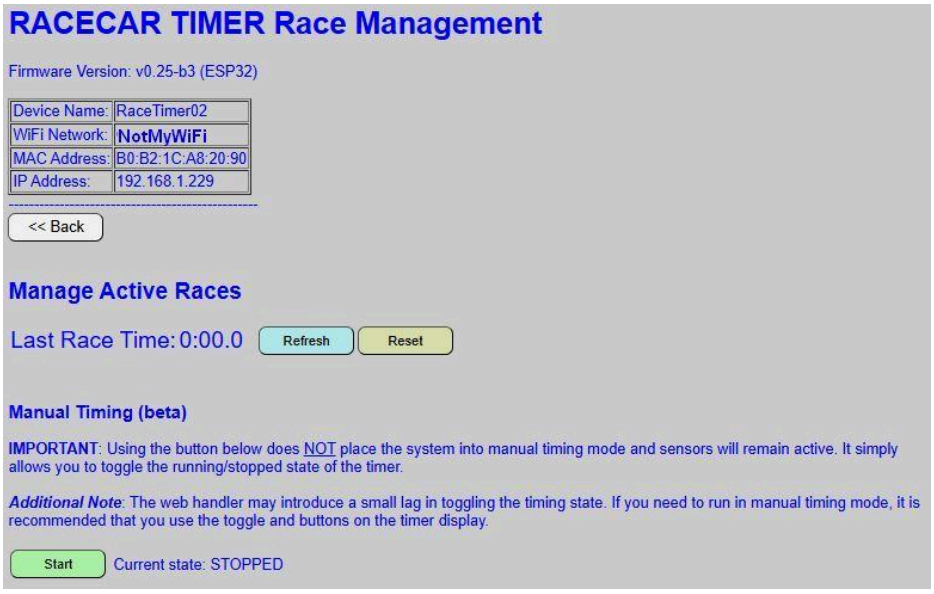
See the appendix for more information on both applying firmware updates and using the Arduino IDE to make source code changes and uploading your new version to the controller.

Race Management Page

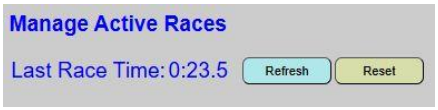
A new race management page was added starting with release v0.25. This new page is accessible via a button on the main page.



The purpose of this page is to provide some of the same controls as the physical timer, but via the web interface.



- **Last Race Time:** This will display the last race time, or the time currently displayed on the timer. This time is ‘static’, meaning it does not automatically update as the timer is running. Instead, click the ‘Refresh’ button to obtain and display the time. Normally this would be done after a race is completed so that the race time can be recorded.



- **Reset:** Pressing this button will immediately place the system back in pre-race “Ready” mode for timing the next race. This action is equivalent to pressing the physical ‘Reset’ button on the timer display.

Manual Timing (beta)

This feature is still considered “in beta” but it allows you to manually start/stop the timer via the web interface.

There are a few very important notes about using this feature (and while it is still considered a “beta” feature):

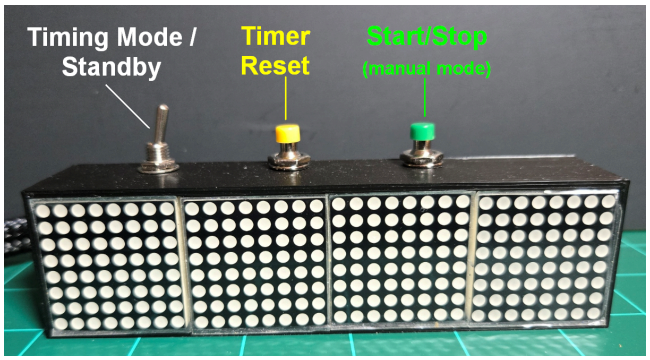
- The start/stop function can be used whether the system is set to auto or manual timing (via the timer toggle). But if used when in auto mode, the ***start and stop sensors are still active*** which means that triggering the sensor may also start/stop the timing. If you only want to use the web buttons to start/stop the timer, then set the timing mode toggle to ‘Manual’.
- The current state is not automatically updated but instead only shows the timer state at the point that the web page loads. Pressing any buttons on the page will refresh the displayed values, including the timer state. You can also just reload the page from the browser controls.
- A small lag may be introduced to the timing as the controller has to “break” its normal loop to process the web request. This could lead to a small delay with the timer starting or stopping and thereby impact the timing results.

From an accuracy perspective, the best results will be received when using the sensors in auto timing mode. If manual timing is needed, the next best option for accuracy is to set the system to manual mode and use the start/stop button on the timer display.

But this page does give you the same general options as the buttons on the timer display.

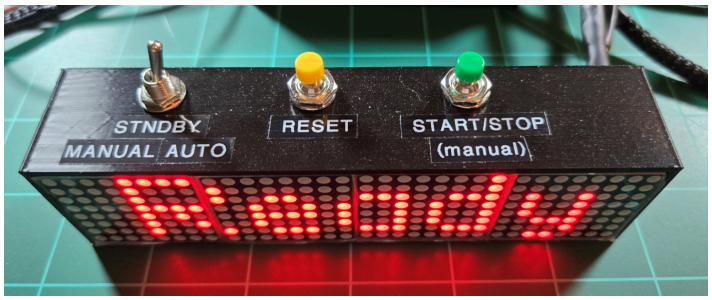
Using the System

Once the system has been set up and configured via the settings, use of the system is very easy and straightforward. The only controls used outside of the settings web page are found on the timer display.



Timing Modes

The system can be used in either automatic or manual timing modes. This is controlled by the three-position toggle switch on top of the timer display.



When set to the “inside” position (as shown in the image above), the system is in auto-timing mode. When to the far opposite position, timing will be manual. Placing the toggle in the center position puts the system in standby mode.

Automatic Timing Mode

When the system is in automatic timing mode, the timer automatically starts when the starting line sensor is tripped. Time automatically stops when the finish line sensor is tripped. Note that the sensors are only active based on the current mode. See the section above under Placing and Aligning the Sensors to understand how the sensors work and when they are active vs. inactive.

When an active race ends, either via crossing the finish line or via a timeout when the max allocated race time is reached, the timer will stop and the LED strips (if used) will either display the race completed or the time out color. At this point, the system is “frozen” and the final time (or Timeout message) will remain displayed on the timer. The sensors will both be inactive at this point.

Use the Timer Reset button on the timer to reset the system and put it back into ‘Ready’ mode for the next race. When in automatic timing mode, the Start/Stop button is not used and is inactive.

Manual Timing Mode

When the toggle on the timer display is set to manual mode, all sensors are ignored at all times. Timing will instead be manual and will start/stop based on presses of the Start/Stop button. This lets the system function much like a stopwatch. There are a couple of additional points for which you should be aware when using manual mode.

