

Purpose of this guide

This guide will describe how to assemble the hardware and software needed to wirelessly control the power delivered by a [Tesla Wall Connector](#) (aka TWC). This process does not work with the older High Power Wall Connector (HPWC) units that were discontinued in April 2016.

TWCs have the capability to wire up to four together to share a single power source. To negotiate power sharing, TWCs talk to each other over two wires known as an **RS-485 interface**. We can use a [Raspberry Pi Zero W](#) plus a **USB to RS-485 adapter** and a script called **TWCManager.py** that pretends the Pi is a TWC set to Master mode. Our Pi can then control the power used by up to three real TWCs set to Slave mode.

TWCManager includes a web interface to let you check, schedule, and adjust charging power:

Power available for all TWCs: 11.55A



TWC 9399: Charging at 10.20A.

Scheduled power: 40A ▼

from 11:00pm ▼ **to** 5:00am ▼

on days ☒ Su ☒ Mo ☒ Tu ☒ We ☒ Th ☐ Fr ☐ Sa

Non-scheduled power: Track green energy ▼

1-day charge, 40A

Save

The **Track green energy** option requires some extra programming on your part to interface with a source of green energy data. For example, if you have solar panels on your roof, ask your installer for access to a “web API” that can tell TWCManager how much power the panels are currently producing, then modify TWCManager to use that web API.

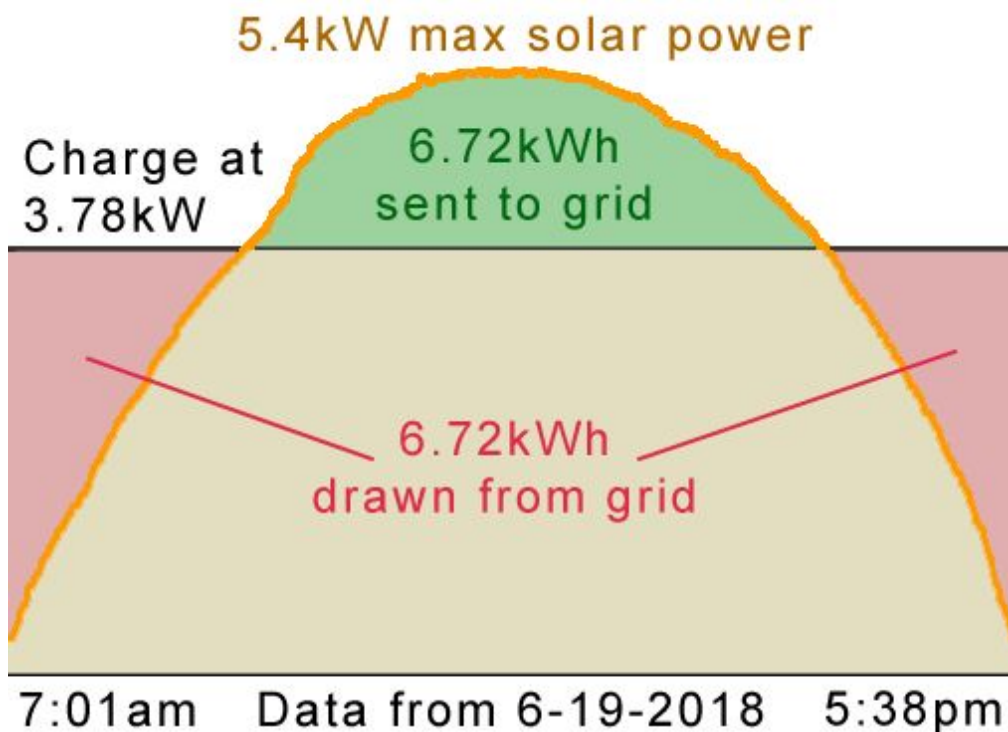
I wrote TWCManager with the intention of saving energy using track green energy. Those who don’t have net metering on their solar system may also want to use it to limit power drawn from the grid, or to charge only when Time of Use

rates are lowest. You can save even more resources by keeping this guide on a phone, tablet, or laptop instead of printing it.

Saving energy

TWCManager can save about **6kWh of energy per month** using the *Track green energy* option. Here's why:

- The Raspberry Pi plus RS-485 adapter uses around 0.911Wh which is **0.655kWh/mo.**
- Looking at an actual day of solar generation from our system:



- If we chose to charge at 3.78kW (15.75A at 240V) just while the panels are producing significant power, we would draw 6.72kWh from the grid and send back 6.72kWh to the grid, using net zero from the grid and providing 40.23kWh of charge to the car. That's more than enough for most people per day.
- The problem is that in 2016, about 4.6% of the energy distributed on the California grid was lost during transmission to customers. Many states are worse. You can [download an excel document for your state](#), open tab 10, then divide "Estimated losses" by "Total supply" to get a loss percentage.
- Drawing 6.72kWh from the grid per day * 22 work days = 147.84kWh per month * 0.046 = **6.8kWh of energy lost to transmission from the grid.**

That's more than 10 times more energy than needed to run the Raspberry Pi.

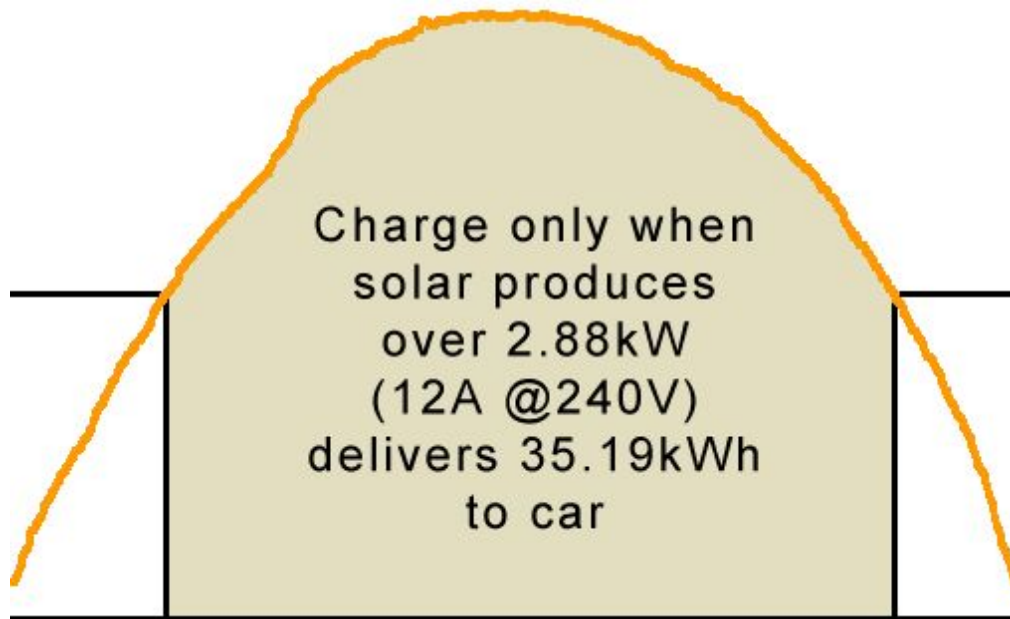
- I'm ignoring the energy lost by transmitting it back to the grid because hopefully it will get used by neighbors with minimal transmission losses, but that won't be true in all areas.
- We can avoid both of these losses by always charging at the same kW as the solar system is producing, and TWCManger lets you do that.

If you don't have solar or can't figure out how to track it, it's still best from a renewable perspective to charge the car during the day starting around 10am, at least in California. Historically, power has been cheapest at night because there's less demand. Yet CA has installed so much solar, there is actually too much power available on sunny days and solar generation is reduced (this is known as 'curtailing'). Looking at [data in 2017](#), the only time renewable sources were curtailed was during the day. At night, more power had to be imported from other states than during the day. This may be rather different in other states/countries, but many places should be heading towards this situation.

Problems with low charge rates

Charging at 5A wastes [8.16% more energy](#) than charging at 10A, which wastes 2.48% more energy than charging at 40A. Charging only at 20A and above would be a decent compromise (wasting less than 1.9%), but few home solar systems generate that high a power level for long enough.

TWCManger defaults to charging at a minimum of 12A (2.88kW) using the **minAmpsPerTWC** near the top of TWCManger.py. This limits losses to somewhere between 2 and 2.5%, in which case you would get 35.19kWh into the car, which is more than enough for most people on most days:



Low charge rates also lead to battery degradation. [Professor Jeff Dahn](#) is considered a leading authority on lithium ion batteries. He [says](#) minimal battery damage happens when charging at total battery capacity / 2. For a Tesla with 60kWh battery, $60\text{kWh} / 2 = 30\text{kW}$. Even $240\text{V} \times 80\text{A} = 19.2\text{kW}$, so there is no way to charge that quickly at home, but basically **charging at the maximum amps your charger supports is best to limit battery degradation**. Of course, we can't track solar output and charge at the max amps, but hopefully 12A minimum is a good compromise.

Jeff Dahn also says **leaving a battery at 75% SoC is best**. Since the car is only [using 95% of the true battery capacity](#) when the UI says 100%, **75% SoC should be more like 80% on the car UI**.

Safety and legalities

For my own installation, I've rigged a method to power a Raspberry Pi Zero W from the low-voltage power supply built in to the TWC. There is some possibility this could overheat and damage the TWC's low-voltage power supply. I've run mine with outside temperature varying between 95-100F, in shade, charging my car at 40A for over 4 hours. The hottest component in the TWC is a yellow and black transformer marked TA1 which reached 194.6F. I can't find a spec sheet for the transformer, but it looks like the minimum standard transformer rating is 284F max temperature. A line of four black resistors above TA1, each marked 3163, reaches about the same temperature. I don't know who makes the resistors, but similar ones I found online will operate up to 311F. Above TA1 is a FQPF5N90 MOSFET marked QA1 which reached 189.3F. Its spec sheet says it can reach 257F before it might have trouble supplying enough power. My tests

show the pi adds ~15F to the heat of these three components, so adding it is not pushing them enormously closer to their maximum temperatures.

All of this should mean nothing will fail even at 110F air temperature in the sun (which may drive the components up to 210-220F), but I can't guarantee it. I also don't know how charging above 40A affects the temperature in the TWC. I do know 80A chargers easily overheat the charger cable in warm air temperatures and will reduce charging rate until the cable cools.

Assuming the TWC won't be damaged by the extra power draw from the Pi, adding electronics inside the TWC still breaks safety codes in most areas and if it does somehow start a fire that destroys your house, your insurance company will likely see it as an unauthorized modification and try not to cover rebuilding your home. The TWC's case should prevent any sort of fire from escaping, but this is still something to consider.

For the highest level of safety and legality, please hire a licensed electrician to install an indoor or outdoor stand-alone enclosure to contain the Raspberry Pi W and RS-485 adapter. Power the Pi using a standard wall-power adapter instead of the TWC's power. Have your electrician run two small wires from the Pi's RS-485 adapter to the RS-485 ports in the TWC.

If you want to take the more risky approach of running the Pi from the TWC's power source, it's important to design the Pi to use as little power as possible. The hardware and software described in this document will draw around 0.911 Watts of power, adding about 5.7% to the peak power load on the TWC's power supply. As far as I can tell, the worst possible thing that might happen is the wiring in the hottest part of the TWC's power supply will melt insulation, short circuit, and trip a safety mechanism that cuts power. In that case, your TWC would need repair or replacement. There is likely some form of overheating protection that will prevent damage to the TWC, but I was not able to understand that aspect of it. Keep in mind, I'm not a trained electrician and I could be completely wrong. I've posted my investigations of the TWC's power supply [here](#) and gotten nobody warning me that adding the Pi is dangerous. If you feel that it might be, please let me know why. Either way, I can offer no guarantees as to the safety or long-term reliability of running the Pi inside the TWC.

In short, USE THIS GUIDE AT YOUR OWN RISK.

THIS GUIDE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT

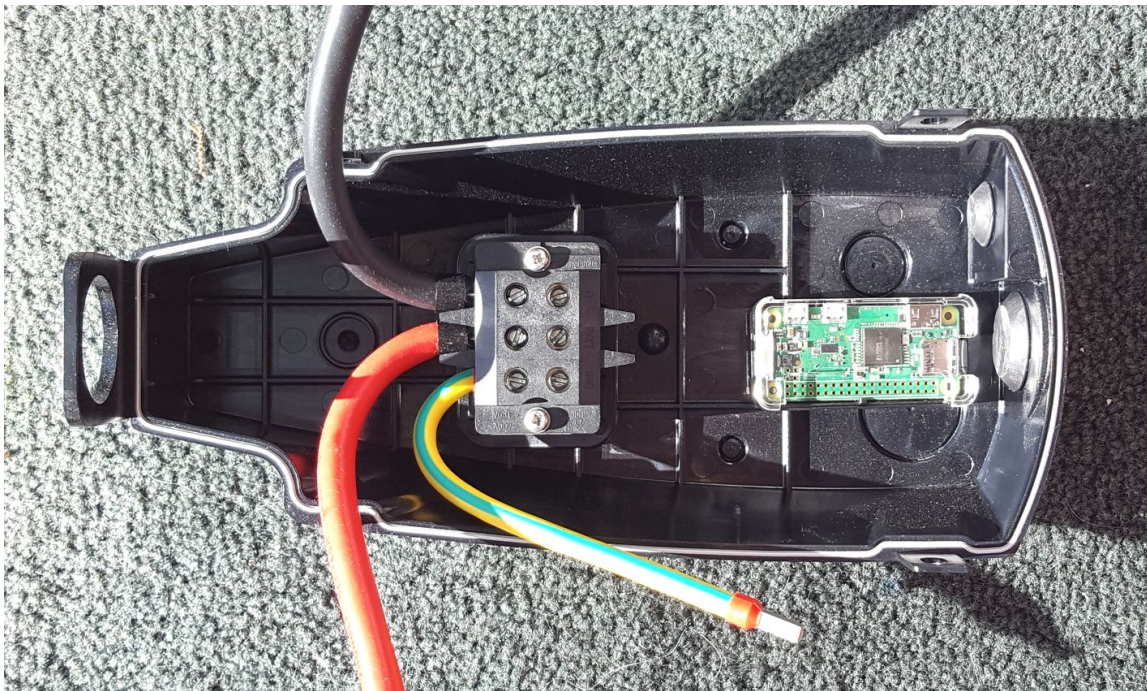
OF, OR IN CONNECTION WITH THE GUIDE OR THE USE OR OTHER DEALINGS IN THE GUIDE.

