Getting on the road with pi-topHELIOS

What is pi-topHELIOS?

pi-topHELIOS is a custom-built telemetric data system, designed to allow Solar Car Challenge teams to collect important data from their solar car during a race, and relay to data back to their chase car, where it can be viewed and logged in real-time by their support team. The hardware and software of the system is all open source - software and schematics are freely available and teams are actively encouraged to heavily customize and tailor the software to their team's specific needs. If properly implemented, data that can be collected from the solar car using the pi-topHELIOS system includes, but isn't limited to, the following:

- Level of charge in the car's main battery
- Level of charge in any auxiliary batteries the car has
- The amount of power coming from the main solar panel array (voltage/current)
- The rate of charge / discharge of the main battery
- The car's GPS position, speed and time

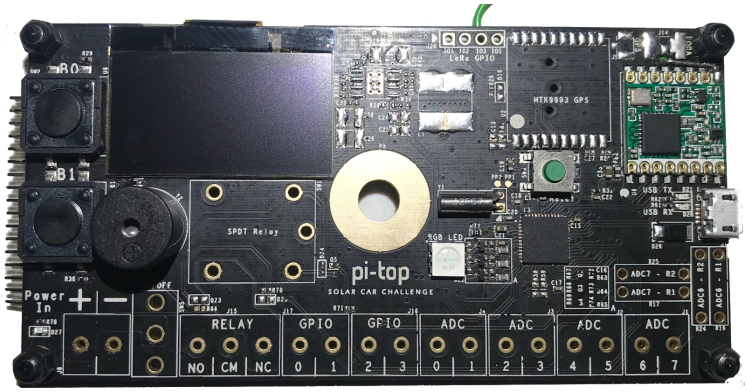
The system is comprised of two parts: The **solar car** board and the **chase car** board. As the name implies, the former is installed in the solar car, where it is connected up to monitor the batteries and solar panels etc. The board collects information on the status of the car, then wirelessly transmits this data back to the chase car board.

The chase car board is installed inside a pi-top, and when it receives data from the solar car, it in turn passes this data to the Raspberry Pi inside the pi-top. From here, members of the support team in the chase car can view and log the data on the pi-topOS system as needed. The system will continue to work so long as the chase car is within range of the solar car board - this has a theoretical design maximum of up to several kilometers, though at the time of writing the actual maximum range is untested.

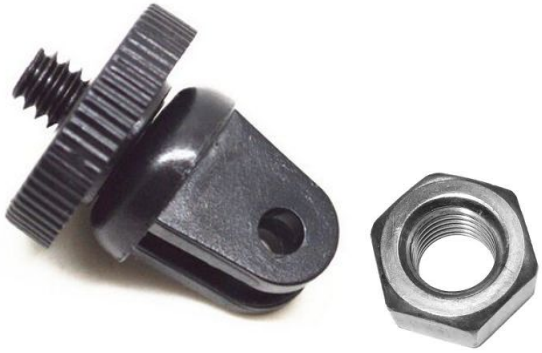
What's in the box

Here's what's included with the pi-topHELIOS kit each team receives.

Solar Car Board
May also be referred to as
"Board Variant 1"



Chase Car Board
May also be referred to as
"Board Variant 2"



¼" GoPro Mount & Nut
Allows the Solar Car Board to be mounted into the solar car using standard GoPro-type accessories

CR1220 Coin Cell
Used to keep time inside the GPS module of the Solar Car Board

Right-Angled Whip Antenna

For Solar Car Board LoRa

High-Gain GPS Antenna

For Solar Car Board

Board Tech Specs

Both board variants

- Powerful central microcontroller controlling whole system:
    - 32-Bit ARM Cortex M0+ (with 48 MHz Core Clock)
    - 256 KB Flash Memory, 32 KB SRAM
    - Pre-installed with Arduino Zero bootloader, so it can be programmed via micro-USB connector with no additional tools needed
- Fully configurable 1.3” monochrome display
    - 128x64 pixel resolution
    - OLED technology gives excellent readability even under midday sun
- Onboard LoRa (Long-Range) Radio Transceiver
    - Range of up to approx. 2Km, depending on obstructions, frequency, antenna etc.
    - Uses the license-free ISM band: "American ISM" @ 915MHz
    - Solar Car Variant: SMA connector for attaching the included whip antenna to improve range
    - Chase Car Variant: Pre-soldered ¼ wavelength wire antenna
- Various customisable ways of interacting with the user:
    - Two large momentary push buttons
    - One 5x5mm ultra-bright RGB LED, with 8-bit PWM depth (16M possible colours)
    - One obnoxiously loud buzzer - useful for emergencies
    - 4 General Purpose Input/Output (GPIO) available. May be used for monitoring or controlling any custom attached hardware. Max I/O voltage 3.3V, 7mA per pin.

Solar Car board variant only

- Onboard GPS module:
    - -165 dBm sensitivity, 10 Hz updates, 66 channels
    - Capable of tracking up to 22 satellites
    - Inbuilt Real-Time Clock (RTC) with battery backup for keeping time/date
    - SMA connector for attaching high-gain external GPS antenna (provided)
- 8-Channel 12-bit analog-to-digital converter (ADC):
    - 6 channels with built-in voltage divider, capable of safely reading up to 65V with accuracy to within 100mV.
    - 2 channels with unfitted through-hole 1/4W resistors for a user-configurable voltage divider, for the team to set to a value optimised for their system.
- Single-Pole-Double-Throw (SPDT) relay:
    - Software controlled
    - Allows for switching of a high-current (up to 5A) , high-voltage (100V) connection.
    - E.g. Could be used to route power from solar panels between different batteries
- Other features:
    - Power input supports wide range of voltages (approx. 8 - 18V), should work in all solar car configurations.
    - Small form factor and features to maximise east of installation, with ¼” mounting hole and included GoPro-style mounting accessories.

Getting ready to program a pi-topHELIOS board

Although the boards come pre-flashed with basic firmware to get you up and running, you should update to the latest firmware available from the Solar Car Challenge GitHub repository.

At their core, the HELIOS boards run the same hardware used in an Arduino Zero board. If you're unfamiliar with Arduino, now's a good time to check out the introduction section of the Arduino site. Note that the Arduino Zero uses a powerful ATSAMD21 chip running at 48 MHz, much more powerful than the standard ATMega328P used in the normal Arduino Uno board. Most libraries will work with this more powerful core, especially devices and sensors that use I2C or SPI.