First, the start button will only start the timer from ‘Ready’ mode. Next, once the time is running and is then stopped by pressing the Start/Stop button a second time, the time cannot be restarted from the existing time. Once stopped, the time can only be started again by first pressing the timer resest button, clearing the existing time and putting the system back into ‘Ready’ mode.

Manual timing mode is provided if there are unforeseen issues with the sensors. If the sensors are misbehaving, the mode can be flipped to manual and continue to be used.

The LEDs, if installed, will still change colors appropriately based on race stage even when using manual timing mode.

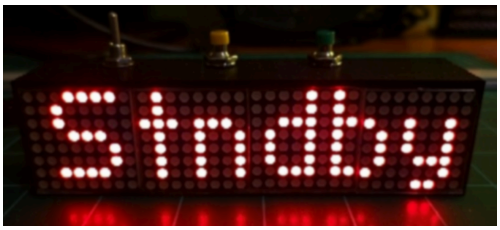
Resetting the Timer

Resetting the timer after a race completes or times out is required before starting the next race as described above, but it can be used at any time, even during an active race to immediately reset the time and put the system back into Ready mode. This might be used if a racer faults or leaves the course without crossing the finish line. Remember that when in auto timing mode, the Start/Stop button is not used and is inactive. So in this situation if you need to stop an active running timer, simply use the Reset button.

Standby Mode

Moving the timer display toggle switch to the center position will immediately place the system in Standby Mode. Standby temporarily suspends the system while keeping the controller running and all current settings intact. When in standby, all sensors and buttons are disabled. Any LED

strip lighting will be turned off, and after showing a brief standby message, the timer display will als0 be turned off.



Standby might be used when a short break occurs and you don't want to power down the entire system and lose any temporary setting changes that may have been made. The following table lists the various features and controls and their availability in the various timing modes:

	Auto Timing	Standby	Manual Timing
Web Settings Page	✓	✓	✓
Timer Mode Toggle Switch	✓	✓	✓
Start/Finish Line Sensors	✓	✗	✗
Timer Reset Button	✓	✗	✓
Timer Start/Stop Button	✗	✗	✓
Timer	✓	✗[1]	✓
LED Lighting	✓	✗[1]	✓

1 - LEDs and timer will still be used for ArduinoOTA mode, but off for all other functions

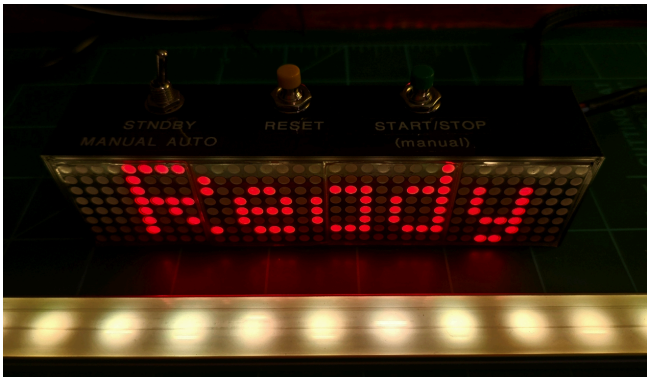
Simply flip the toggle back to auto or manual timing mode to restore the system immediately back to 'Ready' mode. Note that toggling out of standby is instantaneous, while powering down the controller and restarting it takes a few moments as the controller boots up, loads default settings and connects to WiFi. Using 'Standby' can significantly shorten the time before the system is ready, especially when the system may be intermittently used throughout a given time frame.

Running a Typical Auto-Timed Race

For these examples, I'll be using the following options and LED colors:

- Use Tenths Timer: YES
- Max Allotted Race Time: 300 seconds (5 minutes)
- Ready Color: Yellow
- Active Race Colors: Blue and White
- Completed Race Color: Green
- Race Time Expired Color: Red

For an auto-timed race, the timer display toggle switch must be placed in auto-mode. The system should be in "Ready" mode, with this displayed on the timer. The LEDs, if used, will show the ready color (yellow in my case):



If the system is not in ready mode, simply press the reset button on the timer to place the system in ready mode. At this point, the starting line sensor is active.

Pre-Race (Ready Mode)

The racer may stage his/her car behind the start line and may move freely in the area around the track, as long as the car or any other object does not touch the start line or cross the area immediately in front of the start line.

When the racer is ready, they can simply start moving forward. As soon as the car (or other object) touches the start line, the system will automatically move the system to active race mode.

If the sensor gets tripped and the timer starts inadvertently, simply press the timer Reset button to cancel the active timer and place the system back into Pre-Race/Ready mode.

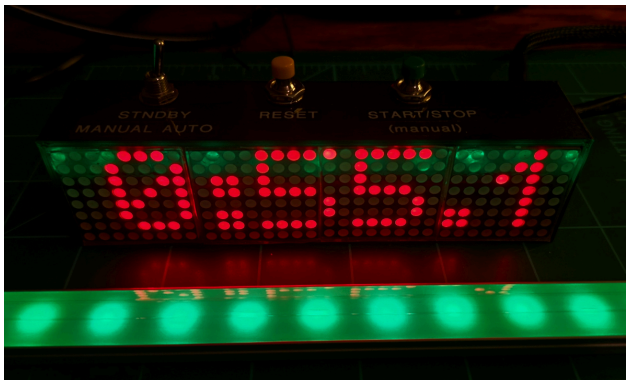
Active Race Mode

As soon as the car or other object touches the start line or enters the area immediately in front of the start line area, the timer will immediately start and the LEDs will switch to the active race color(s) - blue and white here:

At this point, the start line timer is inactive and the finish line time becomes active. The timer will continue to run until the car or object touches the finish line, an object enters the area immediately beyond the finish line, the timer reaches the maximum allotted race time or the maximum time that can be shown (9:59 when using tenths timing and 99:59 when not), or the system is reset via the timer reset button.

Completed Race Mode

When the car or object reaches the finish line, the time will automatically stop and the LEDs will switch to the completed race color:



At this point, both sensors are inactive and the system is “frozen” so that the time may be recorded. When ready, use the timer reset button to place the system back into ‘Ready’ mode for the next race.

Timing Out

If the time exceeds the maximum allotted race time, the timer will stop and display ‘Timeout’ and the LEDs will change to race time expired color:

Much like when a race is successfully completed, the system will “freeze” at this point with both sensors being inactive. To reset the system for the next race, simply press the reset button on the timer.

Aborting a Race

If for any reason, a race needs to be halted after timing has begun, simply press the reset button on the timer. This will immediately stop the active race, reset the timer back to 0:00 and place the system back in ‘Ready’ mode. This might be needed if a racer fouls, false starts, leaves the course, etc. and you want to reset the system before a car crosses the finish line or the system times out.