Cost

You can build your own TWCManger hardware with the instructions below for around \$70 shipped, assuming you already own various tools and parts like a soldering iron, multimeter, power supplies, and adapters.

If you hire an electrician, let them decide the parts you need. You will need to provide a Raspberry Pi with wifi, an RS-485 adapter, and a wall to USB power supply. I still recommend a Pi Zero W for its lower power use and its stability (no crashes in 2.5 months). I wouldn't expect an electrician to need more than 2 hours for the install unless they decide they need to run wall power from a distant location. In that case, ask if it's legal to tap into the wall power wires inside the TWC instead. Point out that the TWC comes with an extra spacer box that fits on the back that may have room to house the Pi.

Spacer box with a Pi Zero W at the right end for scale:



Once the hardware is installed, you will still need to tweak the software to connect with a source of green energy data in order to use the **Track green energy** charging feature.

If you have trouble with any of the steps in this guide, please post a question in the [support thread](#) after checking if anyone has already asked about that particular problem.

Tools and materials required

- The Pi that controls the TWC must be installed in range of a WiFi network you have access to. Alternately, you might be able to set up the Pi to be accessible via Bluetooth or via an “Ad Hoc” WiFi connection, but that’s beyond the scope of this guide.
- [USB-RS485-WE-1800-BT](#) RS-485 adapter \$30
 - o You can also try a \$7-\$10 adapter such as [JBtek](#), but the one I bought stopped working after a couple months and this is a common complaint in reviews. One user on the TWCManager thread reported a similar cheap RS485 adapter did not work out of the box.
- [Raspberry Pi Zero W](#) \$10
 - o **Do not use an older-model Raspberry Pi if you want to run the Pi from the TWC’s power supply.**
 - o Pi Zero W uses about half as much power as older Raspberry Pis and Wifi is built in which saves even more power and space compared to running a USB Wifi dongle on a Pi Zero (non-W version). Also, Pi Zero only has one usable USB port and that port needs to be used for an RS-485 adapter rather than a Wifi dongle.
 - o Power saving is important to reduce the risk of overheating the TWC’s power supply. Even if you want to risk adding hardware that uses more power, there isn’t much space in the TWC and older Pis will not easily fit.
- [Pi Zero case](#) \$5
 - o This case has fewer openings than the [case I used](#), and that helps ensure exposed conductors in the TWC can’t possibly touch anything through a hole in the Pi case. The case has enough space above the SD card slot for power wires to enter.
- [Micro SD card](#) \$10
 - o You may already have a spare micro SD card. Any size 2GB or higher will work. Card speed shouldn’t matter.
- Micro SD card reader
 - o This is necessary to install the OS on the Pi’s SD card.
 - o Most laptops and many newer desktops will have an SD card reader built in. If you have an SD card reader rather than a Micro SD card reader, most micro SD cards come with an adapter to make the card fit in an SD card reader.
 - o If you lack an SD card reader, buy one [like this](#).
- [USB A female to Micro B male adapter](#) \$2.50

- USB Micro B male power supply
 - o You may already have an extra one of these for charging things like cell phones. Anything that puts out 500mA or more should work fine. I'm able to run a Pi Zero W on a 180mA adapter, but only after configuring it for minimum power user. During setup, you will need more power.
 - o If you need a power supply, I recommend [this one](#) for \$7.50.
 - o If you plan to power the Pi using the TWC's power supply, this power supply is only needed to set up the Raspberry Pi. Otherwise, this power supply will be installed within an enclosure your electrician sets up.
- [Mini HDMI to HDMI adapter](#) \$3
 - o This is necessary to connect the Raspberry Pi's video output to the HDMI input of a standard TV or computer monitor. Almost all monitors sold in the last 10 years or so will have an HDMI input, but you might want to double check.
 - o Once the Pi is installed, this adapter is no longer needed.

The tools below are only required if you plan to install the Pi into the TWC yourself. If you hire an electrician to do the install, nothing below should be required, or it should be provided by the electrician.

- Soldering iron.
 - o I struggled for years with cheap soldering irons that would not consistently melt solder or create a strong joint. Following expert guides online did not help. This project has small enough joints that a cheap iron should work fine, but if you plan to do much soldering or are sick of struggling with cheap irons, get something high-powered and temperature controlled like [Hakko FX888D](#) for around \$100.
- [60/40 leaded solder with rosin core](#) \$6
 - o Lead-free solder melts at a higher temperature and is much harder to work with. Unless you'll be eating without washing your hands after working with leaded solder, I don't see the point of struggling with lead-free.
- 3/4" and 1/8" diameter heat-shrink tubing
 - o Finding a mix that includes both 1/8" and 3/4" sizes seems to be impossible, but you can buy these two or check your local electronics or hardware store:
 - o [5/64" to 1/2" assortment](#) \$8
 - o [3/4"](#) \$2
 - o You can also try electrical tape but low-quality tape will unravel eventually, especially if you don't apply it with proper technique.
- Heat gun, hair dryer, or lighter
 - o Any of these tools can be used to shrink heat-shrink tube.
- [UBEC 5V buck converter](#) \$10

- o This converts the TWC's 14V power source to 5V and comes insulated in thick plastic.
 - o The spec sheet shows this regulator is only 75% efficient at the 0.17A load we're putting on it. It might make sense to look for something more efficient if you feel comfortable doing that. Definitely don't use something less efficient that will put more strain on the TWC's power supply and may heat up and risk melting insulation.
- 22AWG wires. Preferably ~12" of black and ~12" of red.
 - o [25ft solid core black](#) \$3
 - o [25ft solid core red](#) \$3
 - o You can scavenge this wire from a broken ethernet or phone cable, or from other electronics. 22AWG (Average Wire Gauge) is ideal, but wire as small as 28AWG (larger AWG numbers mean smaller wire) can also be used but will be harder to work with. Larger wire will not fit the holes in the Raspberry Pi and may not fit the RS-485 screw terminals, but as long as it's stranded wire, the ends could be trimmed down to fit if you feel comfortable with that.
- [Wire strippers](#) \$7
 - o If this is the only time you ever plan to touch an electronics project, you can get by with careful use of a box cutter to strip insulation from the ends of the few wires that need to be soldered together. Otherwise, I recommend the wire strippers.
- 10-pin IDC (Insulation Displacement Connector) cable parts:
 - o [Male IDC connector](#) \$4
 - o [Female IDC connector](#) \$1
 - o [10-wire ribbon cable](#) \$4 for 5 feet, about 8" needed
 - If you have old IDE hard drive cables you can tear off 10 of their wires and use that instead. Make sure wires are labeled 28AWG or 26AWG and are 0.1" apart. I've seen IDE cables in thick and thin styles and if your IDE cables look thinner than what's in the picture at the link above, I don't think they'll work.
- One [velcro wrap](#) or plastic zip tie to hold wires inside TWC.
- Flat-head screwdriver 2-3mm wide. Size "TR20" or "TR25" should work.
 - o This is used to tighten screws that hold wires into the RS-485 terminals.
 - o If you don't already have a flathead of the right size, I recommend buying one of these sets which includes flatheads and lots of other bits useful for opening and repairing most electronic devices:
 - [38-bit set](#) \$15
 - [65-bit set](#) \$30
 - o Or you can be cheap and get [this](#) for \$1.50
- 1/4" hex screwdriver or socket wrench plus "Security Torx" aka "Tamper Torx" bits size TT20 and TT10:

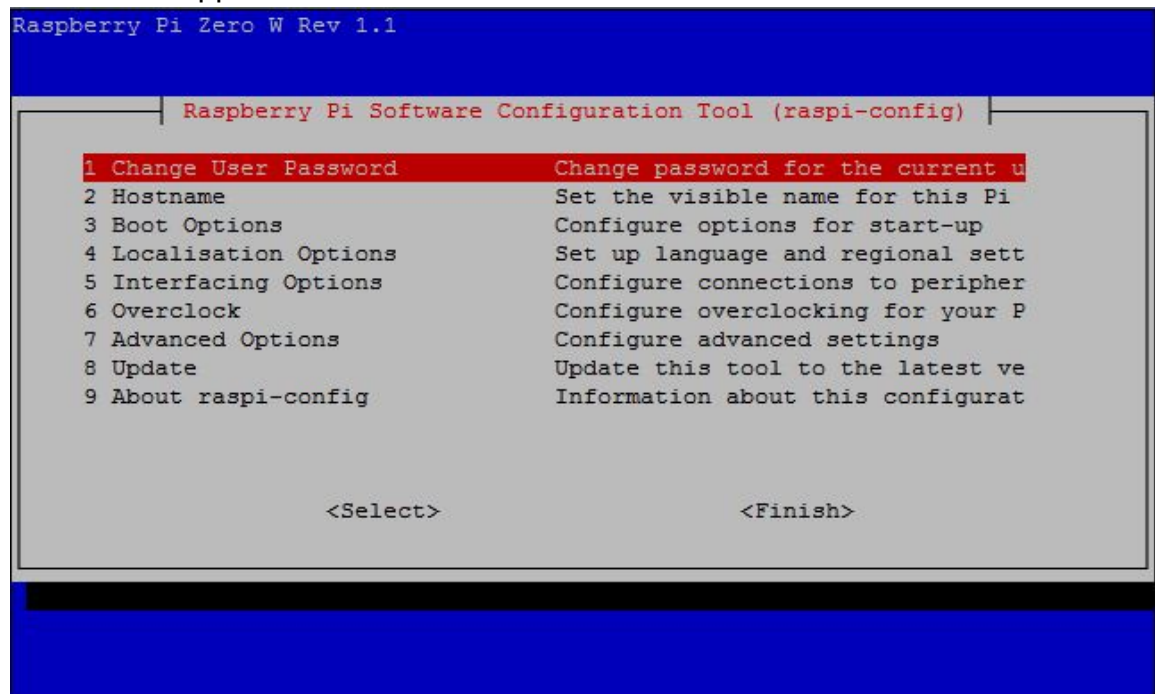
- o The TWC is held closed using six TT20 screws and one TT10 screw. The TWC comes with TT20 and TT10 screwdriver bits in a plastic bag but make sure you have them. They're small, easy to lose, and your installer might have thrown them away or taken them.
 - o If you don't have the bits, buy a kit with an assortment of bits that include TT10 and TT20, such as [this](#). It lists included sizes as T10 and T20 instead of TT10 and TT20 but they're the same thing.
 - o Even if you have the bits, you also need a tool to hold the bits such as [this](#) (which also includes the bits). These 1/4" hex screwdrivers are common and can be found in any hardware store, at least in North America.
- Multimeter
 - o This is necessary to test each pin in the IDC cable you make. The meter will be used to test that each pin at one end is connected to one, and only one pin at the other end. If you don't test, you could end up with something like the 14 volt power pin connected to two pins at the other end which would probably fry the LED board in the TWC.
 - o If you already have a multimeter with continuity test or diode test, you're set. You could also use a battery and light bulb instead of a multimeter for this test.
 - o [Cheap multimeter](#) \$17.50
 - o [Autoranging multimeter](#) \$60
 - o [Clamp meter](#) \$40
 - o If you plan to do other electronics projects in the future, the auto ranging multimeter is significantly easier to use and includes lots of extras like capacitance and frequency measurement. The clamp meter has been even more useful in most of my projects because also does capacitance (but not frequency) and it measures power use of whatever wire you place the clamp around. Unfortunately, it's starting to get flakey after only a year of light use.
 - o A clamp meter is a nice way to double check that your final Pi Zero W setup uses the expected 0.911W.
- Vise
 - o Necessary for squeezing the IDC connectors closed on the 10-wire ribbon cable. You can also use pliers but then you must be very careful to squeeze all four corners of the IDC connector closed at the same rate or you may miss hitting the wires inside the plastic insulation or you may cut through the wires.
- IR thermometer
 - o This is optional but it may help your peace of mind if you can scan an IR thermometer over various components to check for overheating.

- IR thermometers start at [\\$20](#) and normally have a laser to show the center of where you're measuring temperature. Ideally, get one with a mode that shows you the max temperature they've seen since you started holding the trigger.
- I used a FlirONE generation 2 for the temperature images in this guide but it's fairly expensive and prone to battery failure and other problems described [here](#). They now have a third generation but it requires an internet connection during use which is ridiculous. That said, a thermal imager of some sort is definitely valuable if you're into DIY repairs.

Install OS on Raspberry Pi Zero W

- Download [Raspbian Stretch Lite](#).
 - I've linked to the exact version I use (dated 2017-12-01 and based on Kernel 4.9) because newer versions may add features that require more power. Newer versions may also change various things that could make certain instructions in this guide fail.
 - Do not use NOOBS or other non-lite versions of Raspian. They will use more CPU and thus more power and we need to minimize power use.
- Follow Raspberry Pi's [installation guide](#) for installing OS images.
 - Basically, they say to download [Etcher](#), then use it to write Raspbian's .img file to the SD card.
- After writing your SD card, plug it in to your Pi.
- Connect Pi to an HDMI monitor.
- Plug a keyboard in to its Micro USB port nearest the center. Only this port works for USB peripherals. The other USB port only supports supplying power to the Pi.
- Connect power to the micro USB port nearest the end of the Pi.
- You should see a bunch of text appear on your screen. Raspian Lite runs in text mode by default - you won't see a graphical user interface or a mouse pointer. Staying in text mode saves power.
- Eventually, it will say **login as:**
- Type **pi** then hit **Enter**.

- It will ask for a password. Type **raspberry** then hit **Enter**. You won't see anything as you type the password.
- Type **sudo raspi-config** then **Enter**.
- This screen appears:



- Use up and down arrows to change the selected item and **Enter** to activate it.
- Choose **Localisation Options** then **Change Locale**.
 - Scroll down to find your language and country code, and select the UTF-8 version if available. For example, I use **en_US.UTF-8** for English, United States. Press **space bar** to put a star next to the one you selected, then press **Enter**.
 - On the next screen, select the entry you starred in the previous screen.
 - Before you pick the correct localization option, you may notice some keys on your keyboard output an unexpected character.
- Use **Change User Password** to pick a secure password.
- Choose **Hostname** and then type **TWCManager** as the new hostname.
- Choose **Localisation Options** then **Change Timezone**.

