**AC Controller testing with Nissan Leaf Motor (or any motor that has a resolver whose lobes match the number of pole pairs of the motor)**

If your control/driver board was completed by me, it has already been tested. First, let’s verify the wiring to the controller. This is a picture of the surface mount side of the most recent controller (ACControlDriver27.pcb):



You need to build 3 current sensor cables. Let’s say you are using a LEM Hass 200-s. This will give a peak current range of -400amp to +400amp for the phases. Here’s the datasheet for the LEM Hass:

<https://www.lem.com/sites/default/files/products_datasheets/hass_50_600-s.pdf>

Pay attention to the pinout of the current sensor and the pinout of the control board. The current sensor has 4 pins, but you only need pin 2 (current sensor output), pin 3 (ground), and pin 4 (+5v). Pin 1 is not used, so there’s nowhere to plug it into the control board. So, you will need three 4 pin housings like this to mate to the 3 current sensors:

<https://www.mouser.com/ProductDetail/Molex/22-01-1042?qs=5gBG3L055oM6gq%2fgjbx3Gw%3d%3d>

And these pins:

<https://www.mouser.com/ProductDetail/Molex/08-50-0114?qs=%2fha2pyFadujh2f7KQ%252bXWe2d6E0tCKKZpLCtYbdvIlLc%3d>

The other end of the current sensor wire can use the same crimp pins too. You will need a total of 18 for the current sensor cables (none in the unused hole). Make sure you twist the 3 wires together for the current sensor cable to reduce noise. Use 22awg or 24awg wire. A drill works well for twisting the wires together. A shielded cable with the drain wire connected to the ground on one end works well too:



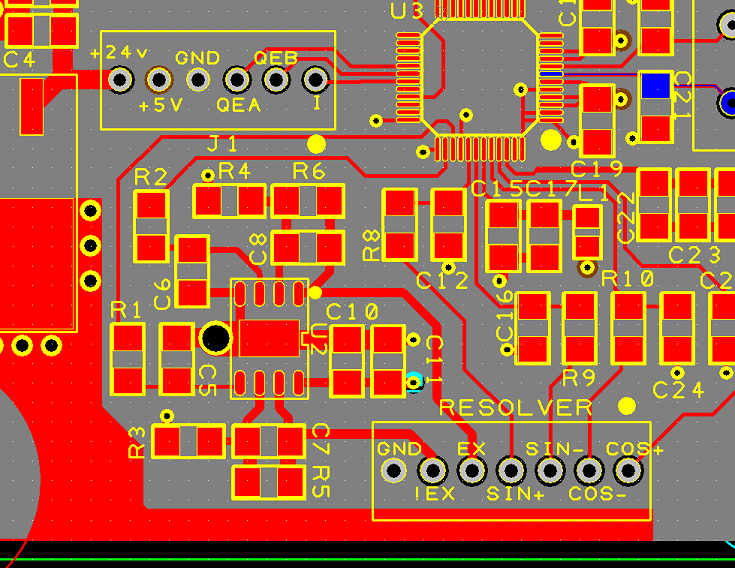
I use the 1.6 and the 1.9 mm for the crimping.

Next you will need a hall effect throttle like this:

<https://www.ebay.com/itm/Throttle-PB-6-Type-0-5V-Hall-Effect-Throttle-Box-generic-/110658990971>

You can get a 2nd one for the brakes if you like, but as of ver. 1.0 of the software, the brake regen is disabled. You only get regen from releasing the throttle. The brake regen will be added most likely in the next software release.

If you are using a resolver to encoder board, you will need to connect the 6 encoder pins on the control board (+24v, +5v, gnd, QEA, QEB, INDEX) to the resolver to encoder board:



You will then also need to connect the !EX, EX, SIN+, SIN-, COS-, COS+ pins of the resolver-to-encoder board to the resolver on the motor. Each motor is different, but here’s the pinout for two types of Nissan Leaf motors (2011 and 2012?):



Next is the forward/reverse pin. This is unimplemented right now, but should be functional in version 1.1. You will need a simple toggle switch (SPST). Flipped one way, the motor will rotate one direction, and flipped the other way will cause the motor to turn the other direction. There are a total of 3 pins for the forward/reverse pin. This is to allow you to use one of the forward/reverse GND connections for an optional shield for the forward reverse cable.