Running a Typical Manually Timed Race

The process and modes used for manual timing is much the same as the auto method. The primary difference is that the start and finish line sensors are always disabled for all portions of a race. Instead the timer Start/Stop button is used to manually control the timer in lieu of the sensors.

Pre-Race (Ready Mode)

This is identical to the auto-timing mode with the exception that the start line sensor is disabled. For this reason, movement anywhere near or on the track including crossing the start line will not trigger or start a race. Instead the timer Start/Stop button will be used.

Active Race Mode

Active race mode is entered as soon as the timer Start/Stop button is pressed. The timer will start and the LEDs, if used, will change to the active race color(s). Time will continue to run until stopped by a second press of the Start/Stop button or either the maximum allocated race time is exceeded or the time exceeds the display capacity of the timer (9:59 when using tenths, 99:59 when not).

Completed Race Mode

A race is considered complete when the Start/Stop button is pressed when the timer is running. The time will not automatically stop based on either sensor. Just like auto mode, when the race is complete, the system is “frozen” with the current time and showing the completed race LED colors. The system must be reset via the timer reset button to put the system back into ready mode for the next timing session.

Important Note: As of v0.22 of the firmware, once the timer is manually stopped after running, it cannot be resumed from the current stopped time. Instead, once the time is stopped, the system must be reset and timing will restart with 0:00 the next time the start button is pressed. It is planned that this may be changed in a future release to allow elapsed time to be resumed. Check the Github Wiki to see if a future version includes this feature.

Timing Out

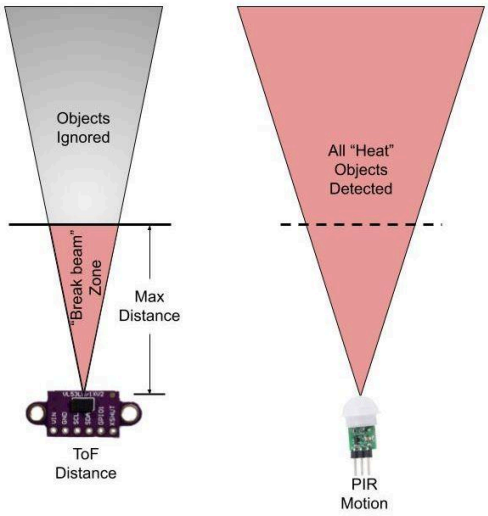
This is identical to the auto timing mode. See above.

Aborting a Race

This is identical to the auto timing mode. See above. The system may be reset at any time, whether the timer is running or stopped.

Other Potential System Uses

While this particular system was designed for timing robot cars, the concepts can be used for many other types of projects. Using the VL53L0x and limiting the trigger distance (instead of a standard PIR or other motion detector), a sort of ‘break beam’ detector can be created that will only trigger when a distance less than the maximum is detected.



Example: Not to scale

Even if the field-of-view angle of the PIR sensor is limited via fins, tube, paint, tape, etc., motion will still be detected out to the max range of the sensor, well beyond the desired range. In addition, PIR sensors use passive IR to detect heat signatures. That means that if trying to detect non-living object (like a robot race car), a PIR sensor simply won’t work. You could use something like a mmWave motion sensor, some of which even allow you to limit the range, but these generally detect movement in 360°, which introduces its own problems in many situations (like a race track which may have movement all around the area).

But creating a “break beam” style sensor can act as a trigger for all sorts of actions. While two sensors are being used here to start and stop a timer, and to control LED lighting strips, they can act as triggers for all sorts of automations or routines. I use two VL53L0x sensors in a break beam style configuration to control the LED lighting on my stairs, which replaced the original PIR sensors and eliminated all the false triggers I was getting with the PIR sensors when someone just walked near, but not on, the stairs.

Of course the timer and timer display has a multitude of other potential uses, especially with the manual start/stop option.

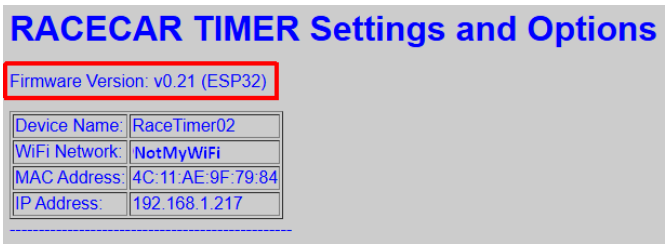
Naturally, other uses will likely require modification of the source code for another purpose.

Appendices

The appendices provide additional information that may prove useful in some situations or covers additional information that isn't necessarily part of the normal operation.

Firmware Updates

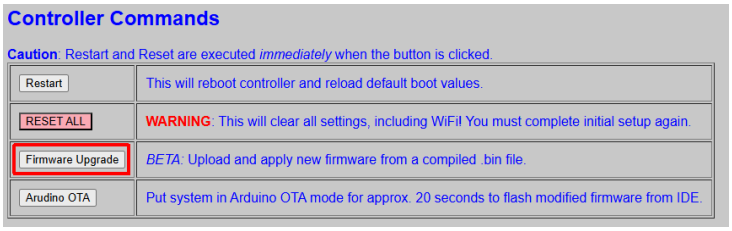
The current version of firmware can always be found at the top of the web settings page:



When new firmware versions are released (these will always be released via Github), a compiled .bin file will be provided that can be flashed onto the controller's ESP32. There are multiple methods for doing this. Also note that regardless of the method, the firmware must be compiled into a .bin file before uploading. You cannot upload the raw source code (.ino) file to the ESP board…. well, you can but it won't work!

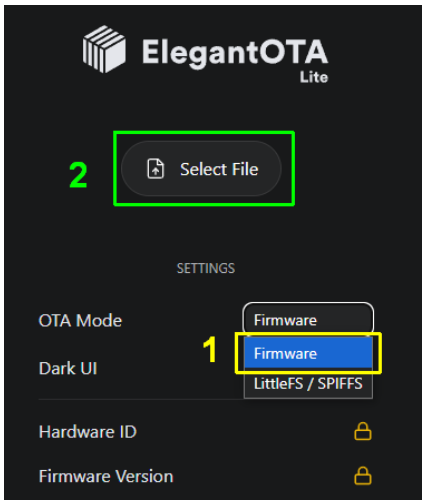
Using the Web Settings Page

Starting with version 0.21, a beta version of a firmware update process is available right from the controller's web interface:



This will currently launch a separate applet in your browser for installing the new firmware.

Note that this process is currently considered in “beta” and may change with future releases.

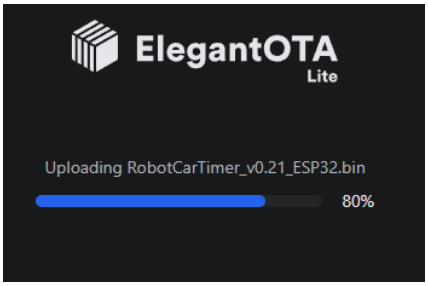


First assure the OTA Mode selected is “Firmware”. This should be the default and you should not need to change it. Unless otherwise noted in the release notes, a firmware update should not overwrite your configuration file and all your settings, including WiFi connection information should remain intact.