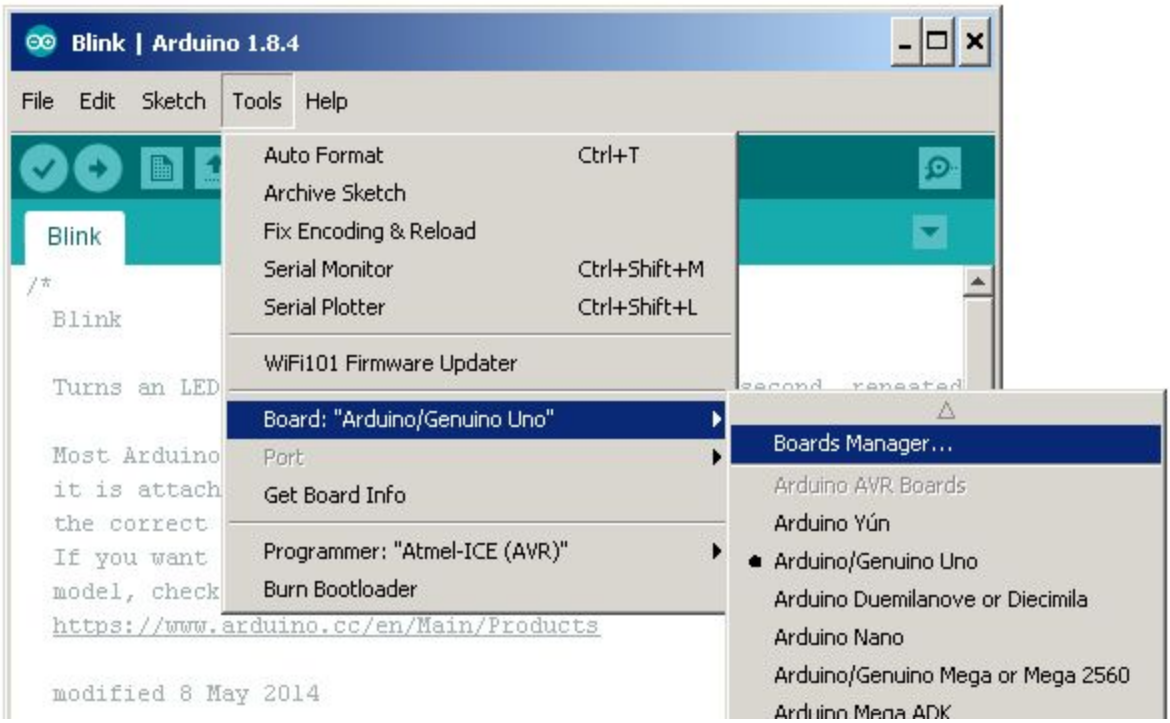
By using the same hardware as the Arduino, the HELIOS boards can be programmed easily and quickly from any computer, with nothing more than a USB cable. The section will guide you through installing and configuring the Arduino software to program a HELIOS board.

Installing the Arduino Integrated Development Environment (IDE)

The first thing you will need to do is to download and install the latest release of the Arduino IDE.

Installing Support for the pi-topHELIOS board

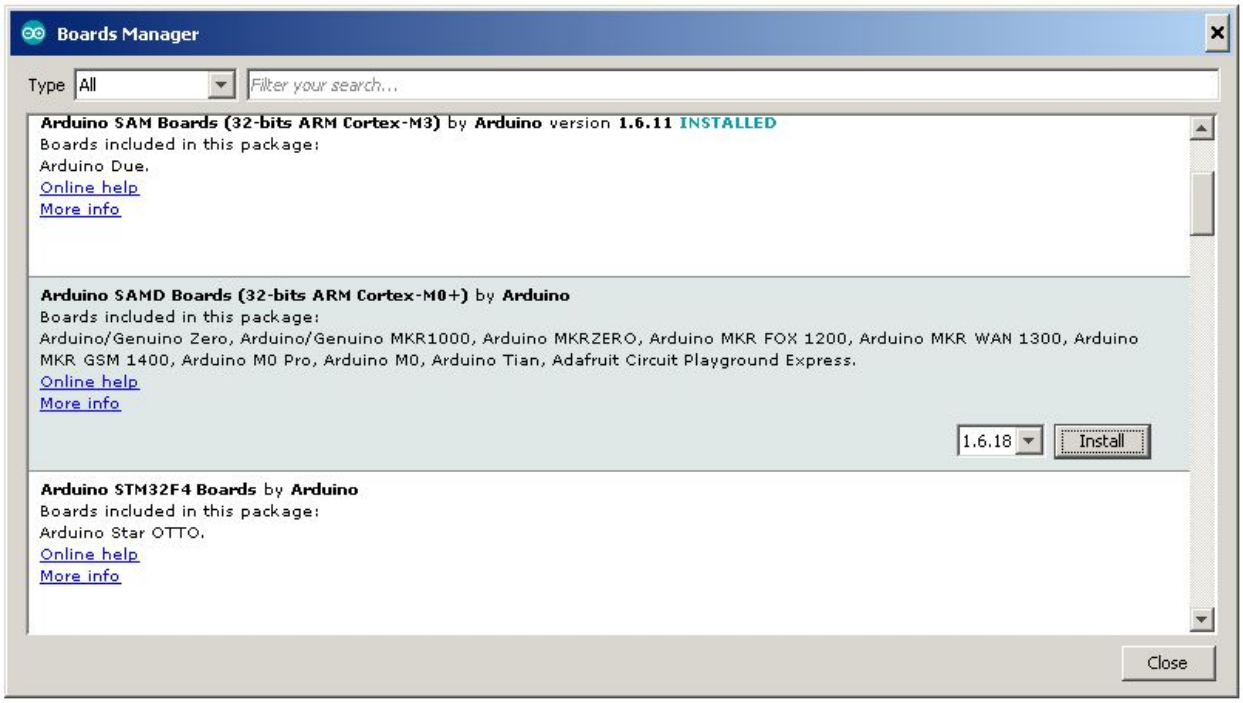
With the Arduino IDE up and running, you'll need to add in support for the Arduino Zero board using the boards manager - remember that as far as the Arduino software is concerned, the pi-topHELIOS board *is an Arduino Zero*.



Open the Boards Manager by navigating to the Tools -> Board menu.

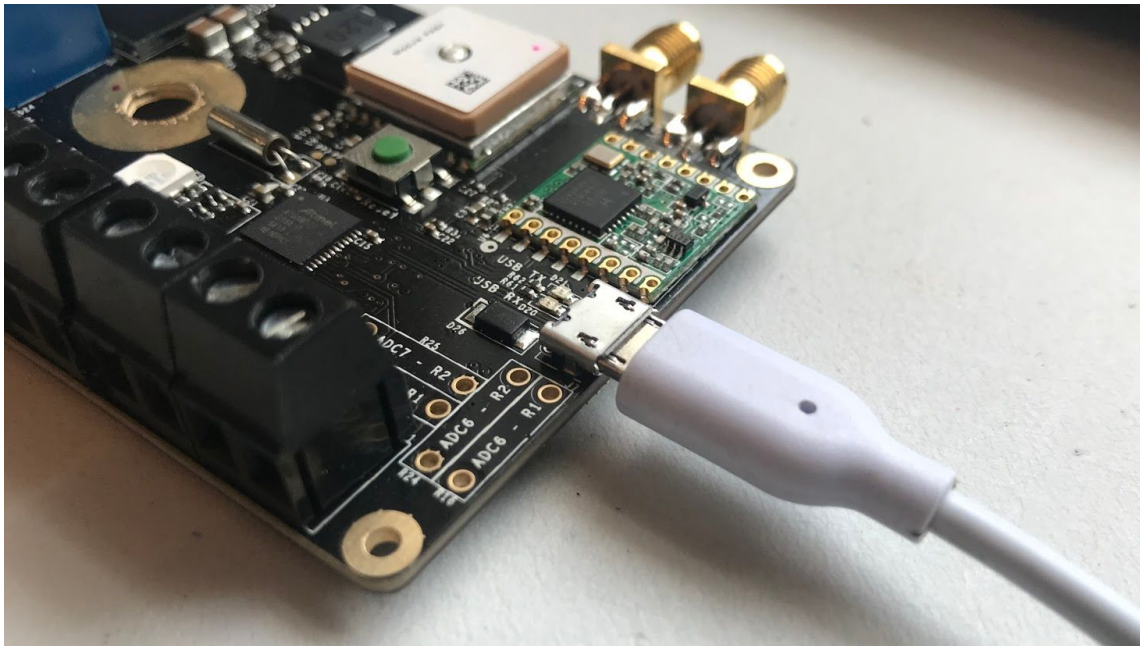
Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select All. You will then be able to select and install boards.

You'll need to install the latest **Arduino SAMD Boards** (version **1.6.18** or later) - to do this you can type Arduino SAMD in the top search bar, then when you see the entry, click Install. Be careful to make sure you install the **SAMD** boards, not the **SAM** boards.



Installing drivers for the board

At this point, connect a HELIOS board to your computer using the USB port on the right hand side. Don't worry about powering the board externally - for now, it can draw all the power it needs from the USB port.