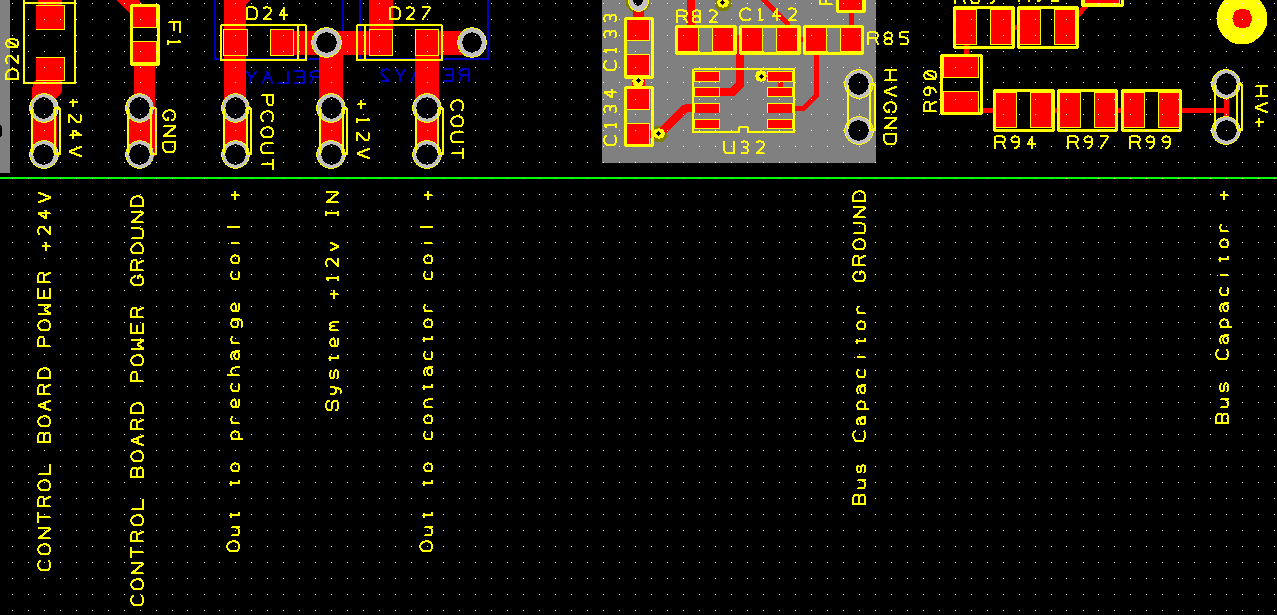
Next is temperature wire 1, and temperature wire 2. Hook up a thermistor like this one to the aluminum base plate, and then connect it to those 2 pins of the controller:

<https://www.mouser.com/ProductDetail/Vishay-BC-Components/NTCALUG03A103GC?qs=sGAEpiMZZMuBd0%252bwiCVS25H%2fdmXUb6DYHYXWCB%2fFtJk%3d>

All that matters is that the B parameter is about 3984K and it is a 10k thermistor. Temperature shield GND is an optional pin for for the shield of the thermistor wires, if you use a shielded cable.

The next 3 pins are CAN GND, CAN LOW, CAN HIGH. This is unimplemented as of version 1.0. In order to use CAN at a standard bit rate such as 250kbps, the crystal oscillator (XTAL1) needs to be switched from 7.3728MHz to a 7.5MHz crystal, and also the USB data rate needs to be reduced to 38.4kbps.

**Now for the spade connection wiring:**



Using a DC/DC such as this, set the output (there’s a pot on the DC/DC) to about 24 to 25vDC:

<https://www.mouser.com/ProductDetail/MEAN-WELL/RSD-60G-24?qs=sGAEpiMZZMvGsmoEFRKS8Koqt8Pjkl39Gn1bpxbR3LLvSpJ6kqonkQ%3d%3d>

Connect the pin labeled +24V to the +24v on the DC/DC. Connect the 24vDC ground to the pin labeled GND. Connect PCOUT to the positive side of the precharge relay coil.