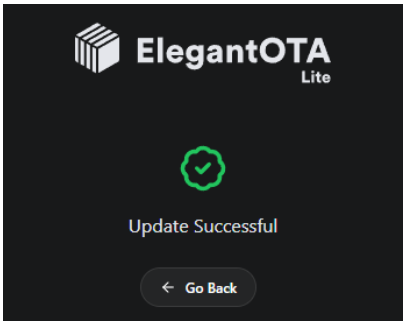
Note that LittleFS is where your configuration file is located. Inadvertently selecting LittleFS/SPIFFS as the OTA mode will try to write the firmware into the space where the configuration file is stored. Not only will this overwrite your configuration, which would require onboarding again to recreate the file and restore WiFi credentials, there is also a possibility that writing the firmware to LittleFS will corrupt/fail, rendering the board unreachable wirelessly. If this occurs, you will need to complete the initial flashing via USB as described at the very start of this guide. So unless otherwise instructed in the release notes, always assure Firmware is selected as the OTA Mode.

Click the ‘Select File’ button and browse to the downloaded firmware .bin file and select it.

NOTE: The firmware update will immediately begin as soon as you select a .bin file without further confirmation, so be sure that an update is what you want and that you’ve selected the proper .bin file.



A progress bar will show the progress of the upload and installation.



When the update completes successfully, the controller will automatically reboot. After the reboot completes (visible via timer/LEDs if connected or via the blue LED on the ESP32 if not), you can return to the settings page. The settings page should reflect the newly installed version of firmware.

Using a Desktop Utility

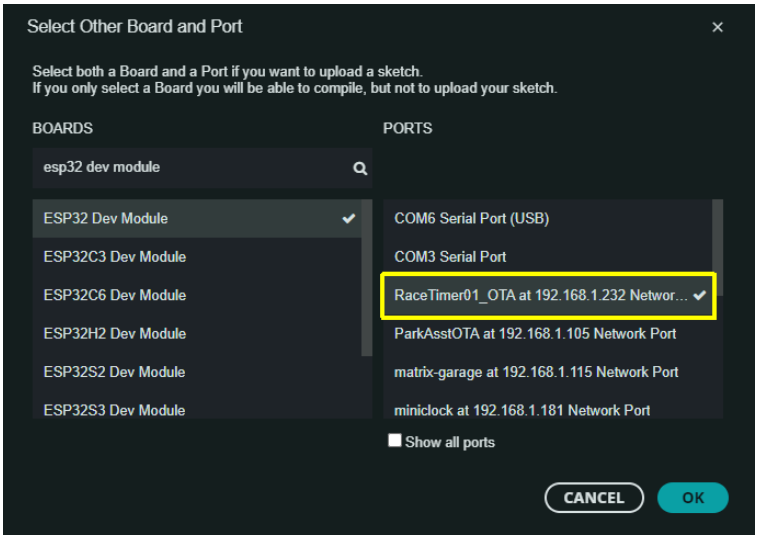
If for some reason you are unable to reach the controller wirelessly over WiFi or are installing the firmware on a new ESP32, you may install firmware updates via USB cable and a desktop flashing utility. Note that this process generally includes erasing the flash on the ESP device, meaning you will likely lose all your local settings, including WiFi credentials. You will have to complete the initial onboarding and reconfigure all your changed settings after flashing firmware using this method.

The method for flashing over USB with a desktop utility is covered in the very first section of this guide. Please refer to that section for step-by-step details.

Using the Arduino IDE

The firmware comes with built-in support for using the ArduinoOTA for sending firmware directly from the IDE to the ESP32 wirelessly over-the-air. Note that the Arduino IDE must be configured for use with the ESP32, have all the necessary libraries installed for the timing system and you will need to compile the firmware within the IDE before sending to the ESP32. If

your device is on WiFi, it will be broadcasting an OTA port consisting of your device's name followed by _OTA:



Arduino OTA Port for the 'RaceTimer01' Device Name

There are a few other requirements, which are covered under modifying the source code section below. This section also covers the process of using the ArduinoOTA functionality with the timing system.

Modifying the Source Code

If you need or desire to make changes to the source code, you can do so with the Arduino IDE and then compile and upload your changes directly to the ESP32 controller wirelessly. Note that you can also use other platforms such as PlatformIO, but I will only be covering the Arduino IDE in this guide. If you are making substantial changes, you probably already have familiarity with writing or modifying C++. But what if you aren't and only need to make a few small changes… such as swapping the starting and ending sensor GPIO pins so that the cables are reversed from the default. Don't fear! You don't need to be a C++ programmer… or a coder at all… to make minor source code changes. If you've never used the Arduino IDE, I recommend you watch this video I created just for folks like you!

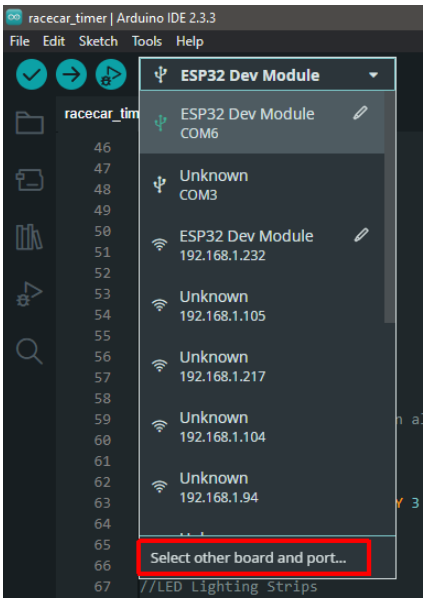
Direct link: https://youtu.be/KS5HOJat88k

This video will get you all set up for making your small changes. Just assure you've installed the particular libraries for this program. The code documents the libraries and versions you need via the #include statements at the top. Make any necessary changes and verify your code.

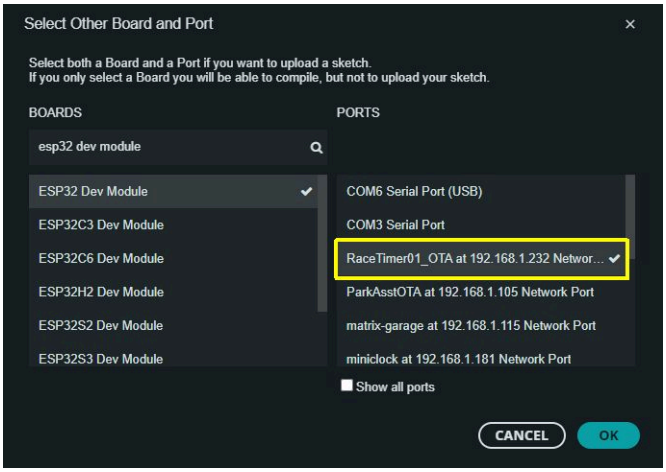
Uploading Modified Code via Arduino OTA

As long as a prior version of the firmware already exists on the ESP32, you can then upload your modified and verified program wirelessly over-the-air to the ESP32. But there are a couple of steps you must complete first.