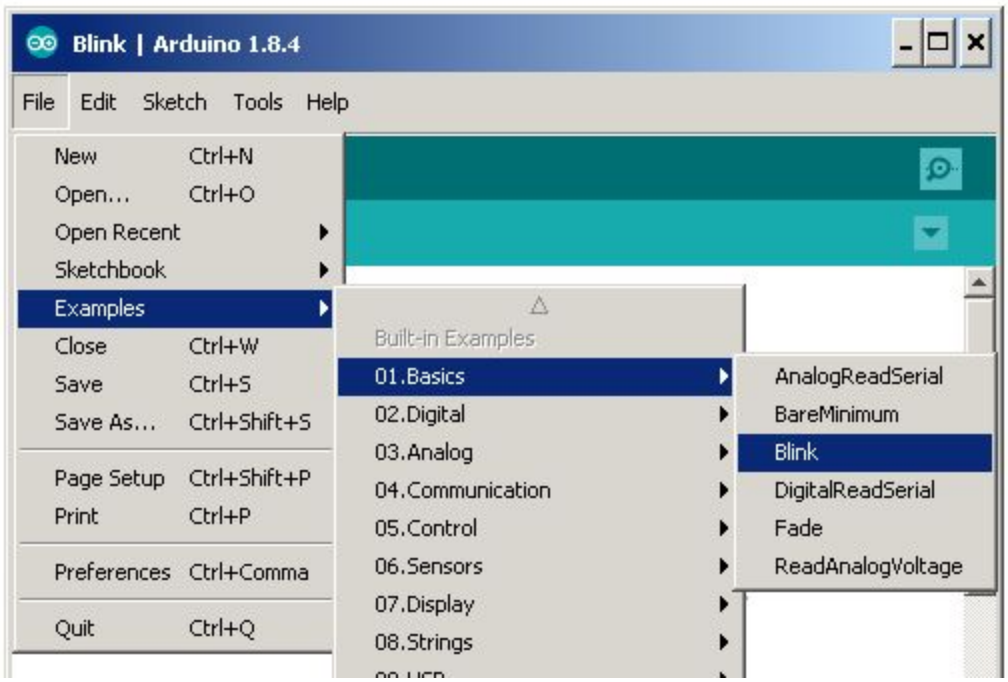


If you're on a windows machine, it should go ahead and automatically recognise the board as an Arduino Zero and install drivers by itself. OSX and Linux won't need to install any drivers.
If you run into any trouble with this step, check out the *Installing Drivers for the Zero* section of this great Getting Started Guide on the Arduino site.

Loading a basic test sketch

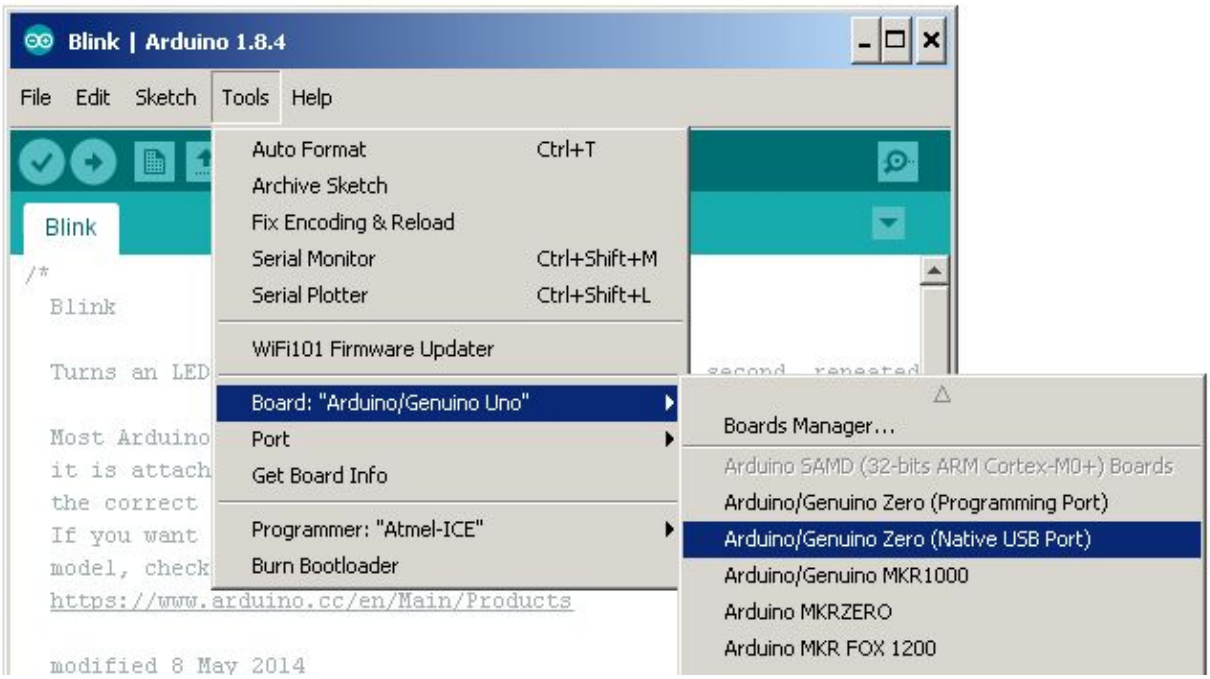
Now we're ready to install a very basic 'test sketch' onto the HELIOS board. Note that this will wipe the firmware currently on there - so this is the point of no return! Don't worry, the rest of this guide will explain how to install the latest HELIOS firmware back onto your board.

First, open the 'blink' example sketch, found in File -> Examples -> Basics.

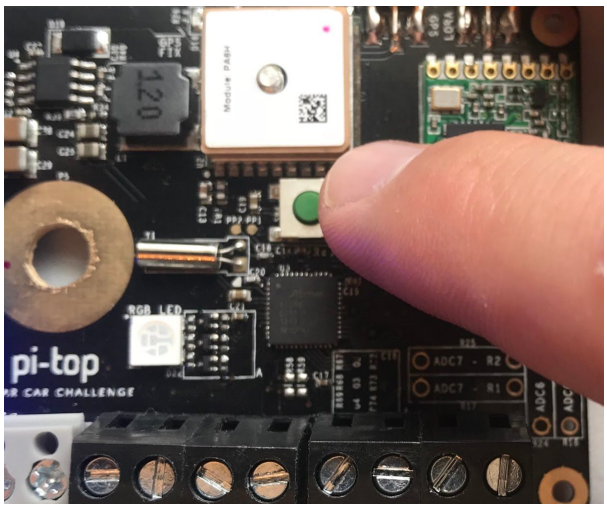


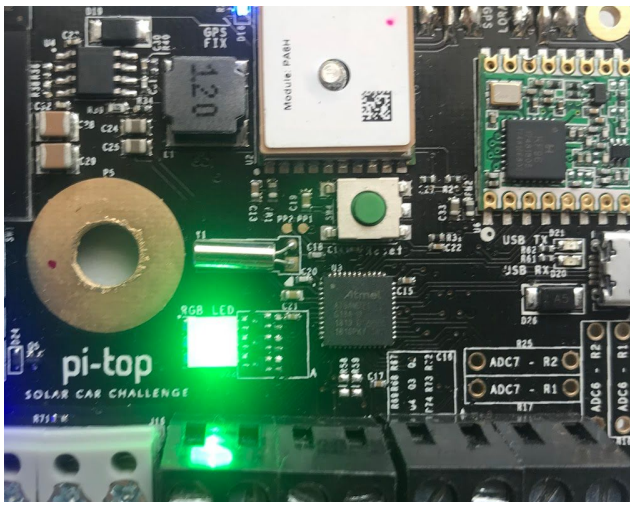
Next, we need to tell the Arduino IDE what type of board we intend to program - this is a very important step, so don't forget it! As already mentioned, as far as the Arduino IDE is concerned, this board is an Arduino Zero. Head to Tools -> Board and select **Arduino/Genuino Zero (Native USB Port).** It's super important to use the **Native USB Port** one and not the programming port one[1].