For a precharge relay, I would suggest the P115-BDA from Gigavac:

<http://www.gigavac.com/sites/default/files/catalog/spec_sheet/p115.pdf>

For a precharge resistor, I would suggest something like this:

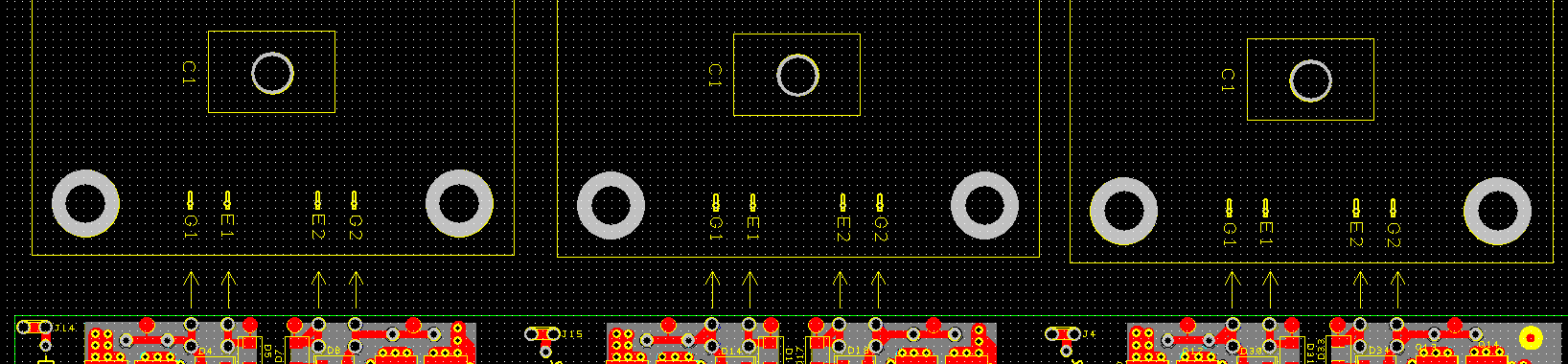
<https://www.mouser.com/ProductDetail/Ohmite/AZ151KE?qs=sGAEpiMZZMvmQ%252bOLa8n%2fM8YMnCQ5o%252b6E0Td1E32eWcg%3d>

Connect the negative side of the precharge and contactor coils to the 12v system ground (usually a car has a system voltage of +12v. Pick a contactor that will energize from 12vDC). On the pin labeled +12V, connect the car’s +12v. Connect COUT to the positive side of the main contactor’s coil. For a main contactor I would suggest this (the one that has a 12v coil):

<http://www.gigavac.com/sites/default/files/catalog/spec_sheet/gv200.pdf>

Next, connect the bus capacitor’s high voltage ground to HVGND, and the bus capacitor’s high voltage positive to HV+ on the control board. This is for DC voltage monitoring, as well as introducing a bleeder resistor into the system.

Next, the 3 IGBTs plug directly into the control/driver board (directly underneath the PCB):



Next, the 3 connectors labeled J4, J14, J15 get connected to the bus capacitor +. That’s all the wiring!

**Debugging:**

To verify that the control/driver board is working like it should, make sure the high voltage is disconnected, and load the file debugging.hex that is included with this .zip package (see the section of this document titled “Programming” to see how to program the chip). Then, measure the 6 rows of 3 test points per row across the top of the board (each row is labeled -8v, 0v, 15v). There are 3 test points on the left and 3 on the right of each “-8v”, “0v”, “15v”. There will be a row of those 3 test points repeated 6 times across the top of the board, for each of the 6 supplies. Measure from 0 to -8v, and 0v to 15v for each of those 6 “three hole rows” to verify that all 6 supplies are working. Next, measure from “gate” to “0v” across the top of the control/driver board (measure all 6 pairs of those) and verify that you are getting about +3v from “gate” to “0v” for each of those 6 supplies. If those are good, program the board with the .hex file attached to this .zip package.

Now, to test that the throttle is working, in realterm type:

stream-raw-throttle 1

Data-stream-period 100

Now, turn the throttle all the way down, and note the number for the raw throttle (between 0 and 1023). Next turn the throttle all the way to 100%, and note the number for the raw throttle. To stop the stream, just type “off”. Let’s say that your low and high numbers were 85 and 879. Then, a good configuration for the throttle mapping would be to type the following in realterm:

max-regen-position 85

min-regen-position 220

min-throttle-position 300

max-throttle-position 879

save

That would set the throttle map so that 0% throttle gives the maximum regen current, 220 would be the starting point for the regen, 300 would be the start of the positive torque, and 879 would be the maximum torque.