You must select the OTA port in the Arduino IDE so that communication can be established.



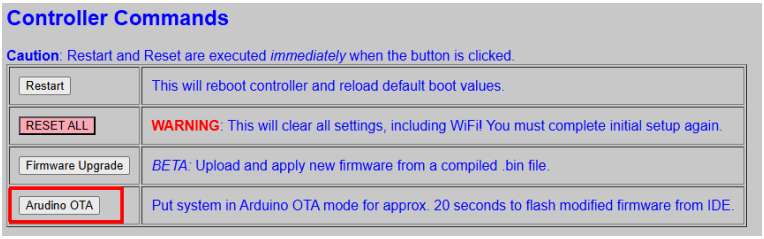
Drop down the board and port selection box at the top of the IDE and choose the option for selecting other board and port.



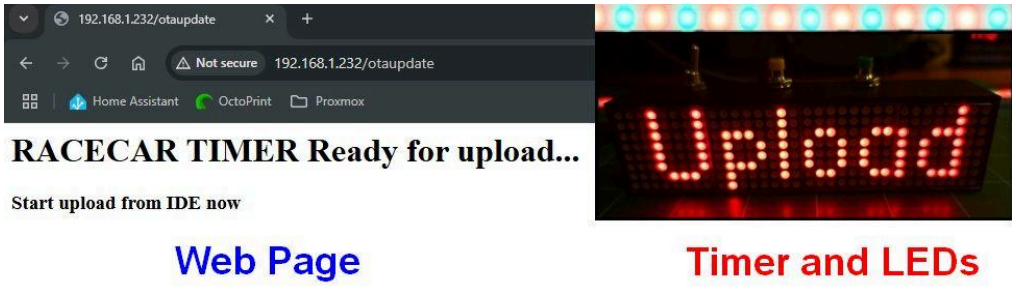
You should see a port listed that consists of the device name you assigned to the controller during initial onboarding followed by OTA for over-the-air. My example device was named RaceTimer01 so the OTA port is called RaceTimer01_OTA. Select this port and while here, also assure you have the proper ESP board selected.

This is yet another reason why you should give your device a meaningful name when onboarding. If you end up with multiple timing systems or a lot of Arduino-based devices that utilize the OTA function, you could see a lot of different ports being broadcast (like in my example above). So a meaningful device name will help you differentiate between different ESP boards that might be in use.

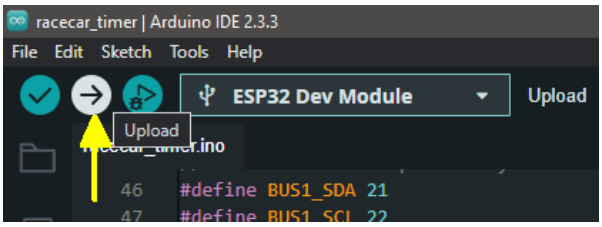
Now the IDE is ready to compile and send your source code file, but before that can happen, the controller must also be placed in OTA mode. The easiest way to accomplish this is to click the Arduino OTA button under the Controller Commands on the web settings page:



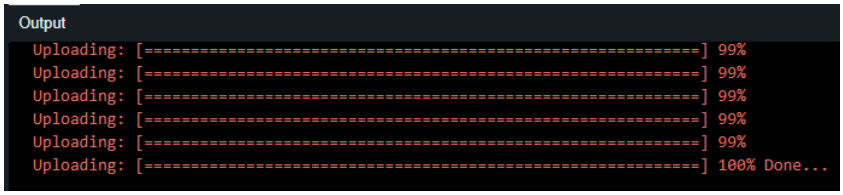
It is recommended that you stop any active races and put the system in Ready or Race Complete mode before issuing the above command. As soon as the controller processes this statement, it will stop any current activity and enter OTA mode, waiting on the new code.



The web site will respond with a message that the system is ready for upload. The timer will also display an Upload message and the LEDs, if used, will change to alternating red and green pixels. You are now ready to send your updated code:



You can watch the output window of the Arduino IDE for the progress of the compilation and upload.



Sample output from the Arduino IDE

Once the upload completes successfully, the controller will reboot and load the new firmware. Once the boot process completes, you can return to the main settings page, check the version number and assure that your update has been applied.

Notes on Timing:

When the controller is placed into OTA Mode via the /otaupdate command, it will remain in OTA mode for approximately 20 seconds. If code doesn't begin sending before this time expires, it exits OTA mode and returns to normal operation.

Depending upon a number of factors, including the speed of your machine, the size of the code and the number of external libraries involved, it might take the Arduino IDE longer than 20 seconds to compile the code and begin sending it. If you find that the OTA mode is timing out on the controller before the IDE can compile and send the code, use the following method:

1. Start the upload via the Arduino IDE and watch the compilation progress in the lower right corner of the IDE:

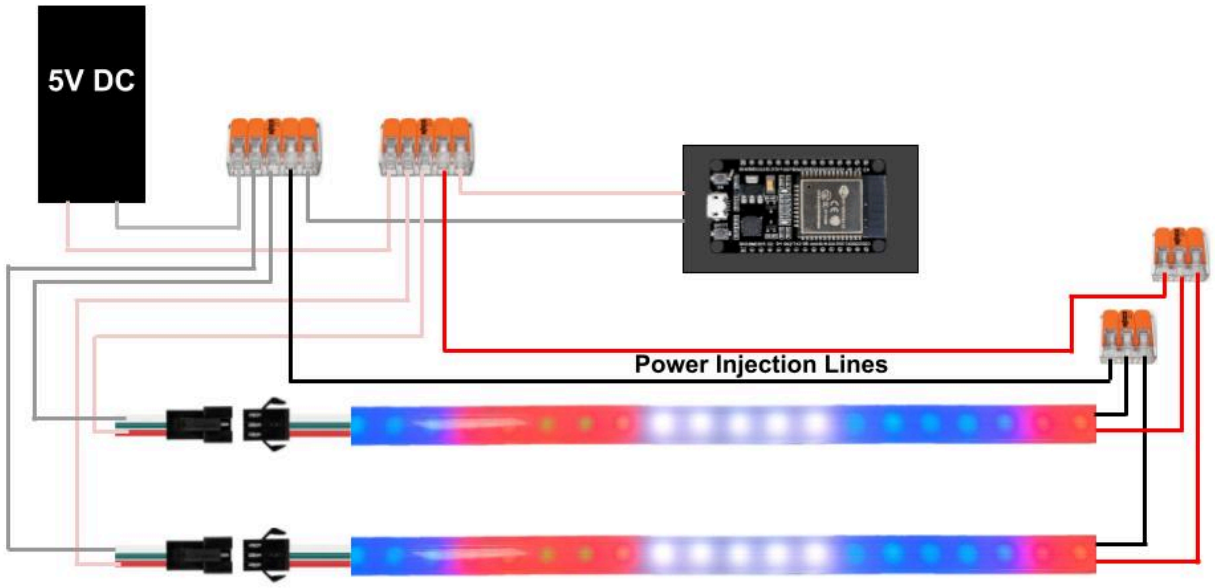
2. When the compilation progress reaches somewhere in the range indicated above, then issue the /otaupdate command in your browser to put the controller into OTA mode. The compilation should complete and the IDE should start sending the code update before the OTA mode times out on the controller.