[1] The 'real' Arduino Zero has two seperate USB ports to provide two different ways of programming and debugging the Arduino. pi-topHELIOS does away with the second port to save complexity, so only the 'Native USB Port' still applies.



Finally, to actually program the board, we have to first put it into “bootloader mode” - essentially a standby mode where the board waits to receive new software from the computer. Press the MCU reset button twice to enter bootloader mode.



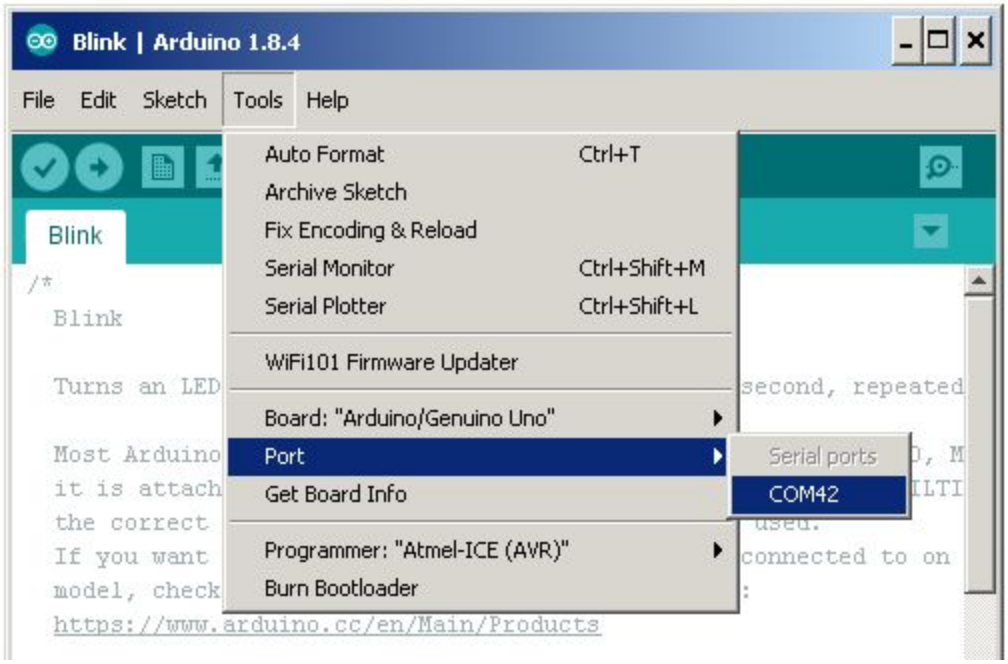


The main LED on the board will glow green to let you know it’s in bootloader mode - as shown in the picture. You may sometimes find that the MCU doesn’t enter bootloader mode as expected, for example if the timing of your button pushes wasn’t quite right - in this case, just press the reset button once to go back to normal mode, then after a moment try pressing twice again to get into bootloader mode.

Once in bootloader mode, give your computer a moment to recognise the ‘new’ device (bootloader mode presents itself to the computer as a different device to ‘normal’ mode). Windows may need to take a moment to automatically install some more drivers.

Once the board has been detected, you should see a new COM device in your port list (Tools -> Port), which you should select. To avoid confusion, you might want to unplug any other USB devices that are also creating

COM ports (e.g. bluetooth dongles). If you're still having trouble seeing the COM port, check back to the earlier mentioned Getting Started Guide on the Arduino site.



With all of that done, we're ready to program the board. Click the 'Upload' button (the right-pointing arrow) - the Arduino IDE will compile the code, and if everything is setup correctly, it'll automatically upload the sketch onto your HELIOS board and you should see the small orange activity LEDs next to the USB port flash a few times to indicate the code is being sent.

After a few moments, your new 'blink' sketch should run! In this case, if it's worked you'll see the main LED turn on and off slowly (it'll be green, so don't mistake it for the bootloader LED which doesn't turn off). It's not very interesting, but you've just programmed in your own code into the HELIOS board, hooray! Now we're ready for more interesting stuff...

Programming pi-topHELIOS firmware onto the board

Now that you've got down the basic toolchain for programming the HELIOS board, we can look at programming in the pi-topHELIOS firmware we've written to get all the basic features of the board running. This will work mostly the same way the programming in the 'blink' sketch worked, but with a few extra files.

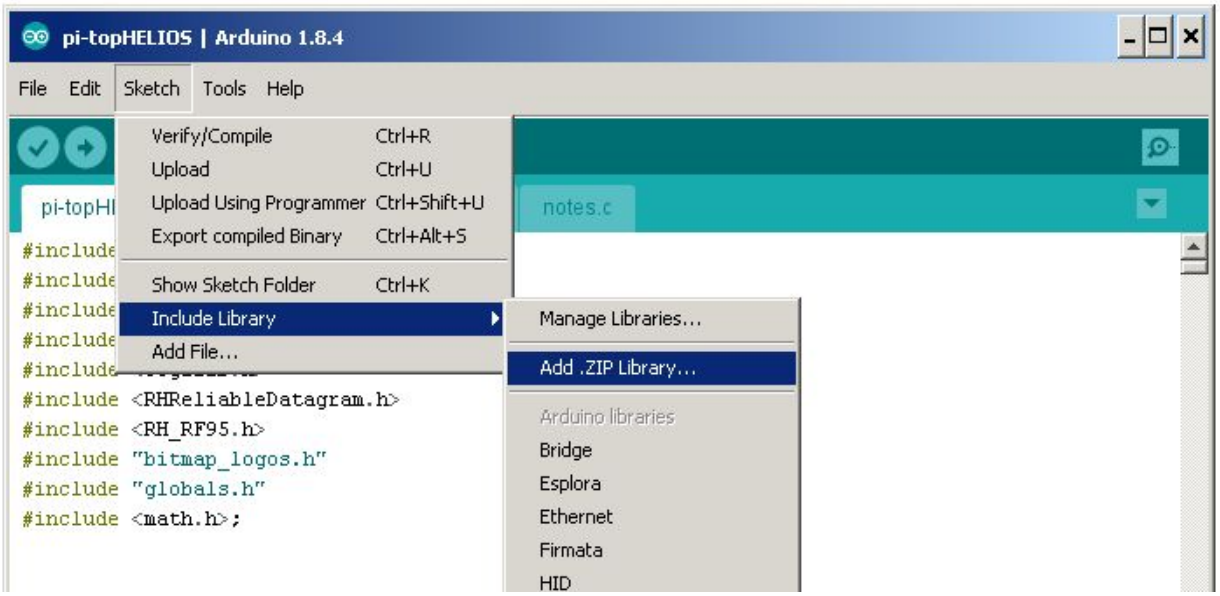
First, download the latest sketch files from the Solar Car Challenge GitHub. You'll find all the files you need in the *'Radio Program'* Folder, so download the whole folder and unzip it. This should consist of a main folder called 'pi-topHELIOS', and a folder called 'libraries'. The Arduino application code is all contained within the 'pi-topHELIOS' folder, but you'll need to copy across the libraries into the Arduino environment for the application to work. Once again, the Arduino site has a lot of useful information about libraries and what they're for. The following is an excerpt from that page:

Importing .zip Libraries

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside

the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library. Do not unzip the downloaded library, leave it as is.

In the Arduino IDE, navigate to Sketch -> Include Library -> Add .ZIP Library. At the top of the drop down list, select the option to "Add .ZIP Library".



Using this tool, add the following four .zip libraries in the libraries folder:

- **Adafruit_GPS_Library** - *For talking to the GPS module on the board*
- **Arduino_280859** - *For basic and advanced graphic functions for the OLED display*
- **MCP3208-master** - *For communicating with the ADC*
- **RadioHead** - *For controlling the LoRa module, and managing low-level communication layers for packet reliability*

If adding the .zip libraries fails for any reason, they can also be copied manually to the libraries folder instead (close the Arduino IDE first) - check out the ‘manual installation’ section of the Arduino libraries page for info on how to do this.