- Pick settings appropriate to your location.
- Choose **Interfacing Options** then **SSH**
 - Say **Yes** when it asks if you want the SSH server to be enabled. SSH is how we'll connect to the Pi when it's not connected to a monitor.
- Choose **Advanced Options** then **Expand Filesystem**.
- Choose **Advanced Options** then **Memory Split**. Type **16** then **Enter**.
- Type **Tab** then choose **Finish** and type **Enter**.
- It will ask if you want to reboot. Choose **Yes**.

Set up Wireless access to Pi

- After your Pi reboots, log in again using the new password you set.
- Type **sudo nano /etc/network/interfaces.d/wlan0** then **Enter**.
- **nano** is a text editor. Add the following text to the **wlan0** file:

```
auto wlan0
iface wlan0 inet dhcp
    wpa-ssid "<name of your wireless network>"
    wpa-psk "<password to your wireless network>"
```

- Type **Ctrl-X** then **Y** to exit and save.
- Type the following commands plus **Enter** after each:

```
sudo chmod 600 /etc/network/interfaces.d/wlan0

chown root:root /etc/network/interfaces.d/wlan0

sudo ifup wlan0

ip addr show dev wlan0
```

- The last command should output something like this:

```
2: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UP group default qlen 1000
link/ether b8:27:f5:25:53:1c brd ff:ff:ff:ff:ff:ff
inet 192.168.1.10/24 brd 192.168.13.255 scope global wlan0
valid_lft forever preferred_lft forever
```



```
inet6 fe80::2059:b0a:bb35:c71f/64 scope link  
valid_lft forever preferred_lft forever
```

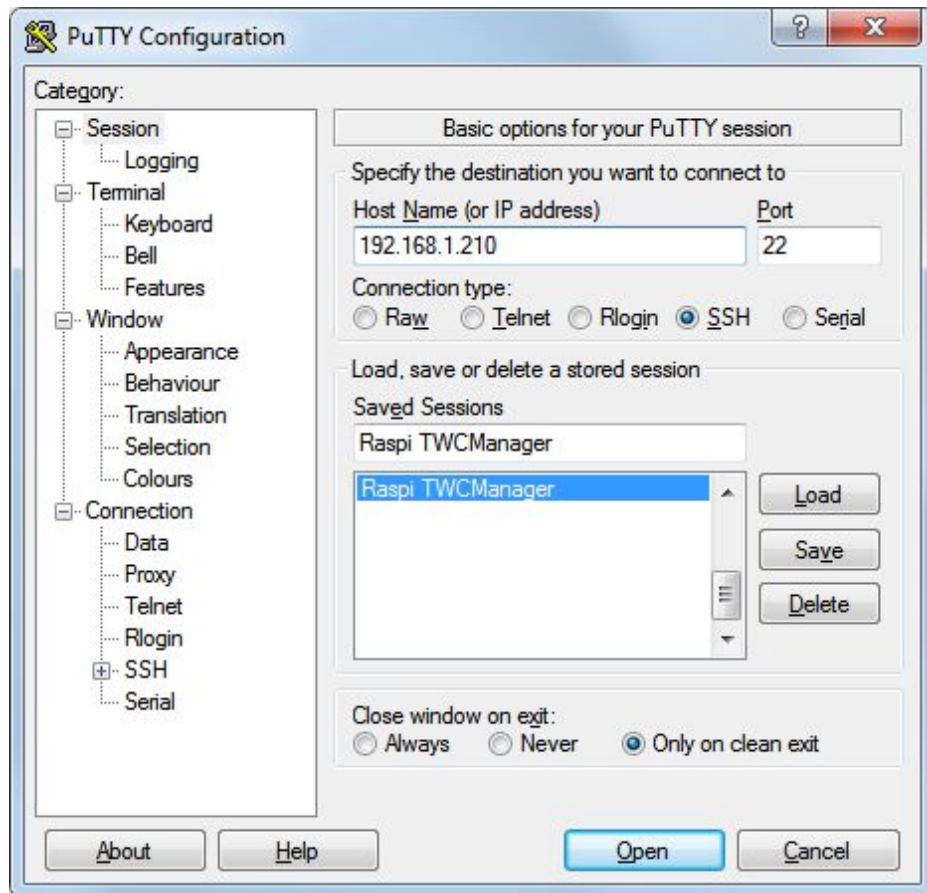
- **192.168.1.10** is the IP address of your Pi. You will need that address to connect to it once you remove the keyboard and monitor.
- Raspberry Pi comes with the **avahi** service which is supposed to allow you to access it by the **TWCManager** hostname you assigned earlier. Try running **ping twcmanager.local** from any machine on your network. If you get a response, you can use **twcmanager.local** to access the Pi instead of using its IP address.
- Personally, I've never had any luck with **avahi** or similar services like Bonjour. I suspect it's the antivirus software I run. If those services work at all, they only work intermittently. So I just access everything by its IP address.
- If you're going to access the Pi by IP address, that address can't change whenever you reboot the Pi. That can be achieved by configuring your wireless router to make the Pi's IP address a static DHCP lease assigned to the **b8:27:f5:25:53:1c** MAC address output above (your MAC address will be different and unique to your particular Pi).
- If you don't know how to create a *static DHCP lease* with your wireless router, you can instead assign the Pi a static IP by editing **/etc/network/interfaces.d/wlan0** to look like this:

```
auto wlan0  
iface wlan0 inet static  
address 192.168.1.210 # Static IP you want  
netmask 255.255.255.0  
gateway 192.168.1.1 # IP of your router  
    wpa-ssid "<name of your wireless network>"  
    wpa-psk "<wireless password>"
```

- The above lines turn off DHCP and tell the Pi to use a particular IP address instead of an address chosen by the DHCP (Dynamic Host Configuration Protocol) server.
- If DHCP assigned your Pi the address **192.168.1.10**, you can usually replace the last number (**10**) with **1** to get the IP of your wireless router: **192.168.1.1** in the example above.
- In most home network configurations, all devices will have an IP where only that last number changes.
- The last number can have a value of 0 to 255. 1 is usually reserved for the router, 0 and 255 should be avoided, and DHCP is

normally configured to choose from a subset of the possible values, say 10 to 100. Therefore, when choosing a static IP address, you want to pick something far away from the number DHCP assigned you. For example, if DHCP assigned you **192.168.1.10**, try making your static IP **192.168.1.210**.

- Before you choose a static IP like **192.168.1.210**, run **ping 192.168.1.210**. If it doesn't say **Destination Host Unreachable**, then something on the network is already using that IP address and you should pick a different number. Type **Ctrl-C** to stop running **ping**.
- Once you have a static IP assigned to your Pi, type **sudo reboot** then **Enter**.
- After it reboots, try connecting with an SSH client.
- In instructions below, replace **192.168.1.210** with your Pi's static IP.
- If using macOS or Unix, ssh should already be installed. Open a terminal (look for Terminal.app on macOS) and run **ssh pi@192.168.1.210**
- If you're using Windows, I recommend using [PuTTY](#). After installing, run PuTTY and set up the dialog like the image below. Click the **Save** button, then click **Open**:



- - The first time you connect, you will get a warning like:
 - The authenticity of host '192.168.1.210 (192.168.1.210)' can't be established.
ECDSA key fingerprint is
SHA256:ljFlkerPjsdfkseFs1jlkdfSAskernal23sasdRsaePI4LKJg.
Are you sure you want to continue connecting (yes/no)?
 - Answer **yes** and it will say it's permanently storing the fingerprint of the Pi's SSH server. This is a security feature that prevents a hostile server from pretending to be your Pi. The hostile server wouldn't be able to provide the same "fingerprint" and your ssh client would warn you the fingerprint doesn't match the fingerprint you just saved.
 - Once SSH is connected, you'll see a command prompt much like you see when the Pi is connected to an HDMI monitor:

```
pi@TWCManger:~ $
```

Configure Pi for minimal power use

- Turn off the HDMI port:
 - Type **sudo nano /etc/rc.local** then press **Enter**. Near the end of the file, before the **exit 0** line, add:

```
/usr/bin/tvservice -o
```
 - Type **Ctrl-X** then **Y** to exit and save.
 - Next time you reboot, no image should appear on your HDMI monitor, but you can still SSH connect to your Pi.
- Turn off the status LED, audio, and Bluetooth:
 - Type **sudo nano /boot/config.txt** then press **Enter**.
 - Press **Page Down** on your keyboard till you reach the end of the file, then add these lines to the end:

```
dtparam=audio=off

# Disable the activity LED on the Pi Zero.
dtparam=act_led_trigger=none
dtparam=act_led_activelow=on

# Disable bluetooth
dtoverlay=pi3-miniuart-bt
```
 - Type **Ctrl-X** then **Y** to exit and save.
 - Next time you reboot, you may see the Pi's green activity LED blink a couple times, but after that it will remain off.

Install the lowest power web server

- The more you can reduce CPU use on your Pi, the less power it will use.
- [This](#) comparison of many web servers found that `lighttpd` uses the least CPU for serving 3 simultaneous users or less. Your Pi may never need to serve more than one user at a time, so `lighttpd` is perfect in this case.
- To install `lighttpd`, type the following commands plus **Enter** after each. Some commands may take a few minutes and output a lot of text. If you

see any error messages, paste them to the [support thread](#) after checking if anyone has already asked about that particular error.

```
sudo apt-get update

sudo apt-get install -y lighttpd

sudo apt-get install -y php7.0-cgi

sudo lighty-enable-mod fastcgi-php

sudo service lighttpd force-reload

sudo chown -R www-data:www-data /var/www/html

sudo chmod -R 775 /var/www/html

sudo usermod -a -G www-data pi

exit
```

- 'exit' above is necessary to apply changes that 'usermod' made to pi's group. You will need to log in again after exiting.

Install TWCManager

- Type the following commands plus **Enter** after each:

```
sudo apt-get install -y screen

sudo apt-get install -y git

sudo apt-get install python3-pip

sudo python3 -m pip install pyserial

sudo python3 -m pip install sysv_ipc

cd ~

git clone -b master --single-branch
https://github.com/cdragon/TWCManager.git TWC

cp TWC/HTML/* /var/www/html
```

- Type **sudo nano ~/TWC/TWCManager.py** then press **Enter**. Read through the licensing notes at the top and the general info if interested. Scroll down till you see **# Configuration parameters**

- Read the notes above **wiringMaxAmpsAllTWCs = 6** and **wiringMaxAmpsPerTWC = 6**. Alter these to values appropriate for your particular TWC installation. **Be careful when choosing values** - the wrong values could trip a circuit breaker or start a fire if your circuit breaker is faulty.
- Type **Ctrl-X** then **Y** to exit and save.
- Start TWCManger automatically when Pi is turned on:
 - Type **sudo nano /etc/rc.local** then press **Enter**. Near the end of the file, before the **exit 0** line, add:

```
su - pi -c "screen -dm -S TWCManger ~/TWC/TWCManger.py"
```
 - Type **Ctrl-X** then **Y** to exit and save.

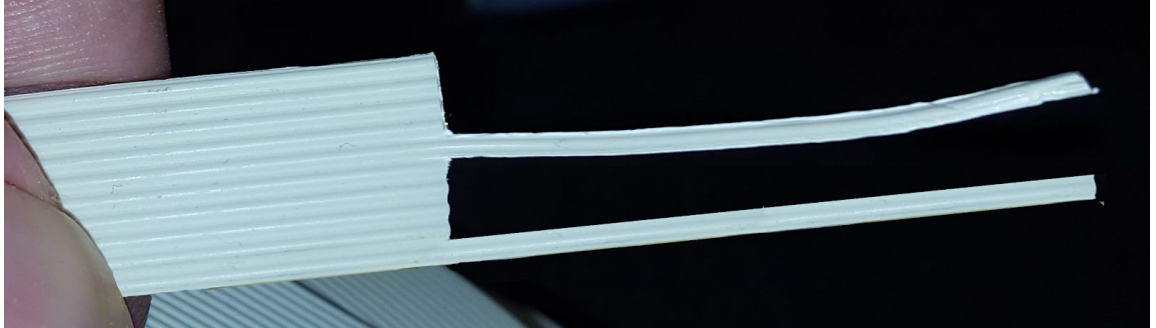
Insulate RS-485 adapter

- Cut 6 inches of black and red wires.
- Use wire strippers or a blade to strip about 1/8" (3mm) of insulation from each end of each wire.
- Screw wire ends into [RS-485 adapter](#). Be sure red wire is in D+ terminal and black is in D- terminal. Tighten screws firmly but not so tight that it feels like something might break. Check the wires can't be pulled out.
- Twist wires together to limit interference.
- Plug RS-485 adapter into [USB A female to Micro B male adapter](#).
- Cover all exposed metal, including the RS-485 screw terminals, in 3/4" heat-shrink tube, then shrink using a heat gun, hair dryer, or lighter flame. End result should look something like this:



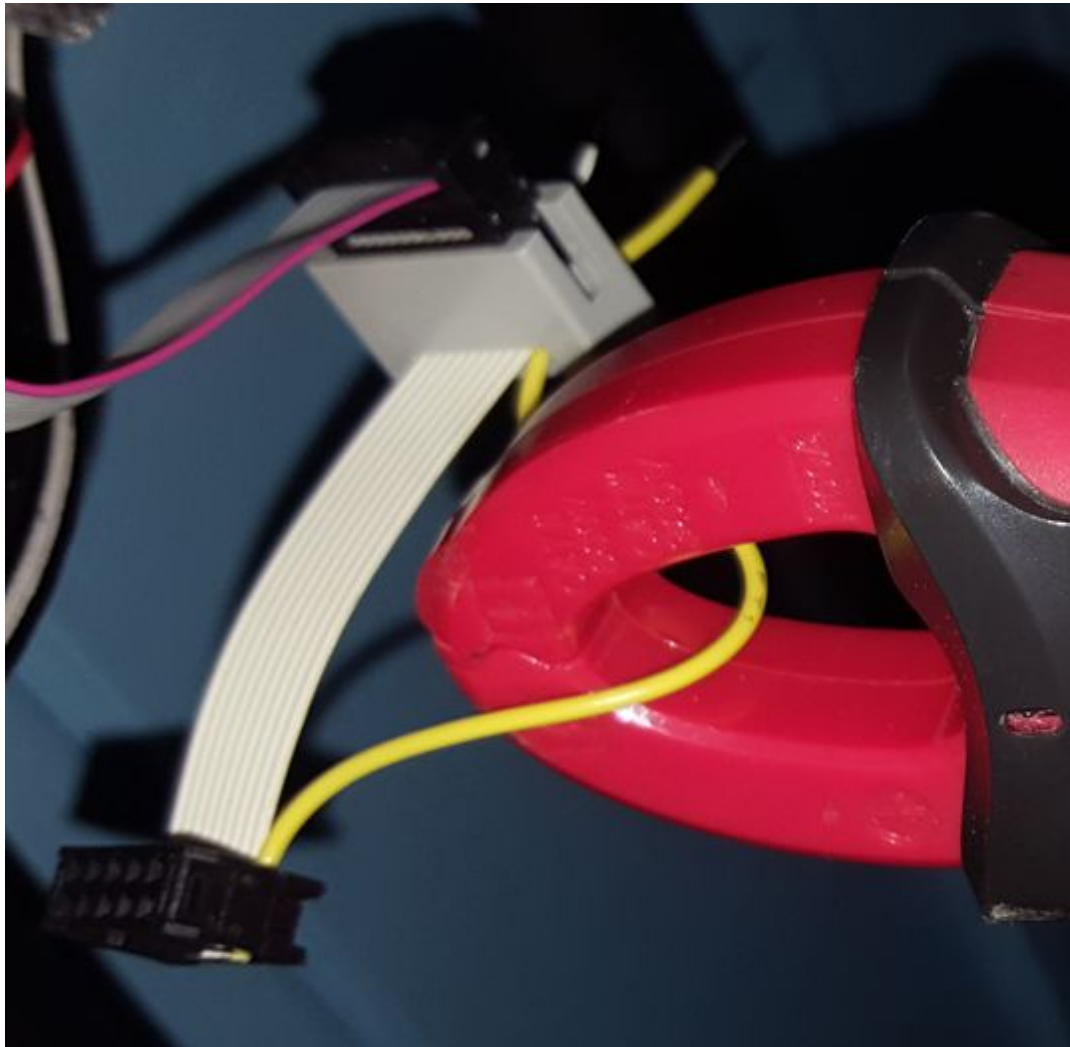
Create power adapter cable

- Cut 7" of [10-wire ribbon cable](#).
- Cut off 2.5" of wire from one end, leaving the 1st and 6th wires at full length so it looks like this:



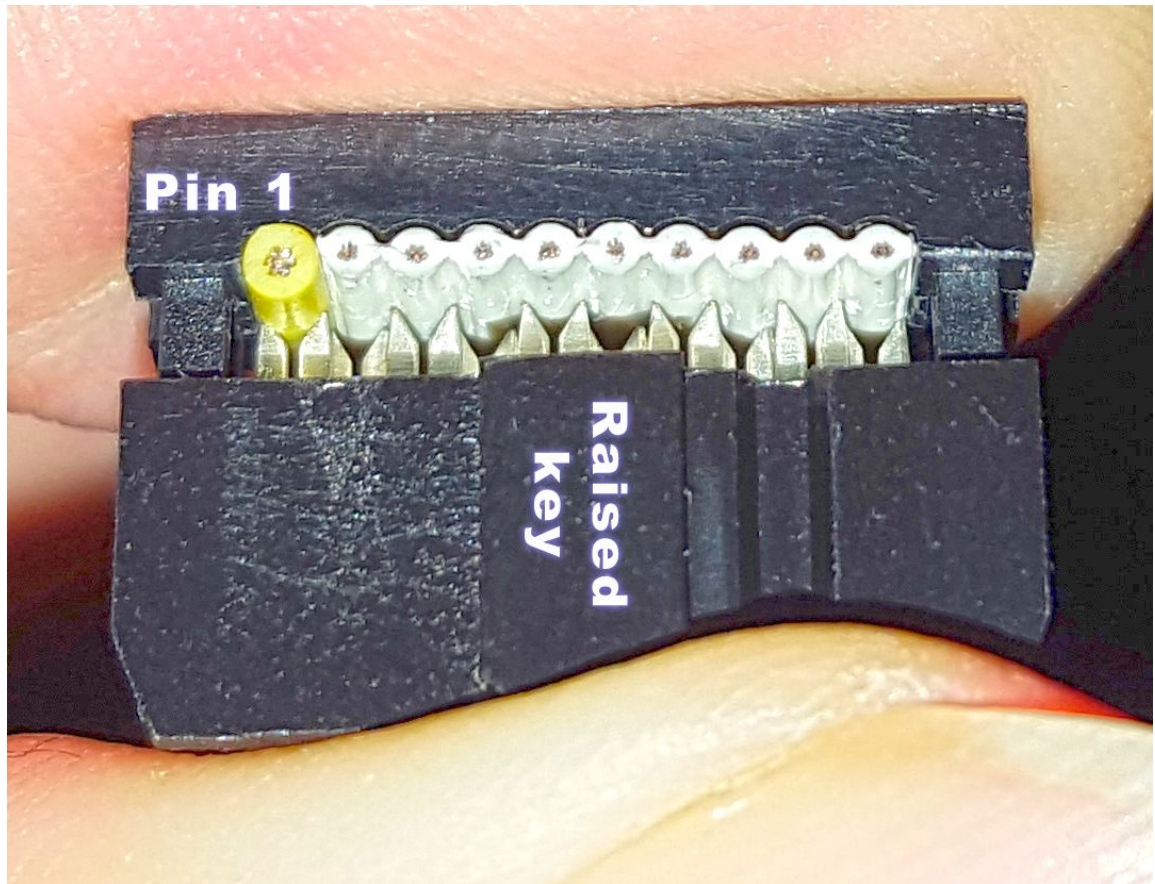
- The lowest wire in the picture connects to **Pin 1** and carries 14V to power the LED board. The middle wire is **Pin 6** which is ground.
- Instead of using ribbon cable for **Pin 1**, I used a long yellow wire which I formed into a loop to fit the clamps of my amp meter in order to measure

power use of the TWC's LED board while it was running:

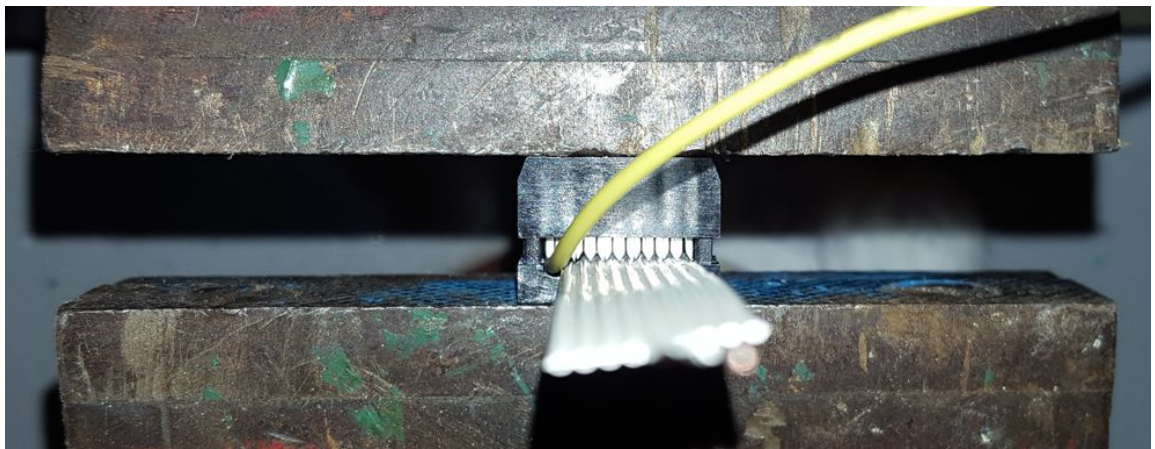


- This will be the cable shown in the remaining pictures. You will have to imagine the yellow wire is just another grey wire from the ribbon cable like in the first picture. Your cable will also be about 50% longer than the one I made which will give you more freedom to position things in the limited space within the TWC.

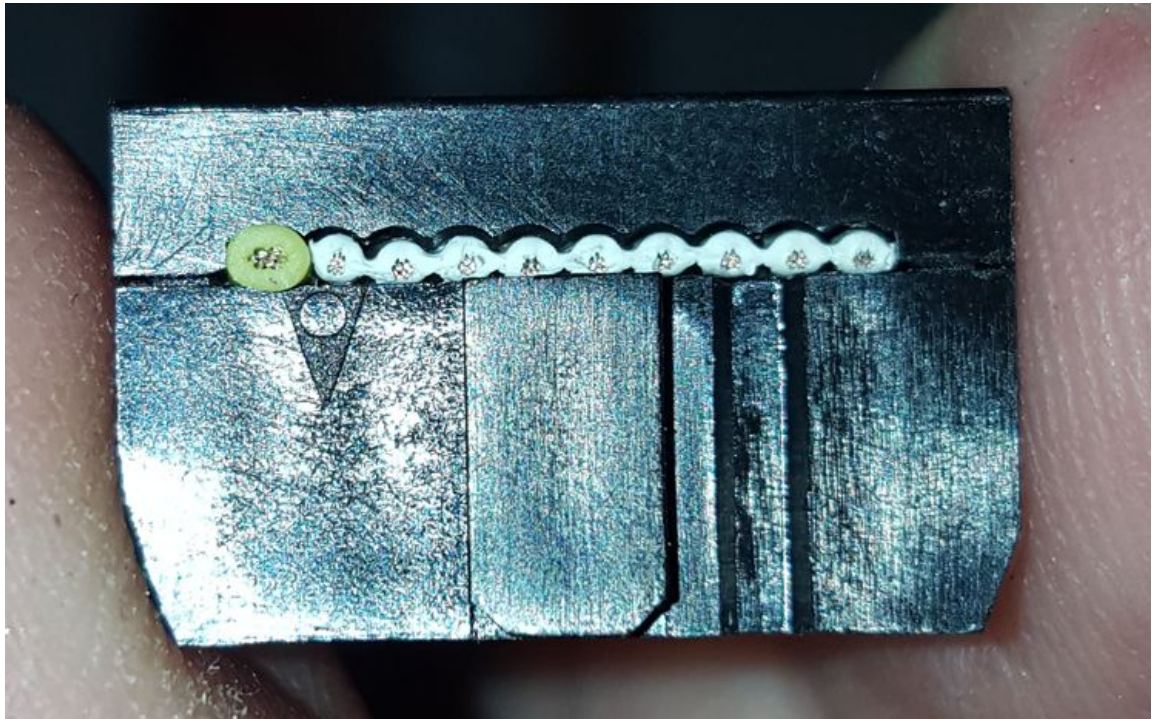
- Position the [Female IDC connector](#) on the end of the ribbon cable without the dangling wires:



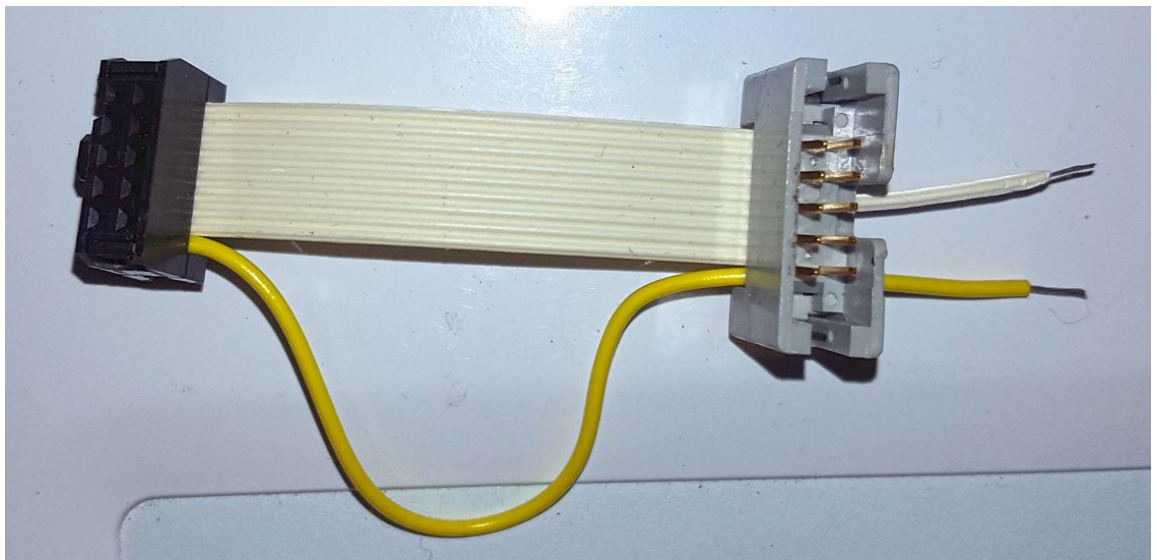
- Make sure the points of the IDC's teeth sit between each wire as in the picture. Also make sure pin 1 is on the left when the plug is pointed down and the raised key is facing you. The raised key fits into a notch in the male plug which prevents plugging the cable in the wrong way.
- Place the connector in a vise:



- Squeeze the vise closed and the connector should look like this:

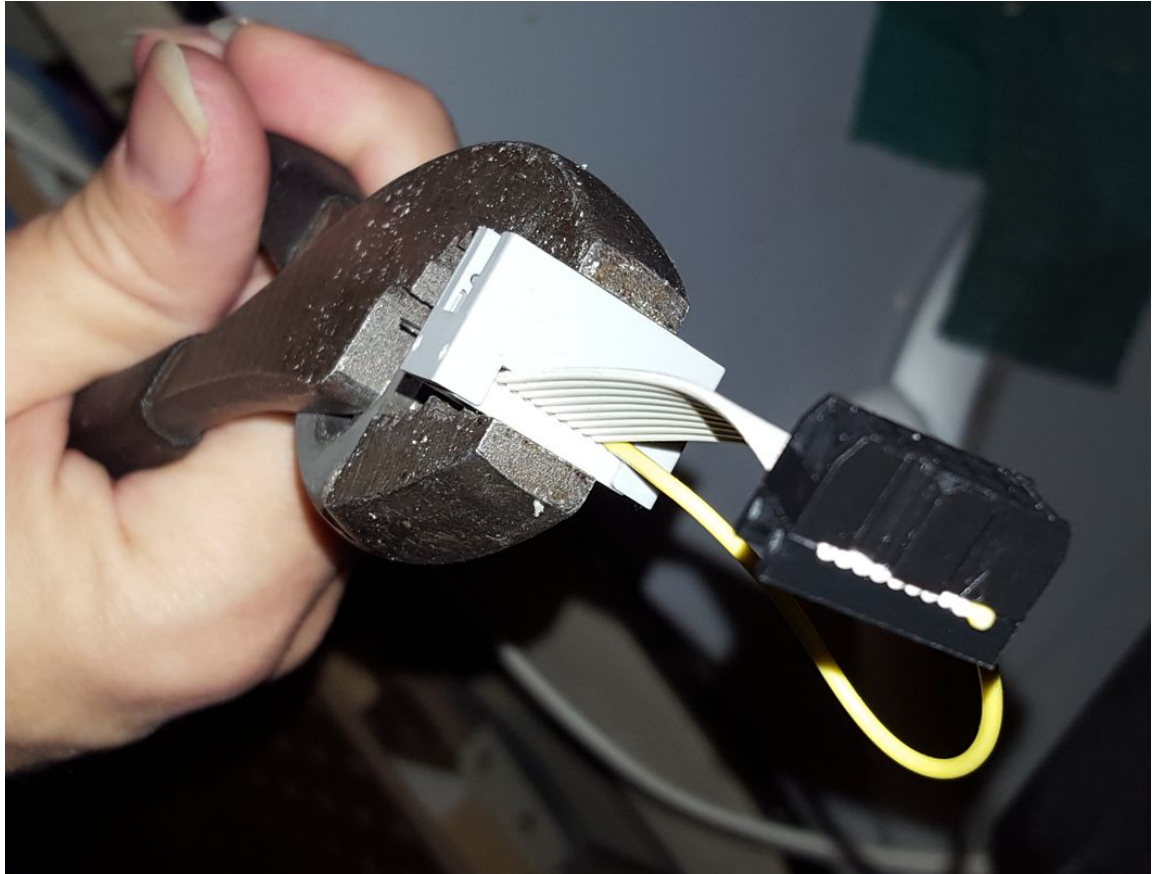


- Place the [Male IDC connector](#) at the other end of the cable:



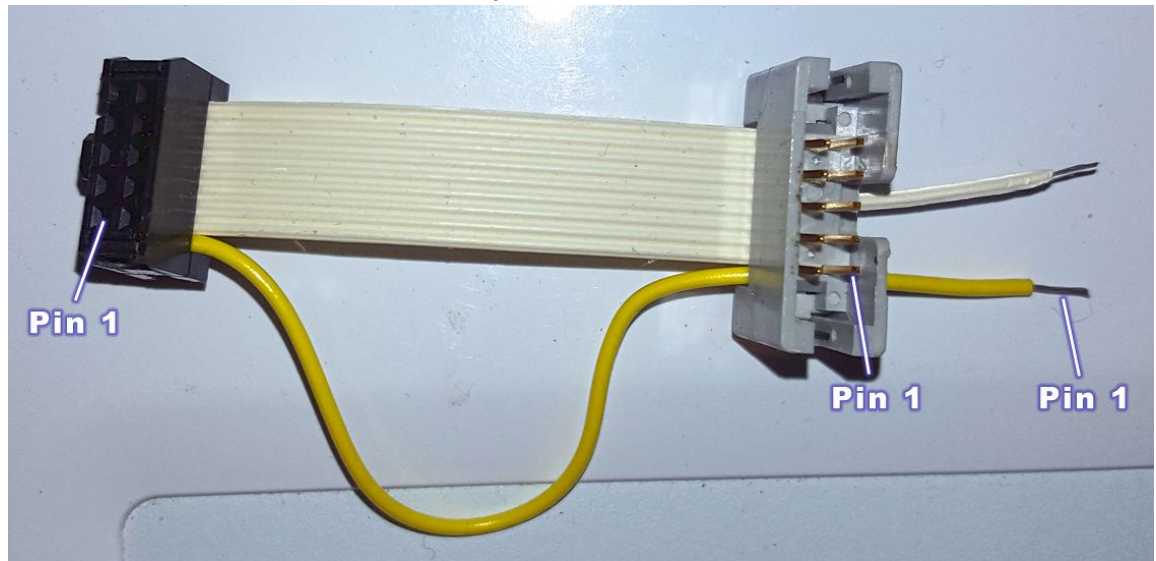
- Make absolutely sure it's oriented just like in the picture with the raised key pointing left on the female connector and the notch for the raised key pointing right on the male connector. If you get the orientation wrong, you will be supplying 14V to the wrong pin of the LED board and it will likely be damaged.
- Squeeze the male IDC connector on with a vise and make sure the points of each tooth sit between each wire in the cable. I had to give it a final

squeeze with a pliers because the vise couldn't squeeze the base in quite far enough:



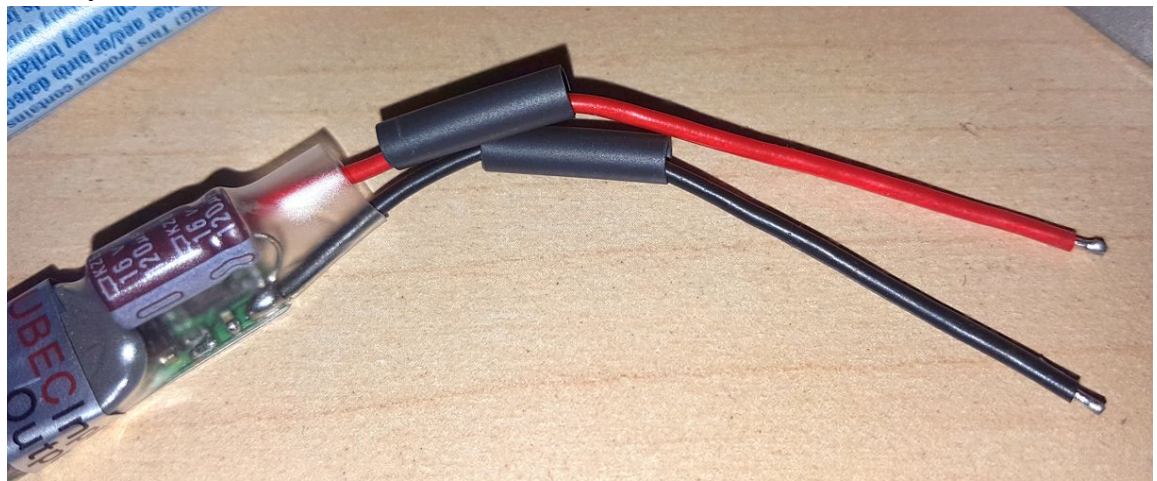
- If you have a multimeter with continuity test, use it to check that each pin on each end connects to one, and only one pin at the other end of the cable. If the connectors were installed correctly, there should be near

zero resistance between the three points labeled **Pin 1**:



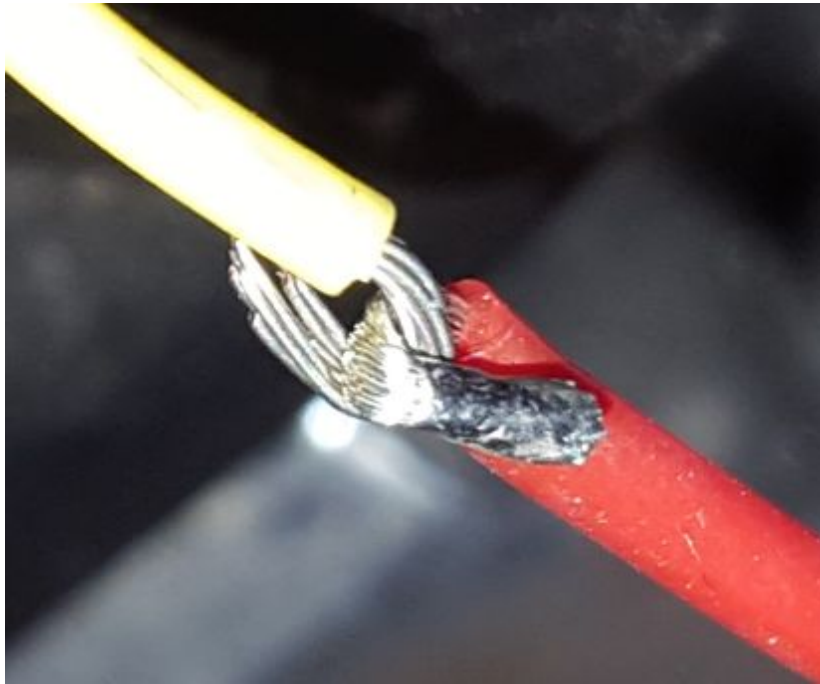
Solder adapter cable to 5V converter

- The 14V from **Pin 1** needs to be converted to 5V to power the Pi. This is done using the [UBEC 5V buck converter](#).
- Cut two pieces of $\frac{1}{8}$ " diameter heat shrink tube about $\frac{3}{4}$ " long. Place them loosely over the thicker black and red wires of the UBEC:



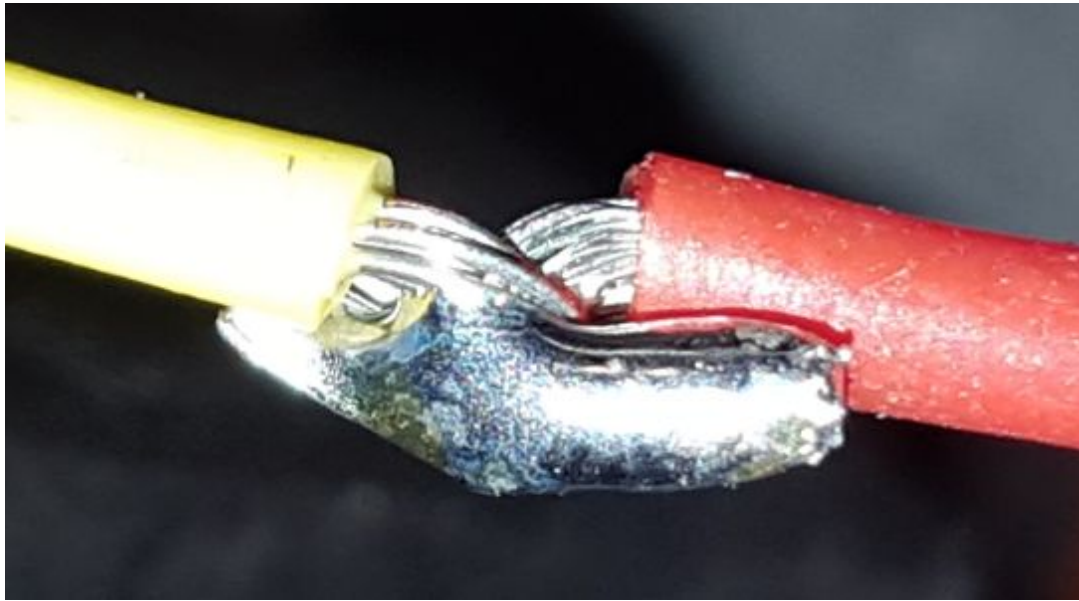
- Bend the end of the dangling **Pin 1** wire from the adapter cable into a hook. Also bend the red UBEC wire into a hook and hook the two wires

together:



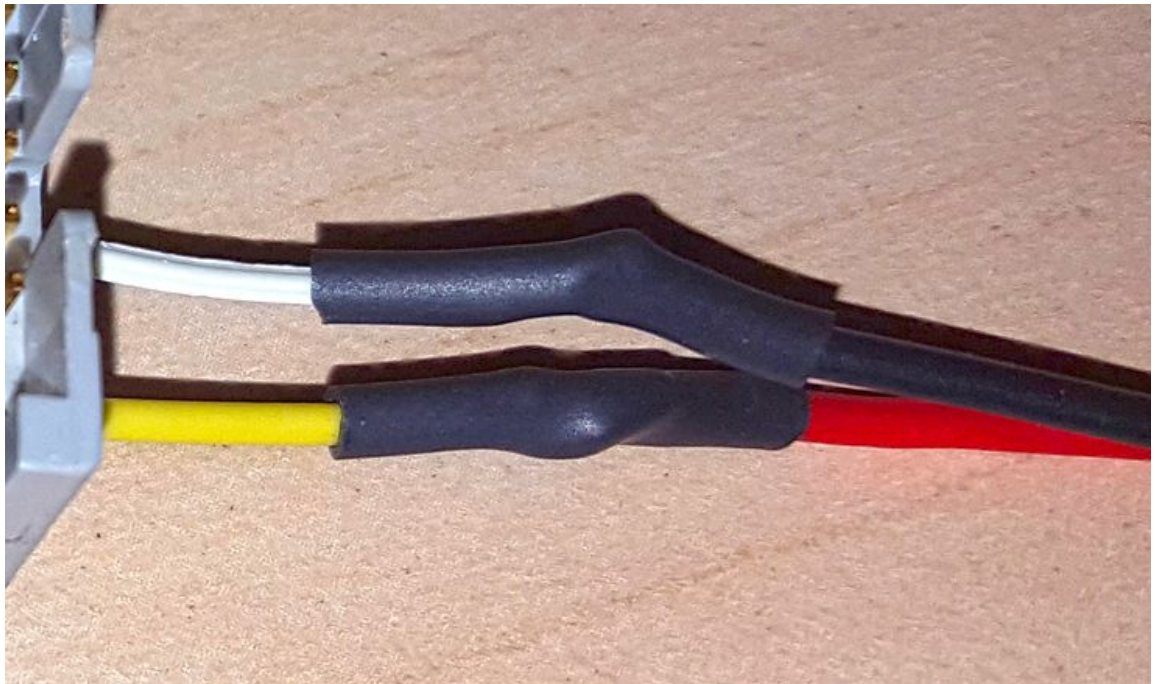
- UBEC's wire ends come stripped and soldered which is why the end of the red wire already looks soldered. I had to strip some extra insulation from the red wire to make the hook large enough.
- The hook can be held taut for soldering if you hold the male end of the cable in a vise and let UBEC dangle from it.
- Solder the hooks together. I set my soldering iron to 800F and touch the tip to the solder, then touch the molten solder on the iron to the wire which instantly transfers heat to the wire. Apply additional solder to the wire while keeping wire in contact with the soldering iron from the older side.