You may need to adjust the above range for your particular circumstances, but after you've done it once or twice, you'll get a feel for when to launch the Arduino OTA Update command..

Power Injection

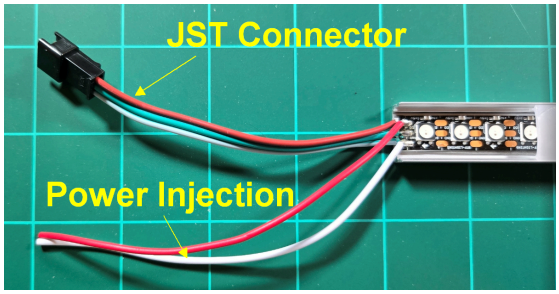
If your build includes LED strips and if those strips reach a certain length/number of LEDs, it is entirely possible that you may see shifting or fading colors, or even flickering/misbehaving LEDs towards the end of the strips, it is likely that due to voltage drop, the end LEDs are not receiving enough power to full illuminate. Reducing the brightness level can help in some cases if the effect is minor, but at some point power injection may be needed to provide additional power to the LEDs at the end of the strip(s).

Power injection simply takes power from the power supply and runs it to the end of the LED strips, connecting to the +5V/GND pads at the end of the strip.



This is the original power wiring diagram from earlier in this guide (the pre-existing wiring is dimmed). You can see where two new lines… one +5V and the other GND… have been run from the power supply to the end of the LED strips. These wires are attached to the +5/GND on the LED strip(s). Note that you do not need to extend or include the LED data line… just the power lines.

The lines, which can be hidden along the side of the track, can be soldered directly to the pads on the LED strips, or if a JST connector or additional power injection wires are already attached to the end of the LED strip, you can just connect your wiring directly to these wires.



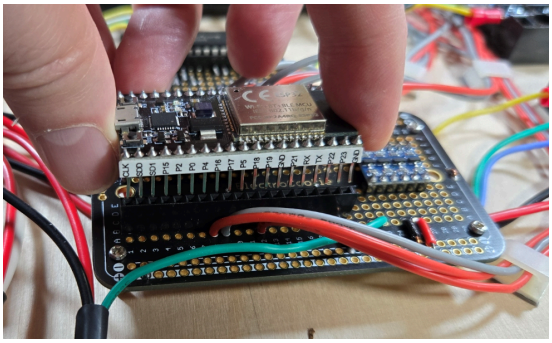
Alternatively, you could use 12V or even 24V LED strips which require power injection much less frequently (in terms of length/number of LEDs) but the controller will only operate on 5V, so you are looking at using two different power supplies (creating a common ground) or by using something like a buck converter to step down the higher LED voltage to 5V for powering the controller and related peripherals. In addition, changing the LED type may also require modification of the source code to work with the different LED types.

But misbehaving LEDs are generally the result of one of two issues:

- A weak or noisy LED data signal - add a logic level shifter.
- Voltage drop due to length/number of LEDs - add power injection to the end of strip

Removing or Replacing the ESP32

If built as shown, the ESP32 should be installed within the main controller via header pins.



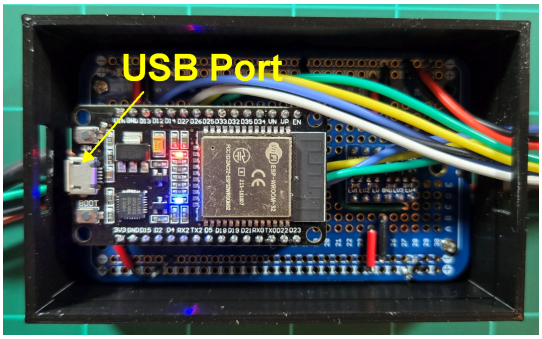
This means that the ESP32 can easily be removed and reinserted, or even replaced, without disturbing the rest of the controller or system. Some reasons why you may wish to remove the ESP32:

- Due to a firmware or other issue, the board is no longer reachable wirelessly. You can remove the ESP32 and reflash via USB.
- You wish to experiment with the source code or try some of your own modifications, but don't want to risk messing up the current working version. You can use two ESP32s, one with the 'production' firmware and one that is your test or development ESP32. You

can easily swap them out while you experiment without fear of leaving the system non-functional.

- The existing ESP32 simply dies or quits working. You can just prepare a new ESP32 by following the initial installation and onboarding steps from this guide.

Just use caution when inserting the ESP32 and assure that the pins line up precisely with the header. ***Also be assured not to reverse the ESP32 and insert it backwards!***



The system is designed so that the USB port on the ESP32 should always point **away** from the center of the board and towards the outside edge of the board. Reversing the board may damage or destroy the ESP32 as incoming voltage would be applied directly to GPIO pins.

Troubleshooting

While the system has been thoroughly tested, there are many things that could go wrong… from faulty components to incorrect wiring or soldering to actual undiscovered bugs in the firmware. But before posting and issue or question, try some of these troubleshooting steps that cover some of the most common issues that folks may experience.

Initial Flash of the Firmware

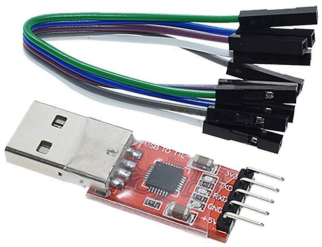
If you cannot complete the initial upload of the firmware via USB and a desktop utility, there could be a number of reasons… and potential solutions.

A new COM port does not appear when I connect the ESP32 to my computer.