Opening the pi-topHELIOS code

Once you’ve installed all four of these libraries, you’re ready to open the main code. It’s good practice to copy your work into Arduino sketch folder (though this isn’t strictly necessary) - You can find (and change) the location of your sketchbook folder at File -> Preferences -> Sketchbook location. The default location on Windows is usually “C:\Users\<username>\Documents\Arduino”. Copy the folder called ‘pi-topHELIOS’ into this location.

Now, inside the pi-topHELIOS folder, open the Arduino file called **pi-topHELIOS.ino**. This will open up the main sketch for pi-topHELIOS firmware. It’s much like the ‘blink’ test sketch we opened earlier, but with a lot more HELIOS-specific code in it.

We’re almost ready to program this into the board, there’s just one more **very important detail** to take care of before you do! You’ll have to make a small modification to the code depending on what **type of board** you’re programming, and **what pairing ID** has been assigned to your board.

Type of board

The software has been written so that both the **Solar Car Board** and the **Chase Car Board** can be programmed from the same piece of software, but one line to needs to be changed in the code to match the type of board you're programming.
Towards the top of the **pi-topHELIOS.ino** file, look for the line '`#define SOLAR_CAR`'. In C (the language this is written in), this is called a **preprocessor**, and is a neat way of controlling / changing large sections of code from a single line.

```
//******* DEVICE ADDRESS & BOARD TYPE *******
//It's essential that the device's unique ID is assigned here
//This number is used to define a unique 100ms window in which to broadcast
//So, it is limited to a maximum value of 30
#define PAIRING_ID  1
//If this is the chase car board, comment out the line below
//If it's a solar car board, leave it uncommented
#define SOLAR_CAR
```

If you're programming the **Solar Car Board**, you can leave this line untouched, and skip straight on to the *Pairing ID* section.
If you're programming the **Chase Car Board**, you'll need to **comment out this line**. In C, commenting is done by putting a double forward slash ('`//`') at the start of the line. So, to program the Chase Car Board, your code should look like this:

```
//******* DEVICE ADDRESS & BOARD TYPE *******
//It's essential that the device's unique ID is assigned here
//This number is used to define a unique 100ms window in which to broadcast
//So, it is limited to a maximum value of 30
#define PAIRING_ID  1
//If this is the chase car board, comment out the line below
//If it's a solar car board, leave it uncommented
//#define SOLAR_CAR
```

Don't forget to change it back to being uncommented if you need to program the Solar Car Board again!

Pairing ID

There are a lot of cars at the *Solar Car Challenge***,** so there will be a lot of pi-topHELIOS systems all in one place! This can cause a lot of trouble for the wireless LoRa system, as a transmitted message from a Solar Car may be received by multiple Chase Cars. How does the Chase Car know a message is from its own Solar Car and not someone else's? The pairing ID!

Your team's pairing ID is written on a numbered label on each board - it should be a value between 1 and 30, and **it should be the same on both boards**. You'll need to enter this number to the '`#define PAIRING_ID`' line of the code. So if your pairing ID is 15, your code should look like this:

```
//******* DEVICE ADDRESS & BOARD TYPE *******
//It's essential that the device's unique ID is assigned here
//This number is used to define a unique 100ms window in which to broadcast
//So, it is limited to a maximum value of 30
#define PAIRING_ID  15
//If this is the chase car board, comment out the line below
//If it's a solar car board, leave it uncommented
#define SOLAR_CAR
```

Ready to program!

With the **board type** and **pairing ID** correctly set, you're good to go. You can now load this code straight to onto your HELIOS board. The procedure is exactly the same as with the 'blink' test code we covered earlier. Here's a quick recap of how to upload the code.

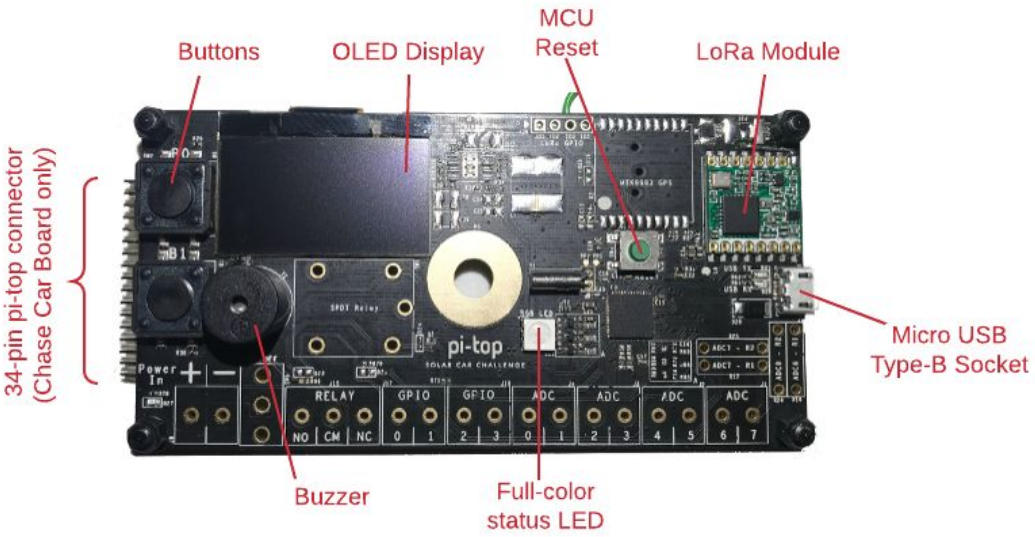
1. Plug in USB cable.
2. Board should turn on and boot as normal (regardless of power switch position).
3. Press 'MCU Reset' button twice, status LED should slowly pulse green to indicate that the board is in 'bootloader' mode. Note that bootloader mode can be exited by pressing the reset button again.
4. New device should appear in windows
5. In the Arduino IDE, select **Arduino/Genuino Zero (Native USB Port)** as the board in Tools -> Board.
6. Also select a whatever port corresponds to the HELIOS board in Tools -> Port.
7. *If using the supplied pi-topHELIOS code:* Make sure you've updated the **board type** and **pairing ID** parameters (see next section).
8. Click upload (the right-pointing arrow).
9. Confirm it's uploading with orange USB activity LEDs, and then away it goes!

Using pi-topHELIOS

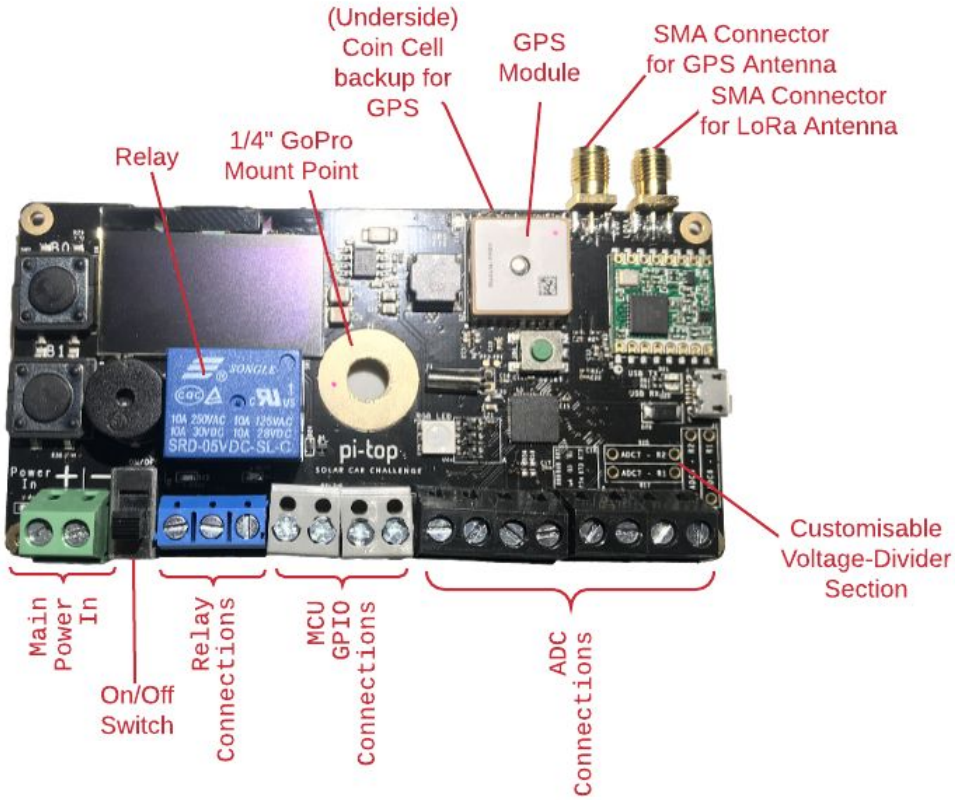
Hardware Overview

So let’s take a closer look at the hardware you’ve got in front of you:

Solar Car Board



Chase Car Board



As you can see, a lot of different toys to play with. There are various connections that can be used in any way your team sees fit. In this section we'll break down how you can get the most out of using each feature of the board.

You'll already have noticed that the Solar Car Board is significantly more advanced than its sister board. The simpler Chase Car Board is the same basic circuit board, but with the ADC, GPS, relay and screw-terminals stripped out since they're not needed in the Chase Car. It's designed to sit inside the pi-top and acts as a simple radio receiver for all data the Solar Car Boards collects, sending the data to the Raspberry Pi for processing.

A note on safety

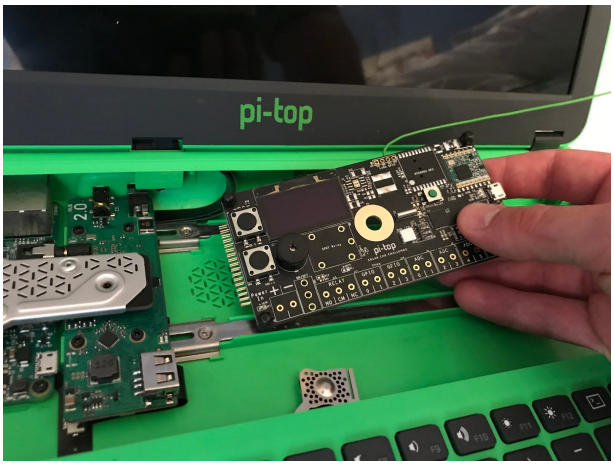
A few parts of the Solar Car board have been designed to handle the relatively large voltages and currents present on the solar car; the relay unit is designed to handle a maximum of 100 volts at 5 amps, and the ADC inputs can all safely handle up to 65 volts without being damaged.
However, these are **electrical design ratings**, and as such they are only indicators of what the board can handle **without being damaged.** They **are absolutely not** a guarantee that you can safely handle the board with bare hands when these voltages are present.

There is exposed, **live** metal in numerous places on both the top and underside of the board - most prominently in the exposed solder pins of the screw terminals and relay module, but also in various surface-mount components. If you have high-voltage (more than about 25 volts) wiring connected to the pi-topHELIOS board e.g. from a solar panel or battery array, **constant care should be taken** to ensure nobody can accidentally touch any of the live metal parts of the HELIOS board. We **strongly** recommend you keep it in a safely isolated plastic enclosure at all times.

Installing the boards

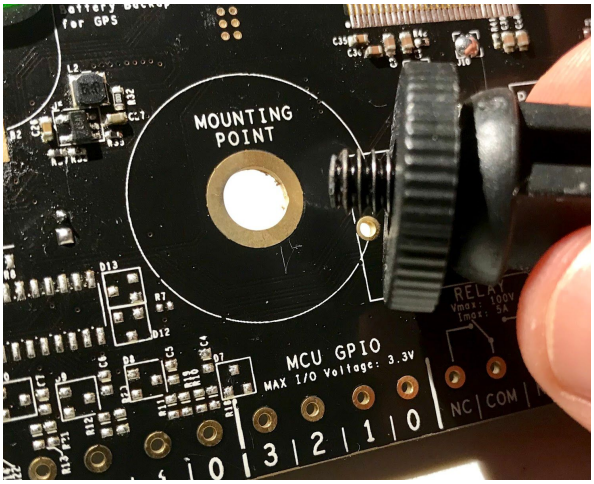
Chase Car Board

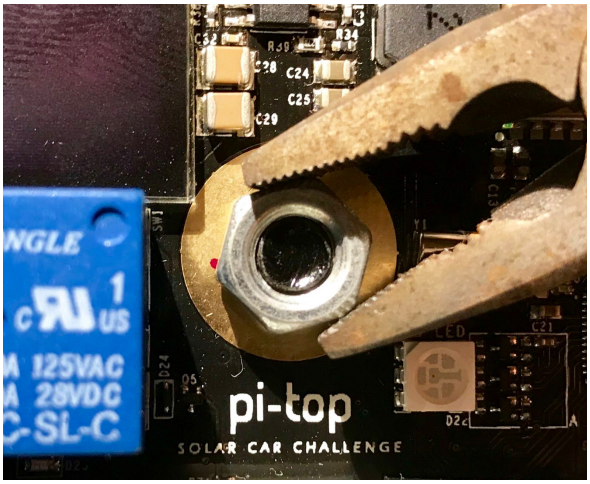
The Chase Car Board has magnetic feet and is designed to go into the pi-top - to install it, first turn off the laptop, then place the board onto the magnetic rail and slide it across until it's connected with the pi-top hub. You might want to wiggle the wire antenna around a bit to make sure it sits neatly flush at the back of the laptop, as illustrated below (if you can make out the green antenna on the green background!).



Solar Car Board

The Solar Car Board has been designed to give as many mounting options as possible, so it can be installed in the most convenient location in your solar car. A mounting hole and some GoPro-compatible accessories are provided to make the board mountable to just about any surface - use it with your own GoPro stuff too!





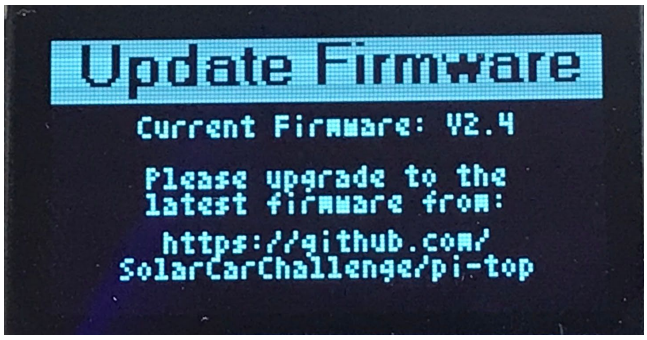
Powering the Solar Car Board

The Solar Car Board has been fitted with the same robust, wide-input range regulator circuitry we use in the pi-top. It can handle an input voltage anywhere in the range of 8 - 18V, and will internally regulate that down to a 5V supply and 3.3V supply. This means you can safely power it straight from a typical lead-acid battery, which will put out somewhere between 10V - 14V through its charge / discharge cycle. Be careful not to allow a supply voltage or more than about 18V, or you might fry the regulator circuitry! As for power consumption, the board doesn't draw much current compared to the various motors and peripherals of the solar cars, but don't forget to turn it off overnight when you're not using it.

The Chase Car Board doesn't have regulator circuitry on board, and is designed to draw power directly from the pi-top, so you don't need to worry about plugging in a power supply.