- After soldering it might look something like this:



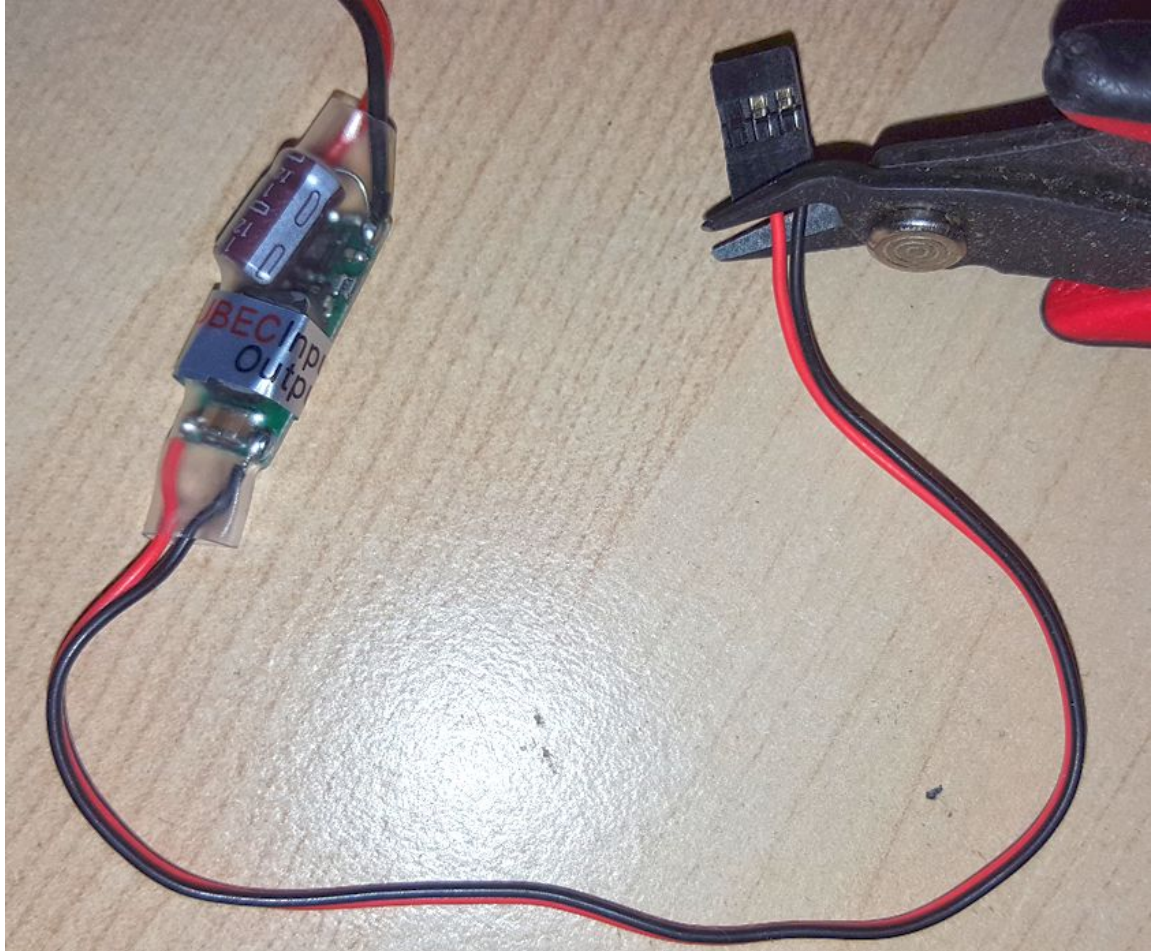
From this angle it looks like two of the yellow wire's strands may not have bonded with the solder. Check for that and add more from the other side if you're not sure.

- Repeat the same process to solder **Pin 6** of the adapter cable to the black wire of UBEC.
- Move the loose shrink tube pieces to cover the soldered areas, then use a heat gun, hair dryer, or lighter flame to shrink them till they look like this:



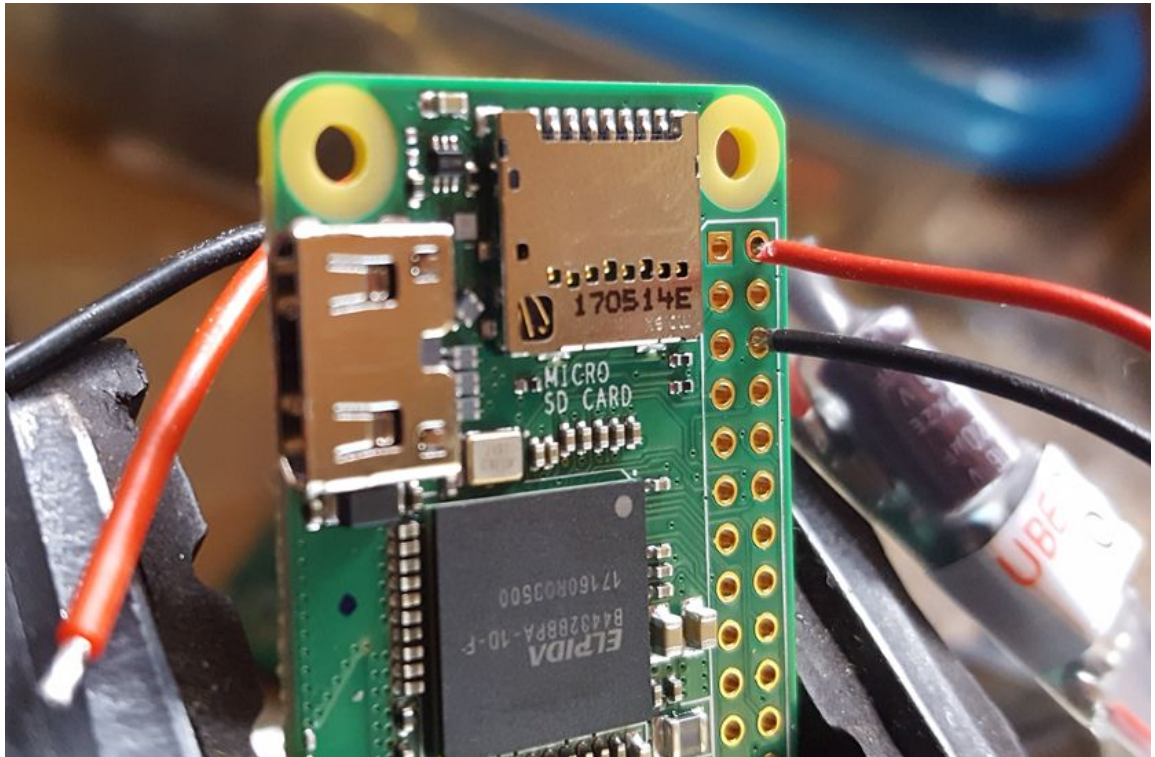
Solder 5V converter to Pi

- Cut the black plug off the thinner wires of the UBEC converter:



- Use wire strippers or a small scissors to strip about $\frac{1}{8}$ " (3mm) of insulation from the cut wires.

- Place the striped end of the red wire into pin 2 from the front side of the pi. I've also placed the black wire in pin 6 but that can be done later:



- Turn the pi over and hold the wire in place while you solder it from the back of the pi. This is easiest if you hold the pi in a [padded vise](#) and hold the wire in a [“helping hand”](#) tool, but you can also ask for help from someone or rig up your own method to at least hold the Pi steady. **Don’t use a helping hand with an unpadded metal clip that will cut into the wire insulation.**

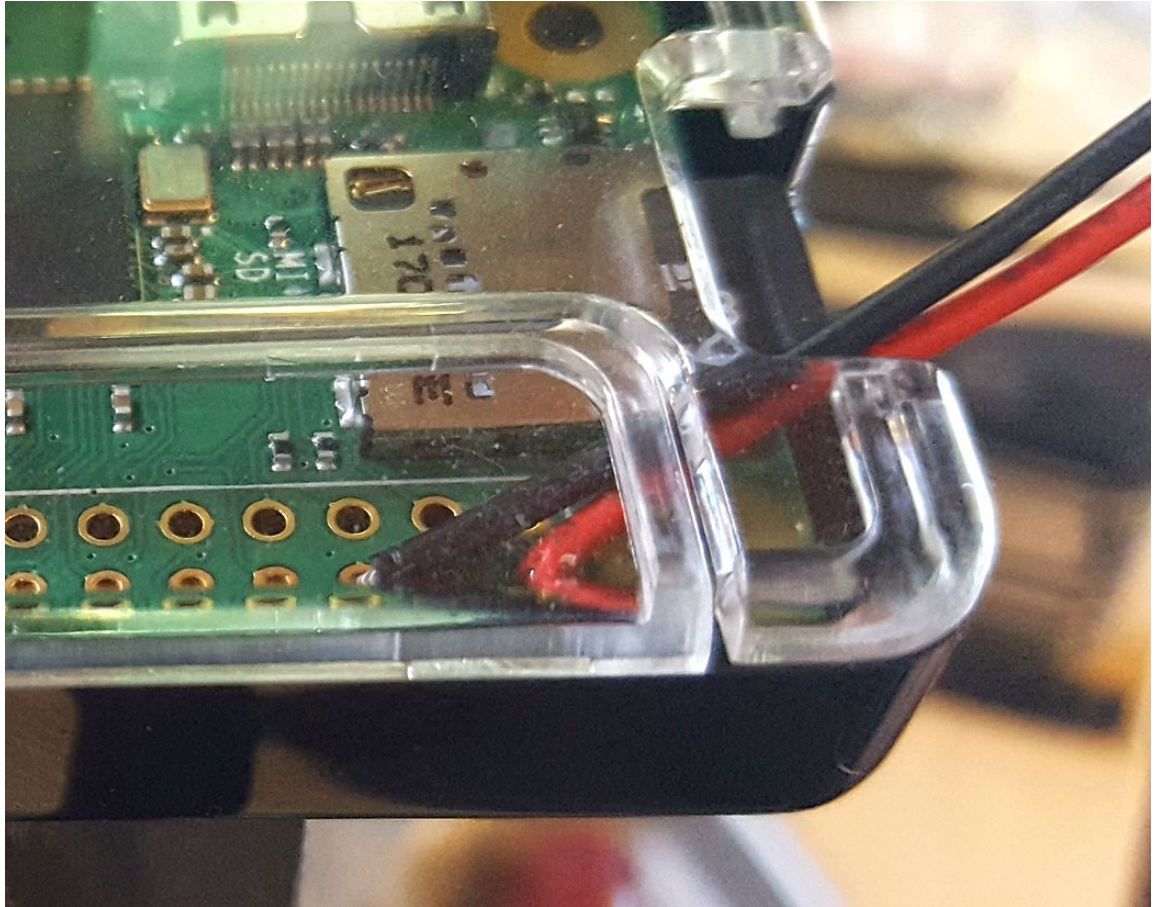
- Solder should form a cone between the Pi's circuit board and end of wire:



In this case I soldered in a lot of wires to a different pi and you can see the variation in each solder cone. The important part is to fully cover the brass-colored ring at the base of the cone along with most of the wire that sticks through the hole.

- Place the stripped end of the black wire into pin 6 from the front of the pi. Solder it in as you did the red wire.

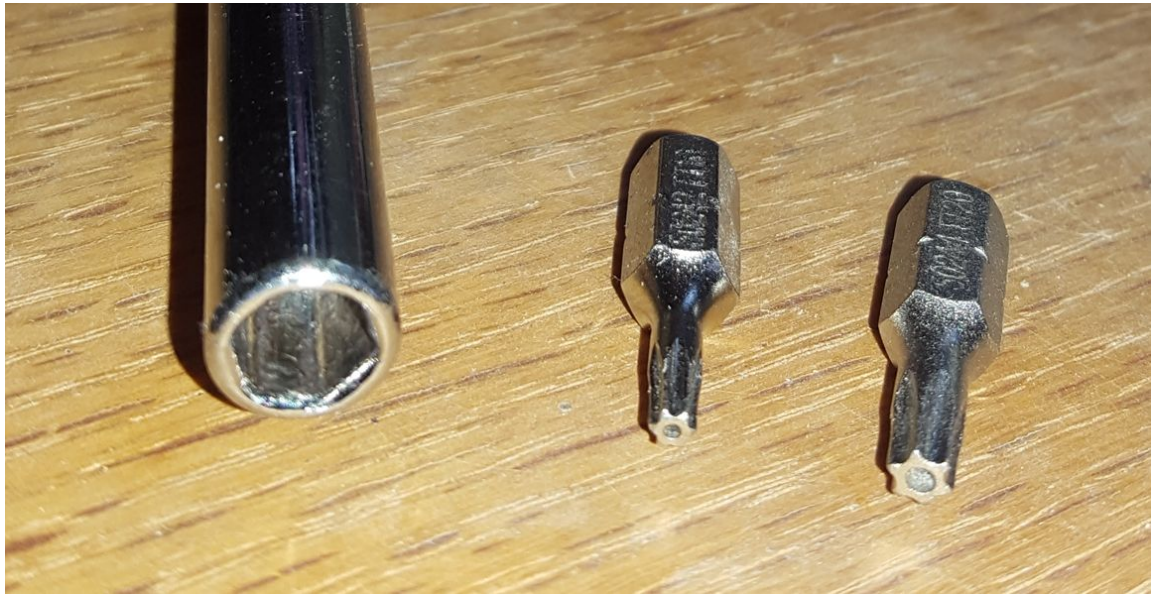
- When done, put the pi in its case and snap on the clear cover with the red and black wires run out of the hole for the SD card:



Install Pi in TWC

- This section of the guide should probably be performed by a licensed electrician in many areas of the world. Use at your own risk. See the Safety and legalities section for more information.
- **Unplug TWC from any car it's attached to, then turn off all power to the TWC from your main circuit-breaker panel.** The green LED on the front of the TWC must turn off. If you see any lit LED, the TWC still has power and is not safe to open.