If a COM port does not appear, you may not have the proper USB to UART/Seria drivers installed. These generally come preinstalled for most Windows 10/11 and Mac OSes, but if not, you can generally resolve the issue just by installing the driver(s). The driver needed may vary based on the ESP32's onboard USB chip, but CP2102 and CH340 are the most common. Check your board's documentation. Here are links to the various drivers if needed (current at time of publication):

- CP2102 Drivers: Windows & Mac
- CH342, CH343, CH9102 Drivers: Windows or Mac
- CH340, CH341 Drivers: Windows or Mac

If you cannot get the drivers installed or still do not see a new COM port appear when connecting the ESP32, you may need to resort to using an external USB-to-UART programmer like this one:

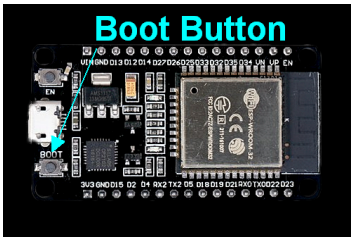


USB-to-Serial Adapter Example: https://amzn.to/3UAMpkK

If you are still unable to establish a connection, try the steps in Espressif's official guide.

The flashing utility cannot connect to the ESP32

If you've connected your ESP32 and selected the new COM port but when you try to flash the firmware, the utility just continues to try to connect until it times out. The board must be placed into flashing mode, generally by pulling GPIO0 to ground. Some ESP32 boards cannot be placed into flashing mode via software. But the BOOT button found on most boards is connected to GPIO0 and you can place the board into flashing mode by pressing and holding this button.



After connecting to the COM port in the flashing utility, press and hold the boot button. Continue to hold this button and start the flash. Once a connection is established, you can release the button.

If this still doesn't work, you may need to press and hold the button when connecting the ESP32 to the USB port (pressed when power is applied). If this doesn't work, then press and hold the button when connecting the USB cable, but ***continue to hold the boot button down*** while you start the firmware flashing launch. Again, once you see that communication is established, you can release the button.

Are you flashing a compiled .bin file? You cannot flash a source code .ino file to the board. Or if it does actually write to the board, the firmware will not function. ***You must flash a compiled .bin file and not the raw source code .ino file!***

If all else fails, try a different flashing utility. I've had situations where one utility couldn't flash a file to a particular ESP32 while a different utility had no problem. You can also try a different ESP32. I have received batches of ESP32 boards where only a couple out of a half dozen could be flashed.

Onboarding and Initial Configuration Issues

If you've completed the initial flash of the board but yet to onboard it to your WiFi, the controller should begin broadcasting a local hotspot, called **RaceTimer_AP**. You will connect a mobile or other device to this hotspot for entering WiFi and other initial info as per on the onboarding instructions in this guide.

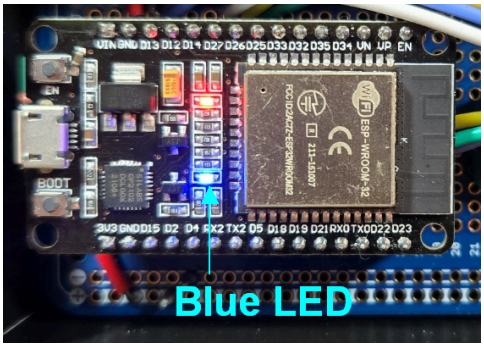
No RaceTimer_AP hotspot

- Assure you are near the controller… ideally within 10 feet. The ESP32 does not broadcast a strong signal, so you must be nearby.
- Try power cycling the ESP32. It may still be in flashing mode.
- Wait a few minutes and refresh the mobile device's available WiFi connections.
- Try a different mobile device if possible.

If you still do not see the hotspot, there may have been an issue with the flash. Try flashing again or following some of the troubleshooting steps under initial flashing above.

ESP32 does not appear to join WiFi

Even if the ESP32 is not yet installed in the controller, there are a few easy ways to check if the onboarding worked and the ESP32 has joined your WiFi. The first and simplest is to simply watch the ESP32 when it is powered on.

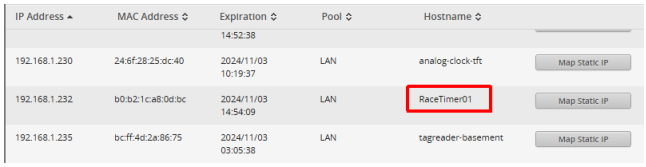


The ESP32 has two onboard LEDs (most versions). The red led is the power indicator and will come on as soon as power is supplied. However, the blue LED is used to indicate a successful connection to WiFi. After applying power via USB port *or* via the controller board (**but never both at the same time**), watch the blue LED. If the boot process completes successfully and the ESP32 connects to your WiFi, the blue LED will blink three times and then remain lit.

Other ways that you can check whether the ESP32 has joined your WiFi or not:

- Is the **RaceTimer_AP** still being broadcast? This is only visible when the ESP32 cannot connect to WiFi. If you see the hotspot, try onboarding again, assuring your WiFi credentials (which are case-sensitive) are correct.

- Check your router or network utility to see if the ESP32 device has joined the network. The firmware will use the device name you entered during onboarding as the WiFi hostname (not all routers will recognize this):



Example device with an onboarding name of 'RaceTimer01'

- If you do see the device on the network, try entering the IP address into a browser of a device on the same WiFi network. This should display the web settings page and if so, you know the device is on your network.

- Assure you do not have firewall or other network rules that may be preventing the controller from joining or that the device is being placed in an incorrect VLAN.

If you see the blue LED flash three times and remain on, the boot process is completing successfully and the board is joining your WiFi. If the timer, LEDs or sensors are not working as expected, then that is a different issue. Check other parts of this troubleshooting guide.

If you do not see the flashing blue LED and it never illuminates, then the board is either not joining WiFi (and the RaceTimer_AP will be visible) or there is some other issue with the boot sequence. The first step is to try power cycling the ESP32/Controller a few times.

If none of the above work, you may have an invalid or corrupted flash. Try repeating the flash and following the troubleshooting steps until initial flashing.

If you are still unable to resolve the issue, you may need to try a different ESP32 or even a different brand/manufacturer. While the firmware should work with any standard ESP32 deb board, unfortunately there are manufacturers of these low cost components that cut corners with cheaper or non standard components/configurations that may prevent a particular board working with particular firmware. It is simply impossible to test every type of ESP32.

No boot sequence shown on timer or LEDs

If you've established that the controller is completing the boot process via either the blue LED on the ESP32 or by finding the device on your network, but you don't see any output or the expected boot sequence indications on the timer and LEDs (if installed) as described earlier in this guide, try the following:

- The first step is to alway try a couple of power cycles of the system. On rare occasions, a device may not initialize properly when power is first applied.
- Use the web settings page to check your settings. You may have disabled a feature or configured something incorrectly. For example, setting a brightness level of 0 for the LEDs will effectively turn them off.
- Assure that the timer is not set to standby (center position of the toggle switch). When in standby, the timer and LED strips are turned off. Although even when in standby, you should still see the initial red, green and blue test, along with the OTA update indication.