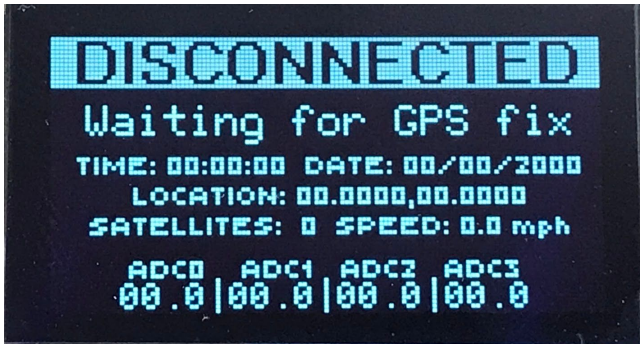
Navigating the User Interface

Starting up your HELIOS board for the first time, you'll probably be greeted with a message that looks something like this:



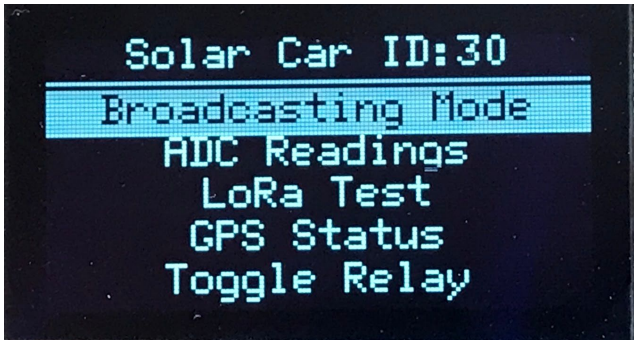
When the boards left the factory, we were still putting the final touches on the firmware, so there might be more up-to-date and reliable firmware available on the Solar Car Challenge GitHub pages. We strongly recommend you update to the latest available firmware when you receive your board, to make sure you get the most from it! You can dismiss the update message by clicking any button, or just wait 30 seconds and it'll disappear automatically.

At the time of writing, the default mode of the HELIOS firmware for the Solar Car Board is to go straight into attempting to get a GPS fix and connect to its paired Chase Car board. That looks something like this:



Once the board has acquired a GPS position and has connected to the Chase Car Board, it will automatically start transmitting GPS and ADC data at an interval of 5 seconds.

To make life easier, the firmware has been packed with some extra debugging features, to allow you to check if individual things like the ADC, GPS, Radio or Relay are working. Whilst on the 'broadcasting page', click any button to drop into the main menu.



You'll see your programmed **pairing ID** in the status bar at the top, and a list of testing modes you can try out. Use the lower button to scroll, and the upper button to select. Exit any of these modes by pressing or in some cases holding-then-releasing any button.

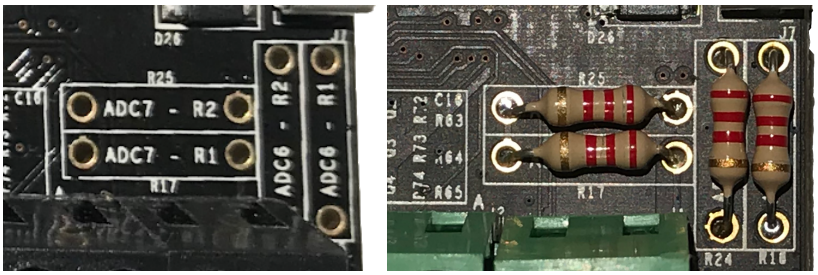
The user interface of the Chase Car Board is much simpler. There is no navigable menu, and the only page you see (after dismissing the firmware update and passing the 'splash screen' images) is a basic status page that indicates if there is currently a connection to the Solar Car Board. Of course, you're free to dive into the code and add whatever features you want, to either board.

Analog-to-Digital Converter (ADC) Connections

The Solar Car Board features an 8-channel ADC, with an internally connected voltage divider on channels 0 - 5, allowing them to safely read voltages up to 65V without the need for any additional hardware. They should be perfect for reading battery voltages, solar panel voltages and even differential voltages across a shunt resistor used to measure current. Although they update relatively slowly in the stock firmware the boards are shipped with, they're capable of updating at rates >100Hz, and have a maximum resolution[2] of 16mV. The ADC testing/debugging page, accessible from the main menu, shows the capability of the ADCs running at full speed.

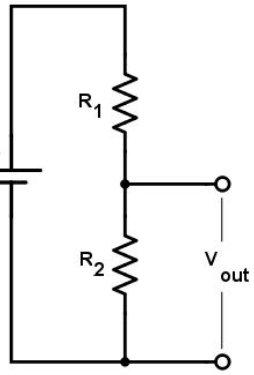
Configurable Voltage Divider

Two channels, 6 and 7, have been left with an unfitted voltage divider, in the bottom right corner of the board, to allow teams to customise the divider ratio and hence maximum voltage & resolution of their boards (e.g. using a smaller voltage divider will limit maximum readable voltage, but allow for greater resolution). The footprints will fit any standard through-hole ½ watt or ¼ watt resistor.



Calculating values for a potential divider is very straightforward - just follow the equation below, where V_{in} is the input voltage at the screw-terminal and V_{out} is the voltage going into the ADC. It's very important to keep in mind that the **maximum voltage the ADC can take is 3V.** Don't allow your V_{out} voltage to be higher than that, or your ADC will meet a rapid, traumatic end.

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

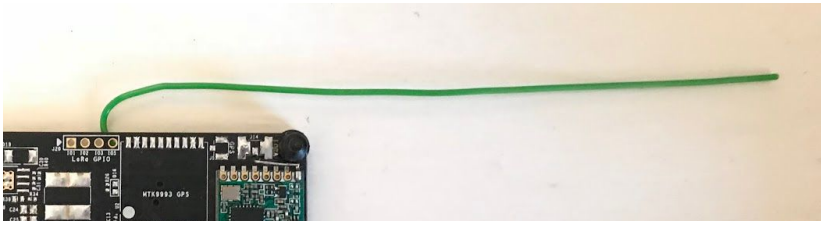


[2] This is the maximum resolution readable at the ADC - but keep in mind that this is *after* the signal has passed through a large step-down voltage divider, so the maximum usable resolution at the screw-terminals is significantly more coarse.

This excellent tool from *Electronics2000* will let you play with different values to find something that suits your application. We recommend using an R2 value of at least 10K ohms, to ensure the current flowing through your voltage divider is kept to a minimum. The footprints on the Solar Car Board marked ‘R1’ and ‘R2’ respectively correspond to the diagram above.

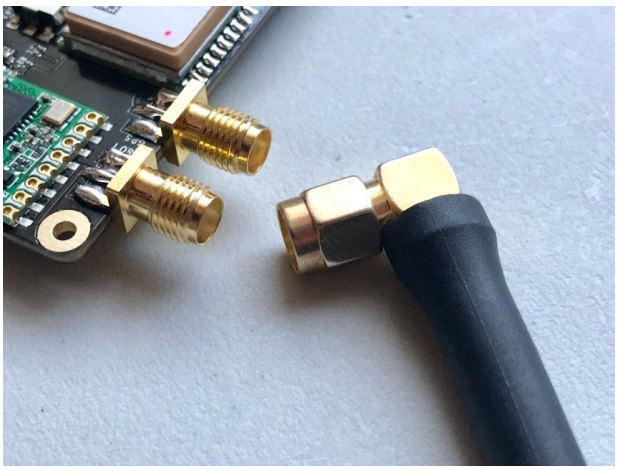
LoRa Radio

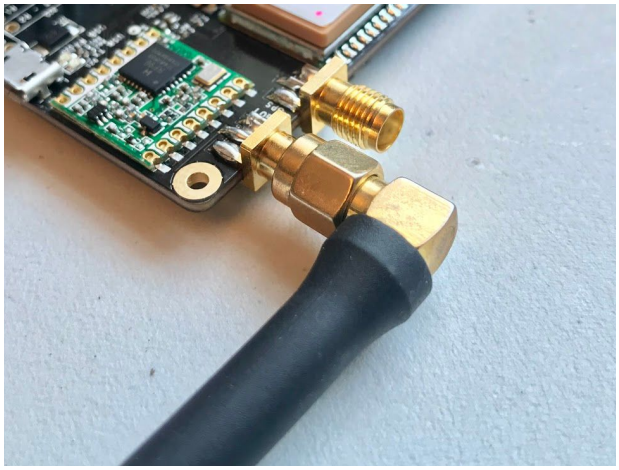
To get good range out of the LoRa radio, you’ll need to make sure both boards have an **antenna** attached. The Chase Car Board already has a simple **wire antenna** soldered on (see picture).