- Find your 1/4" hex screwdriver, TT10, and TT20 bits:



- Use the TT10 (smaller) bit to unscrew a small screw on the bottom of the TWC. Be careful not to drop and lose this screw because it looks difficult to replace:



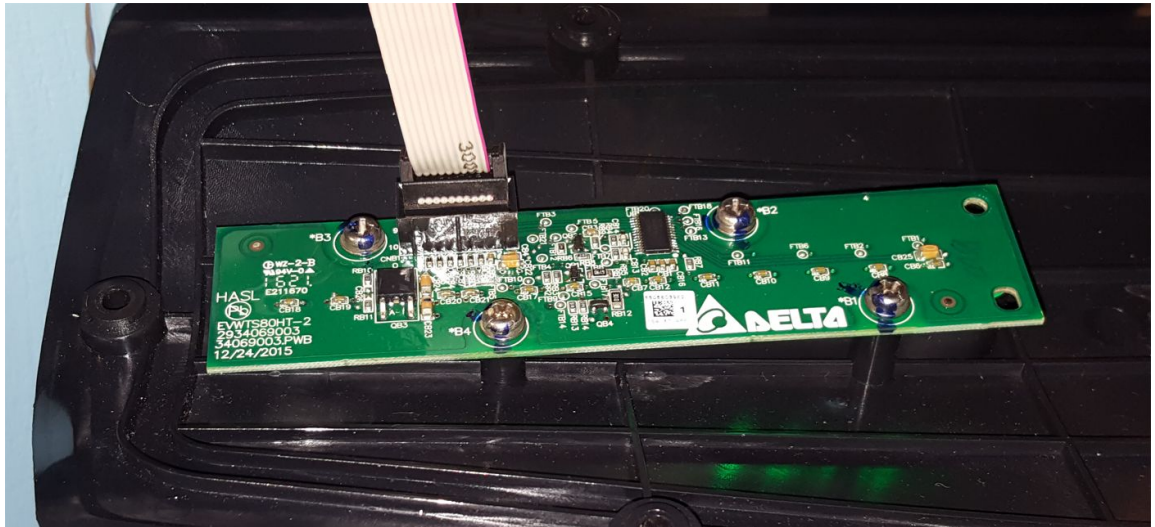
- Remove the silver cover by pulling it firmly at the bottom till the tab that was screwed down pops free. The silver cover then lifts off two grooves

near the top, revealing 6 more screws in a black cover:

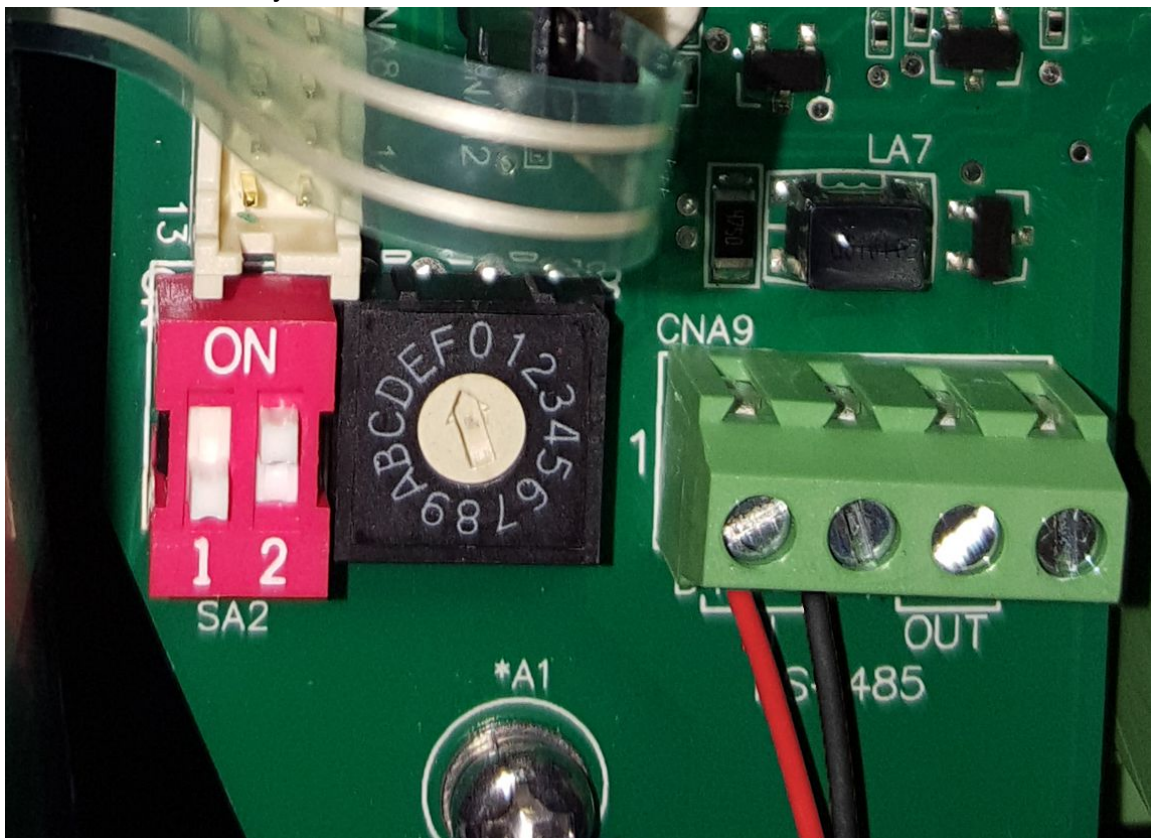


- Switch to the TT20 bit and unscrew the 6 screws holding the black cover on. **Be extraordinarily careful not to lose or damage these screws.** Put down a sheet or something to catch them if necessary. There is a special coating on the back of each screw head that acts as a gasket to keep water from entering the screw hole. I have no idea where you could find a replacement screw other than asking Tesla. You could perhaps add caulk or something behind a standard screw if necessary. The TWC packages of parts do not include any extras of these screws, though I wouldn't be surprised if they started including extras at some point.

- As you remove the black cover, you will see a grey cable is plugged in to a green board behind the cover. The black plug of this cable should be pulled out of the board and it should not take much force:



- Near the lower left of the TWC, find a black square containing a white circle surrounded by numbers and letters:



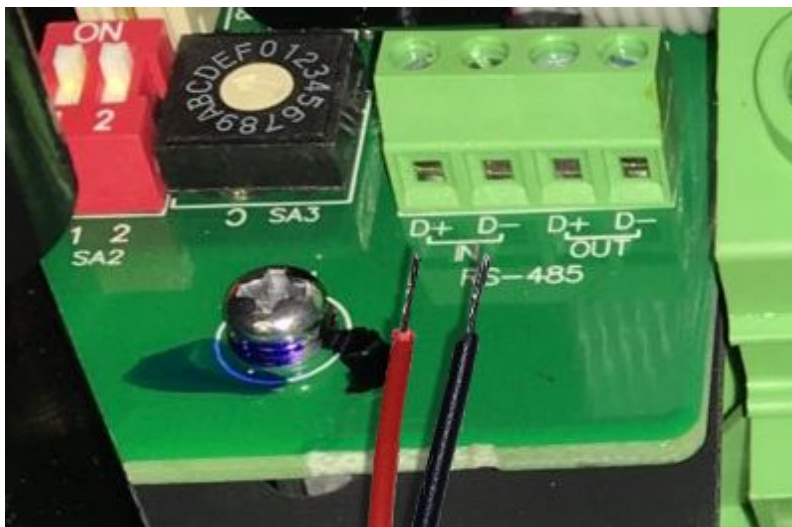
The black square is a “rotary switch” that sets the maximum amperage the TWC is allowed to charge your car at. Write down what number or letter the arrow is pointing at, then search the TWC’s manual for the term

“rotary switch”. That part of the manual will explain the meaning of your number or letter.

- Use a 3mm-wide flathead screwdriver to rotate the arrow in the rotary switch till it points at the F as in the picture above. This sets the TWC into slave mode such that TWCManager running on the Raspberry Pi will control how many amps the TWC charges a car at.
- Find the insulated RS-485 adapter you made earlier. Twist the black and red wires coming from the adapter so they spiral around each other:

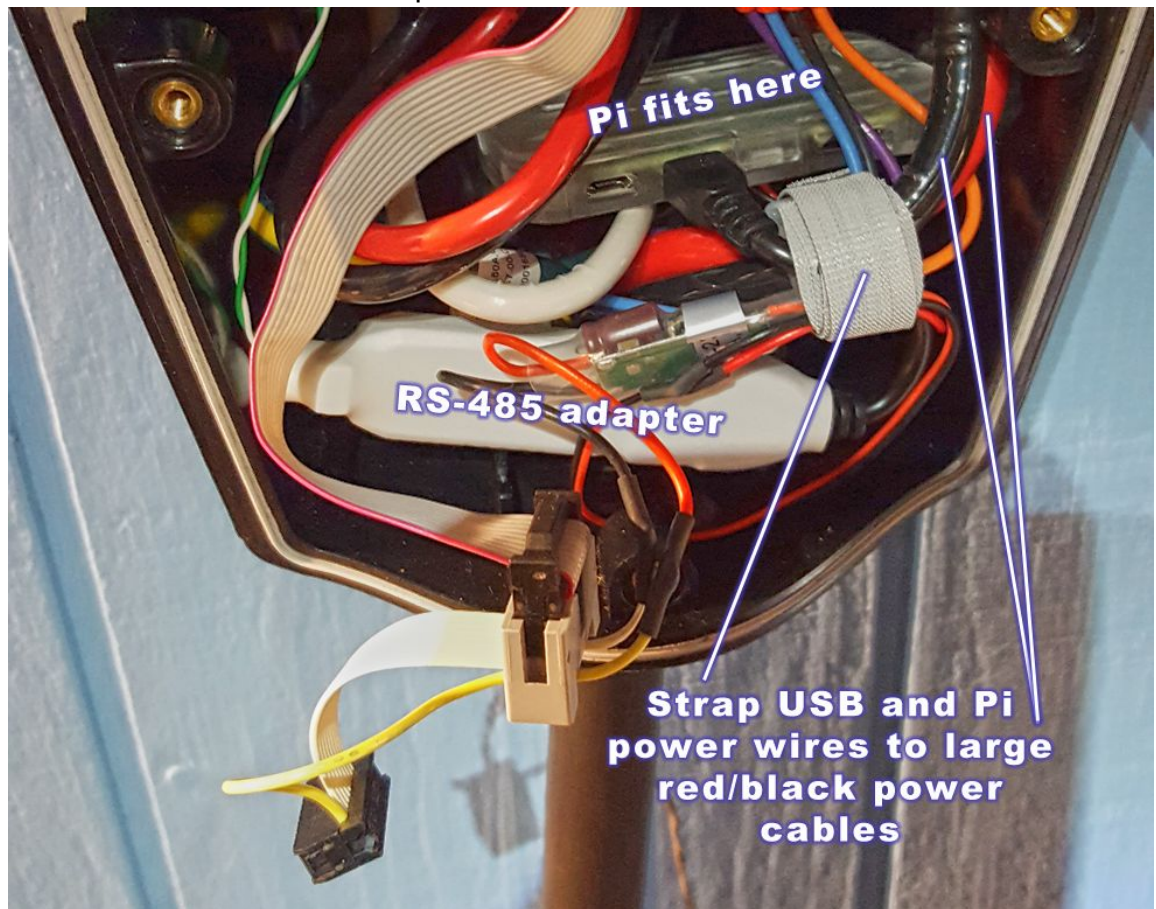


- Place the end of the red wire in either of the holes labeled **D+** in the green terminal block to the right of the rotary switch, and place the black wire in either hole labeled **D-**:



- Use the 3mm flathead to tighten the screws above each hole firmly, then make sure wires can't be pulled out.

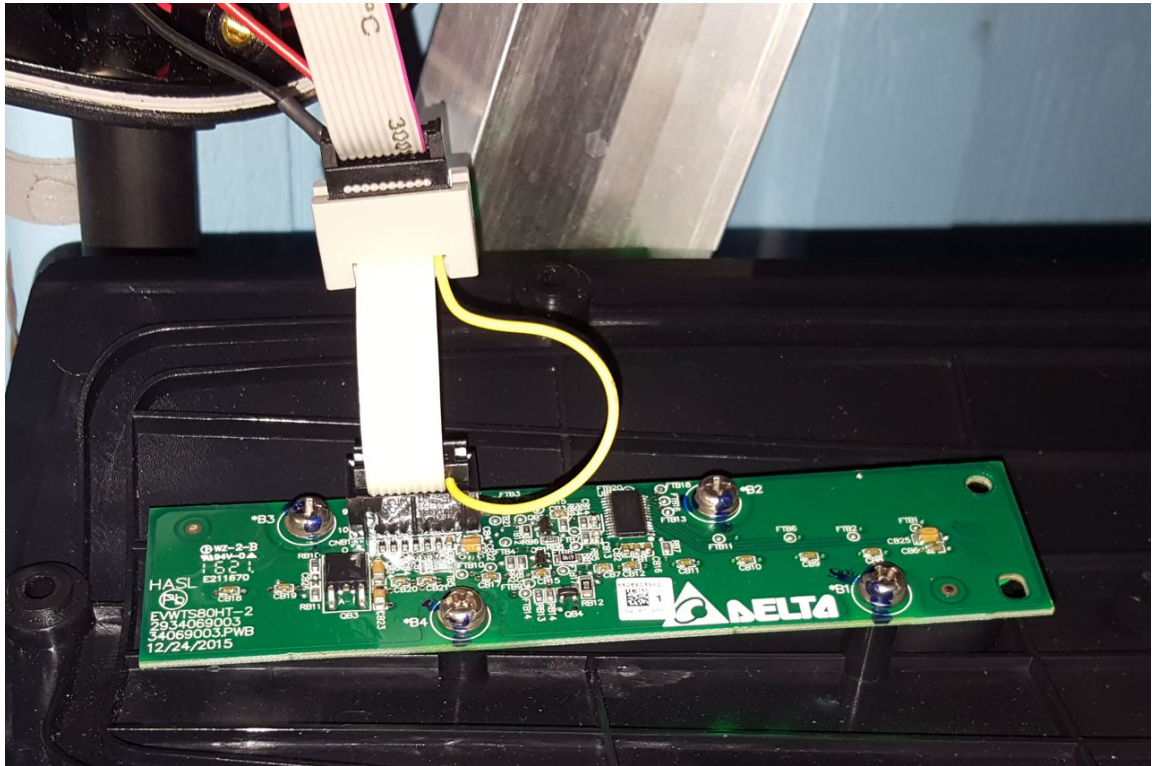
- Place the Pi and RS-485 adapter in the TWC:



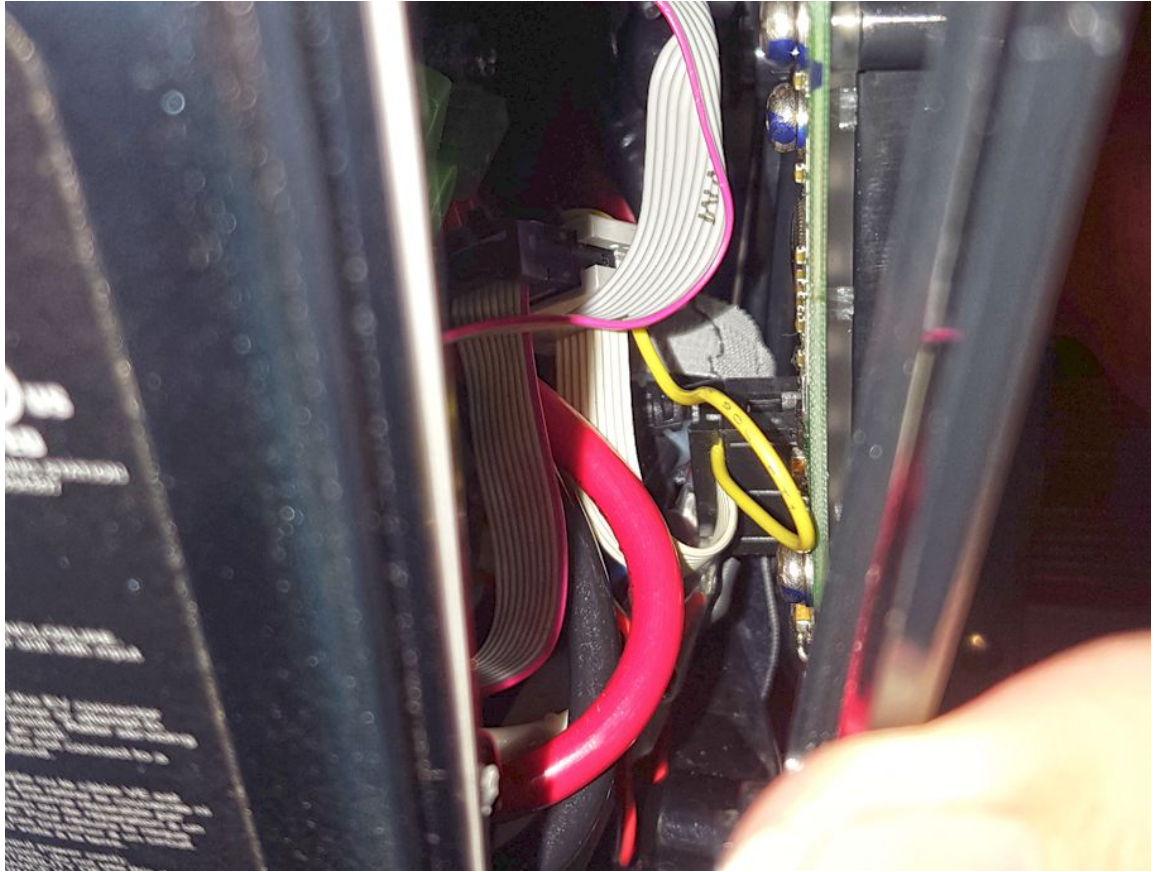
Note that the pi case in this picture is not the case I recommend using, but it's what I used in my install. Also, you may have to fit things differently if the large power wires in your installation are routed differently. **Do not try to make extra room by unscrewing or altering any existing power wires without help from a qualified electrician.** Power wires need to be screwed in to a specific tightness to prevent overheating that may damage your TWC.

- Coil and wrap wires against existing thick wires in TWC using [velcro](#) or a zip tie. Avoid using anything with metal like a twist tie.

- Plug the adapter cable into the TWC's LED board:



- Place the cover with LEDs back on the TWC. Look in from the side to make sure you won't pinch or crush anything when it's fully closed:



- Place the top and bottom screws in loosely. Do not tighten till we've tested TWCManger's control of the TWC.

Test installation

- Turn on power to the TWC at the main breaker panel.
- The green LEDs should flash and change for around 30 seconds. Eventually, the top green LED should remain lit and not flash or move. You will probably see a red LED flash near the center of the LED strip. This happens while the Raspberry Pi is booting. If you continue to see the red LED flashing after 3 minutes, something is wrong with the Pi's control of the TWC.
- If the red LED stops flashing and you just see one green LED at the top remain lit, TWCManger is working. Test its web interface as follows:
 - Open a web browser to the same name or IP address you connected to using SSH during setup. You should see a web page

like this:

Power available for all TWCs: None



TWC 9399: No power available.

Scheduled power: Disabled ▾

Non-scheduled power: Track green energy ▾

1-day charge, 40A

Save

- If you see the above, everything is working and you should skip to the next section: **Close TWC**
- If you don't see the web interface, go back and follow the installation instructions again in the **Install the lowest power web server** section.
- If you can't connect to the Pi via SSH to check the web server installation, it could be because the WiFi signal from your wireless router is too far from the Pi inside the TWC. Try moving them closer together if possible or add a wireless repeater unit near the TWC.
- If the red LED does not stop flashing:
 - Connect to your Pi via SSH.
 - Once connected to the Pi, log in, then type **screen -r TWCManger** then **Enter**.
 - If it says "There is no screen to be resumed matching TWCManger", go back to the **Install TWCManger** section and carefully repeat all instructions.
 - You should see what TWCManger.pl is printing out. A properly-running TWCManger might have output like this:

```
19:43:53: S 032e 0.25/0.00A: 00 0000 0019 0000 M: 05 0000 0000 0000
19:44:25: Solar generating 1Wh so limit car charging to:
          0.00A + 0.00A = 0.00A.
```
 - Look for error messages. If there are a lot of error messages or you think there might be errors above what you can see, you can

write a log of the last 10000 or so lines TWCMManager output as follows:

- Hold **Ctrl** and type **A**
- Type **:** (the colon key)
- Type **hardcopy -h MyLog.txt** then type **Enter**
- Hold **Ctrl** and type **A**. **Release Ctrl**, then type **C** to create a new screen.
- Type **less MyLog.txt** to view the log of TWCMManager's output.
- Return to viewing TWCMManager's output by holding **Ctrl** and typing **A**, then **release Ctrl** and type **N**. This command takes you to the "next screen". **Ctrl-A** then **P** takes you to the "previous screen".
- To quit viewing TWCMManager's output, hold **Ctrl** and type **A**, then **release Ctrl**, then type **D** to "detach" from screen's output. Or you can simply close SSH.


Close TWC

- Cut power to the TWC at the main breaker panel again so the TWC's LEDs turn off.
- Shift the TWC's black cover around and press gently until the white-grey gasket under the cover fits into valleys in the cover. The gasket is what keeps water out of the TWC and it's easy to get the cover misaligned which may crush the gasket in a way that leaves a gap that lets water in.
- Once you're sure the gasket is properly seated in the cover, hold the cover in place while you tighten the top and bottom screws just enough to keep the cover from shifting.
- While you tighten each screw, push the part of the cover nearest the screw against the gasket. This keeps the waterproof material behind the screw head from grinding against the cover until the screw is near fully seated.
- Get each screw tight enough to prevent water penetration, but don't over tighten so much that you risk damaging the waterproof backing.

- With all six screws in the black cover, clip the silver cover onto the top of the black cover, then bend the tab at the bottom until it can slide into place.
- Push the bottom of the silver cover firmly against the black cover as you insert and tighten the tiny screw that holds it down at the bottom. Check that the screw is perpendicular to the surface of the tab as it goes in, not perpendicular to the ground. If you get the angle wrong you may damage the threads. Don't over tighten this screw or you may break the tab.
- Turn power to the TWC on and wait until the red LED turns off before using TWCMManager.

Using the TWCMManager web interface

- Open a web browser to the same name or IP address you connected to using SSH during setup and you'll see this screen:

Power available for all TWCs: None 

TWC 9399: No power available.

Scheduled power:

Non-scheduled power:

- **Power available for all TWCs: None** means a car will never charge when connected to the TWC. Each TWC is identified by a 4-character ID such as **032E** in the case above. If you have more than one TWC, you will see each of them listed, followed by what each one is doing. In this case, **No power available** means the TWC is unable to charge a car.
- The default **Non-scheduled power: Track green energy** setting will never charge a car unless you edit the **TWCMManager.pl** source code to track a source of green energy. It could track output from your solar panels using an API provided by your solar-panel installer. It could track wind energy production from a local wind producer's API. It could track battery state or anything you want, as long as it's accessible via some sort of network interface. Alternately, you could program **Track green energy**

to charge the car at a level that matches average green energy output each hour without tracking actual green energy output. Read the **TWCManager.pl** source code for more ideas, review posts at the [support thread](#) thread, or ask for help there.

- If you just want to charge the car at a certain level between certain hours when energy rates are cheapest or tend to be most green, you can do that immediately using the **Scheduled power** section:

Scheduled power: 40A
from 11:00pm to 5:00am
on days ☒ Su ☒ Mo ☒ Tu ☒ We ☒ Th ☐ Fr ☐ Sa

- In the case above, we set charging to **40A** between **11:00pm** and **5:00am**, Sunday night through Thursday night. The **on days** checkbox only applies to the start time of the charge - it won't stop charging at 12am when it's technically become Friday.
- Be sure to click **Save** after any changes you make.
- You may want to keep these settings even after you set up **Track green energy** to ensure the car charges fully on days when there is too little green energy available.
- If you disable **Scheduled power**, there may still be times when you need a one-time fast charge before returning to **Track green energy**. In that case, set **Non-scheduled power** to a high amp value:

Non-scheduled power: 40A
Resume 'Track green energy' at: 6:00am

- Set **Resume 'Track green energy' at** to any time after you're sure charging will be complete. At that time, **Non-scheduled power** will return to **Track green energy**.
- If you want a one-time charge without returning to **Track green energy**, click the **1-day charge, 40A** button. 40A (or whatever the max power your charger supports) will be provided for 24 hours.

TWCManager with Tesla car API

If you have a TWC purchased after approximately May 2017, it probably has newer firmware on it that doesn't support stopping the car from charging like the older firmware did. In this case, TWCManager uses the Tesla car API to stop the car from charging. You will see this section of text at the bottom of the web interface:

Enter your email and password to allow TWCManager to start and stop Tesla vehicles you own from charging. These credentials are sent once to Tesla and are not stored. Credentials must be entered again if no cars are connected to this charger for over 45 days.

Email:

Password:

TWCManager never stores the email/password that controls the car. Instead, Once entered, it's sent to Tesla and they return a "bearer token" and a "refresh token" which we do store. Both are valid for 45 days. TWCManager uses the refresh token to renew the 45 day period after 15 days.

The bearer token is like a temporary password. If someone breaks into your TWC and steals the bearer token off the Pi and uses it within 45 days, they can do everything they can do with the real password except start the car. Tesla requires entering the real password to start the car. Unfortunately, you don't need the real password to unlock the car.

The first time a car is contacted by TWCManager, its GPS location is saved to **TWCManagerSettings.txt** as **homeLat** and **homeLon**. Only cars located within 2000 feet of **homeLat/Lon** will have their charge started or stopped. This prevents starting/stopping charge on a car plugged in away from home.

If **homeLat/Lon** are set incorrectly because a car was away from home when you first started TWCManager, stop TWCManager and correct **homeLat/Lon** values manually by editing **TWCManagerSettings.txt**. You can also delete the values and start TWCManager only when all vehicles are at home. You may need to start or stop charge via the web UI to trigger the car API to set **homeLat/Lon** values.

Updating to a new TWCManager version

TWCManager can be updated to the latest version without removing any configuration changes or custom code you've added to the program. This is done using the **git** command. git tracks all additions and deletions of lines of code in TWCManger. It automatically adds or deletes the same lines in your local version of TWCManger, and that usually preserves any changes you made. If you happened to change a line that was added or deleted in the latest TWCManger version, you must manually decide how to resolve that problem, and we'll discuss that later.

To set up your own identity in git, run the following commands:

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

These commands only need to be run once and can be skipped next time you want to update to the latest TWCManger version. Obviously, replace the email and name with your email and name. I'm not sure if that info is shared with the github server unless you do something like try to push changes back to the repository but you can use fake info if you want. Mostly the info is displayed if you show a local history of who changed what in the git history.

Run these commands to integrate the latest changes to TWCManger with your local version:

```
cd ~/TWC
```

```
cp /var/www/html/* HTML
```

```
git stash
```

```
git pull
```

```
git stash pop
```

Most people won't see any errors, but if you see errors like:

```
CONFLICT (content): Merge conflict in TWCManger.py
```

```
CONFLICT (content): Merge conflict in HTML/index.php
```

it means you must manually edit the files with merge conflicts (**TWCManager.py** and **HTML/index.php** above). Merge conflicts occur on any line of code that was changed in your local file and in the git repository. git doesn't know whether to

use your version of the line or the git repository version. Generally you want to manually edit the line to contain parts of both changes.

Open each file mentioned in a CONFLICT line in a text editor. Search for <<<<<< and you might find something like this:

```
<<<<<< Updated upstream
    global debugLevel, newGitVariable, \
=====
    global debugLevel, newLocalVariable, \
>>>>>> Stashed changes
```

In this case, I've added **newLocalVariable** to my local code (the line between **=====** and **>>>>>> Stashed changes**), but the latest version (the “upstream repository”) also has a new variable added on the same line called **newGitVariable** (between **<<<<<< Updated upstream** and **=====**). We want to keep both new variables, so we can edit the above code to look like this:

```
    global debugLevel, newGitVariable, newLocalVariable, \
```

After we've eliminated all merge conflicts, run this command:

```
git stash drop
```

Whether or not there were merge conflicts, run this command to copy the new html files back to where the web server expects them to be:

```
cp HTML/* /var/www/html
```