If the timer and/or LED strips still do not work, then you likely need to check your wiring and GPIO pins. Assure you have not reversed wiring or something is offset in the pin or JST connections.

Timing, Timer and Sensor Issues

Here are some of the most common issues, and potential fixes, when it comes to the timer and sensor parts of the project:

Time immediately starts or stops

This is almost certainly an issue with the start/finish line sensors. Recall that these sensors are measuring distance and not motion. And they will detect **any solid object** within their field-of-view and within the distances specified in settings. If the time is starting or stopping unexpectedly, check the position of the sensors and the maximum sensor distance specified in settings. It is likely that the sensor is measuring a random object. This could even be the track surface itself, meaning you need to adjust the vertical angle of the sensor until the track is outside of the field-of-view. You can also modify the sensor distances in the settings to prevent detection of objects outside of the race area.

Time randomly starts or stops

If the timer appears to randomly start when in 'Ready' mode or randomly stops when the timer is active (and you are sure no object entered the detection zone), it is possible that noise in the sensor readings are resulting in occasional false triggers. Noise can be introduced by a number of factors, but can be exacerbated by long cable lengths, small gauge wire or other nearby electromagnetic sources.

If you experience intermittent and random triggers of either the start or end sensor (or both), you can try increasing the debounce count by 1 until the false triggers stop.

At lower values, the debounce has a negligible impact on timing accuracy, as each measurement cycle only takes a few milliseconds. But higher values could introduce a small delay to the start or end triggers. For example, if a "cycle" of the code takes 10 ms, then a debounce setting of 10 cycles means that an object must be in the detection zone for 100 ms, or a tenth of a second, before the sensor will start or stop the timer. For this reason, a maximum debounce value of 5 is enforced.

In most cases, a value of 2-3 will resolve false triggers due to noise without impacting the general timing accuracy. If you are still receiving false triggers even with a value of 5, then you need to re-evaluate the physical sensor and installation. Try a shorter cable if possible, using larger gauge wiring between the sensor and controller and/or assuring the sensor is as far away from other electrical components as possible.

See the section on using the Web Settings page for info on how to change these settings and optionally save them as default boot values.

Time does not start/stop when an object crosses the line

Again, check your sensor alignment and distances. But also assure the timer toggle switch has not been set to manual mode. In manual mode, the time will not start automatically based on the sensors, but only when the start/stop button is pressed on the timer.

Timer "times out" before desired

Adjust the maximum allotted race time in the settings, remembering that ***this value is entered in seconds, not minutes***! In addition, recall that the system will automatically time out when the timer display reaches the maximum value that can be displayed. This is also dependent on whether you are showing tenths of a second or only whole seconds on the timer.

When using "Tenths Timing", the maximum race time is 9 minutes, 59 seconds (599 seconds). If an active race exceeds this time, the system will automatically time out.

When not using "Tenths Timing", then the maximum race time is 99 minutes, 59 seconds (5,999 seconds).

If you need a race time that may last 10 minutes or longer, you must disable tenths timing in the settings.

Settings and Web Settings Page

I cannot access the web settings page

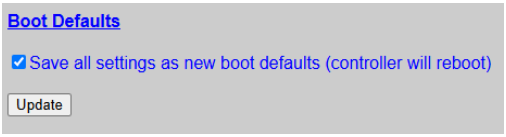
If you've assured that the controller is on your WiFi network (as per the the onboarding and onboarding troubleshooting sections of this guide), but you cannot get the web settings page to display, try or check the following:

- Assure the controller and the device you are using to access the settings page are on the same WiFi network/VLAN.
- Assure you are trying the correct IP address (use your router or other network app to locate the IP address).
- Put the system into Ready, Race Complete, Timeout or Standby mode. Remember that the ESP32 is somewhat limited in terms of processing power and trying to run an active race, control the timer and LEDs and also display a web page can be a bit taxing and lead to delayed load times for the web page.
- Did you flash a non-WiFi version of the firmware? While no longer supported, older versions of the firmware included a non-WiFi version that did not include features that rely on WiFi… like the web settings page. If using the non-WiFi version, the only option for changing settings is to modify the source code, compile and then upload to the ESP32.
- As always, try power cycling the controller.

If connected to WiFi, the settings page should be available. If you still cannot access the web settings page after trying the above, then it is likely a network-related issue and you need to look at your particular network settings.

My settings keep getting lost

If you make settings changes via the web settings page and they work fine until the controller is restarted, at which point the settings changes are lost and the board reverts back to the previous settings, then you may not be saving your changes as the new boot defaults.



Recall that all changes made on the settings page are ***temporary*** and will only remain in effect until the controller is restarted unless you check the box to save the current settings as the new boot defaults. Checking the box will save your current settings to the stored configuration file and they will be used moving forward when the controller restarts.

It is also possible that new settings or features in a firmware update requires a rewriting/overwriting of the configuration file. In this case, your settings may be lost or the controller may need to be onboarded again. But this will always be noted in the release notes for that version, and the release notes will contain additional instructions for the settings/configuration file.

Other Problems or Questions

If you experience an issue or problem not covered here, try reviewing comments in the Github issues and discussion areas and related media (e.g. YouTube, Blog) to see if someone else might have a similar problem (and if there is a solution or workaround). If you don't find your issue or problem, post a new discussion topic in the Github discussion area. The developer and other users monitor this platform and you may find a solution there.

Please reserve the Github issue area for true errors or issues directly related to the firmware. For general questions, hardware or other related matters, please use the Github discussion area.

Also remember to check the Github repository/Wiki and version release notes as later versions of the firmware may have introduced changes not reflected in this user guide.

Links and Additional Info

Here are a few additional links that may be of interest as related to this project.

Github Repo For this Project
Latest Release of the Firmware
YouTube Video of the Project (coming soon)
Written Blog Article, including parts and build info

ESP Desktop Flashing Utilities:
ESPHome Flasher (archived, but still functions at time of publication)
NodeMCU PyFlasher

Specs and Misc Info

While it is certainly possible to duplicate the functionality of this timing system with other controllers, languages, etc. those are not officially supported nor will requests for modifications be accepted. The source code is freely available via Github so you can fork or clone this version into your own. I am unable to do so on your behalf. The following is information as related to the build covered in this guide and related documentation.

Processor:	ESP32 Dev Board
Power Supply:	5V 10A (may be different for your build)
LEDs Used:	WS2812b 60 LEDs/m
Arduino IDE Version:	2.3.3
Timer Matrix:	MAX7219
Start/Finish Sensors:	VL53L0x
Original Firmware Version:	0.20 (October, 2024)

A full list of parts used can be found in the related blog article.

Document Revision History:

v0.21.0 - Original document for firmware v0.21
v0.22.0 - Updated with changes for firmware v0.22
v0.23.0 - Updated with changes for firmware v0.23