Wire antennas are cheap, easy and give pretty good signal strength[3], but are a little unsightly to look at, and can get caught or break off easily. For this reason it’s okay to have one safely tucked inside the pi-top, but probably not on the Solar Car Board.

For the Solar Car Board, we’ve supplied a rugged external **whip antenna** designed specifically for the 915 MHz frequency these boards use. Screw it onto the **SMA-type** connector on the corner of the board. **Be careful not to mix it up with the connector for the GPS antenna!**



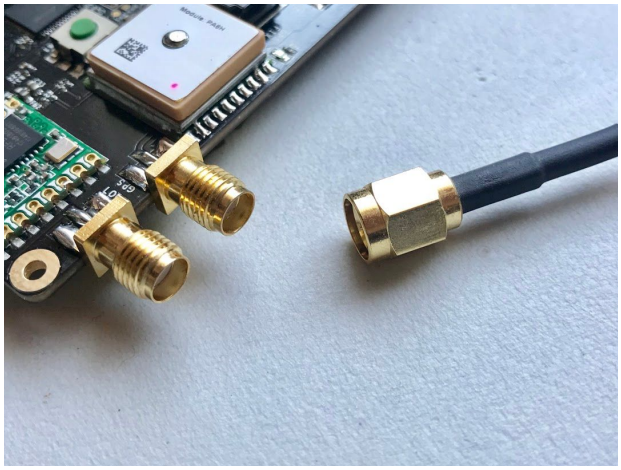


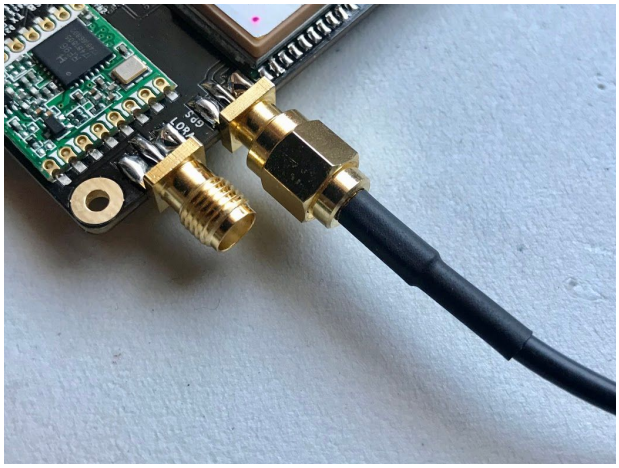
[3] To give the best possible reception, a wire antenna needs to be tuned to a length corresponding to the wavelength of the main frequency being used. In this case, the main frequency is 915 MHz (which corresponds to a wavelength of 330mm), so this is a ½ wavelength antenna measured to be exactly 165mm long!

GPS Connectivity

The Solar Car Board has a great little GPS module on-board, which gives accurate and reliable data, but only when it's getting good signal from its antenna. The type of antenna often used for a GPS module is called a **ceramic patch antenna** - it's the distinctive white/beige cuboid in the top right of the board, and looks very different to a typical radio antenna you might use for your LoRa radio or WiFi router.

Ceramic patch antennas are more **directional** than wire antennas, so their orientation is critical to get a strong GPS signal. The patch antenna needs to have its **top surface facing flat towards the sky** (or more accurately, towards the GPS satellite constellation). Whilst it's possible the design of your solar car allows you to install the Solar Car Board in a position where you can achieve this relatively fussy design requirement, we figured it'd be much easier if you had a powerful external GPS antenna available to connect to the board and run to somewhere convenient.

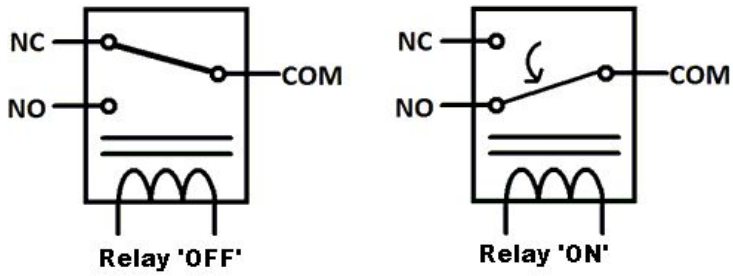




Attach the external GPS antenna to the GPS **SMA connector**, and once again **make sure not to mix it up with the LoRa connector!** Once you've done that, you can run the antenna to anywhere you like, but keep in mind that inside the external antenna is just *another* ceramic patch antenna (albeit a more sensitive one), so it has the same requirements - keep it somewhere external on the vehicle if possible, and make sure it's flat, so the top surface roughly faces the sky.

Relay

The set of blue screw terminal connections on the Solar Car Board are connected directly to the large blue relay next to them. The relay allows for software-controlled switching of a signal connected to the screw terminals, based on the simple arrangement illustrated below.



On the underside of the board, the screw terminals are marked NC, COM and NO for ‘normally closed’, ‘common’ and ‘normally open’ respectively. Two blue LEDs indicate which of NC/NO is currently connected to COM. The maximum rated voltage for signals connected to these terminals is 100V, and the maximum current is 5A.

MCU GPIO Connections

There are four GPIO connections available on the **grey** screw terminal blocks, which connect directly to the MCU. These GPIO connections are the same connections you’d find available on a normal Arduino board - but note that the pin numbering on the bottom of the board (0-3) doesn’t correspond to actual Arduino pin numbers. To use these GPIO pins, simple call them using their C precompiler definitions, found in the ‘globals.h’ header file. These are `GPIO0_PIN`, `GPIO1_PIN`, `GPIO2_PIN` and `GPIO3_PIN` respectively.

Note that the MCU of the pi-topHELIOS (and the Arduino Zero) board runs on a 3.3V core voltage, and its GPIO’s are all 3.3V logic as a result, **different to the standard 5V on an Arduino Uno.** Be careful not to plug in anything with 5V logic as you might fry the microcontroller on the board! Also be sure to observe the 7mA current limit of these pins - they’re not designed to kick out lots of current, just a enough to talk to other chips or drive a small LED!

Coming Soon

~~In car adaptor~~

~~The in-car adaptor has been built especially for this application, and is only a quick-fix for the otherwise inconvenient problem of the pi-top running out of power during the race! As such, it isn't a long-term solution, and it isn't perfect.~~

~~Since the pi-top is designed to normally be charged from an 18V power supply, the 12V (closer to 14V with engine running) from a car socket isn't enough to properly charge the battery - but it can keep the laptop running, with a few caveats.~~

~~When plugged in to the car, the pi-top will think it's on mains power, and can be turned on and used as normal. The pi-top battery will even charge (though slowly and possibly not to 100%) - but it's important to note that if you unplug it from the car, the pi-top battery shouldn't be allowed to drop beneath 20% if you want to charge it from the car again! At this level, the batteries are too discharged to be powered by the car socket, and will need to be plugged into power from the regular wall-wart power supply to get them going again.~~

~~You'll also find that charging the pi-top from the car will be much more effective with the engine running, as the car's alternator will kick out closer to 14V to keep the car batteries charged, and this higher voltage suits the pi-top's charging circuits much better.~~

GoPro Mounting Accessories

Various basic accessories for mounting the Solar Car Board to the solar car.