To verify that the resolver to encoder board is working correctly with the control board, connect the resolver to encoder board to the motor, and to the control board. Then, with no high voltage attached, connect the control board to Realterm using the USB plug on the board. The settings are 115200bps, 8N1 (8 data bits, no parity, and 1 stop bit) :

<https://sourceforge.net/projects/realterm/>

Then, turn on the 24v power to the control board. Then, type the following in Realterm:

stream-rotor-flux-angle 1

data-stream-period 100

Then, turn the motor shaft by hand, and you should see the rotor flux angle go from 0 to 511, and then start over again at 0 (or it might go backwards).

Connecting the IGBTs to the outside world:

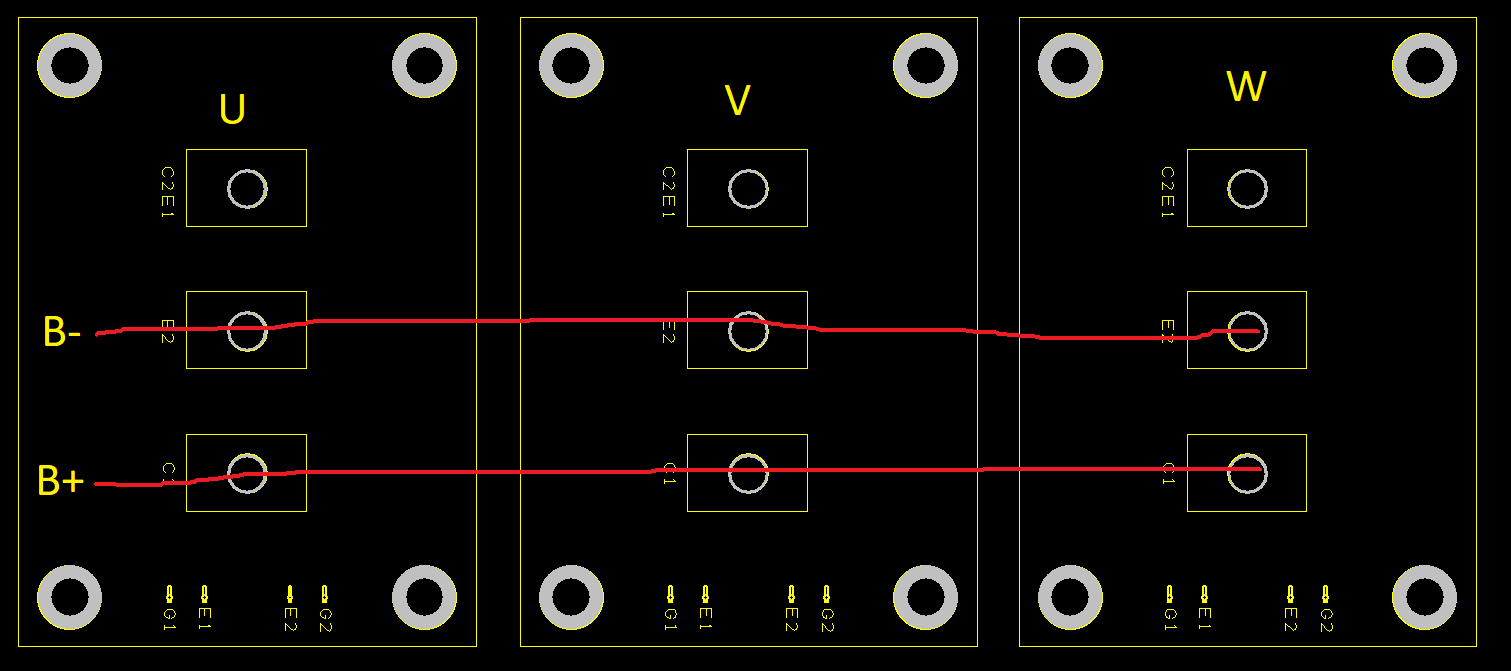
In the picture below, B+ goes to the bus capacitor plus, and also to the main contactor. The other side of the main contactor goes to a fuse. The other side of the fuse goes to the high voltage battery pack +. B- can go straight to the bus capacitor -, and also to the high voltage battery pack -. U, V, and W goes to U,V,W on the 3 phase motor. B+ also goes to one side of a precharge relay. The other side of the precharge relay goes to one side of a precharge resistor. The other side of the precharge resistor goes to the high voltage battery pack +. For setting the precharge time in realterm, you go like this:

precharge-time 47

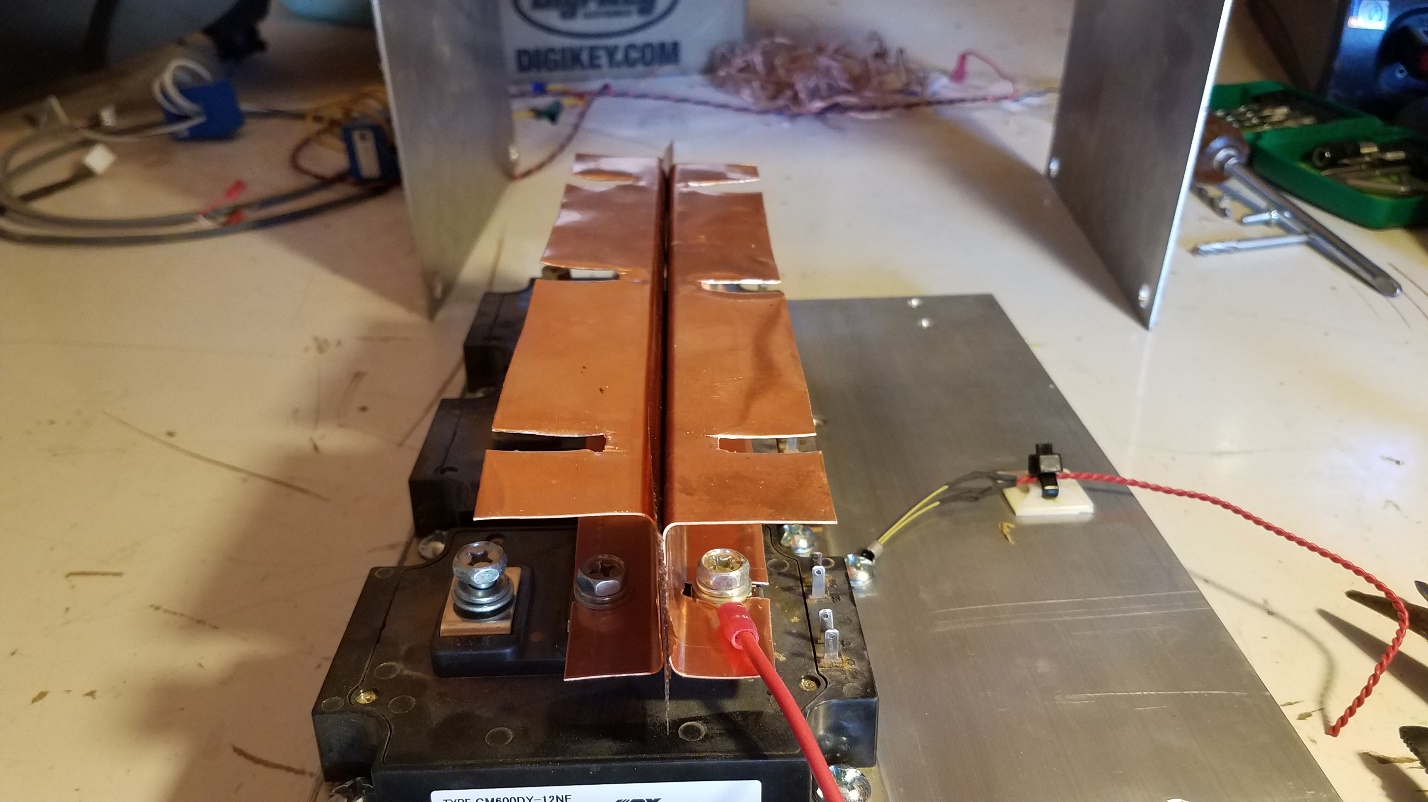
save

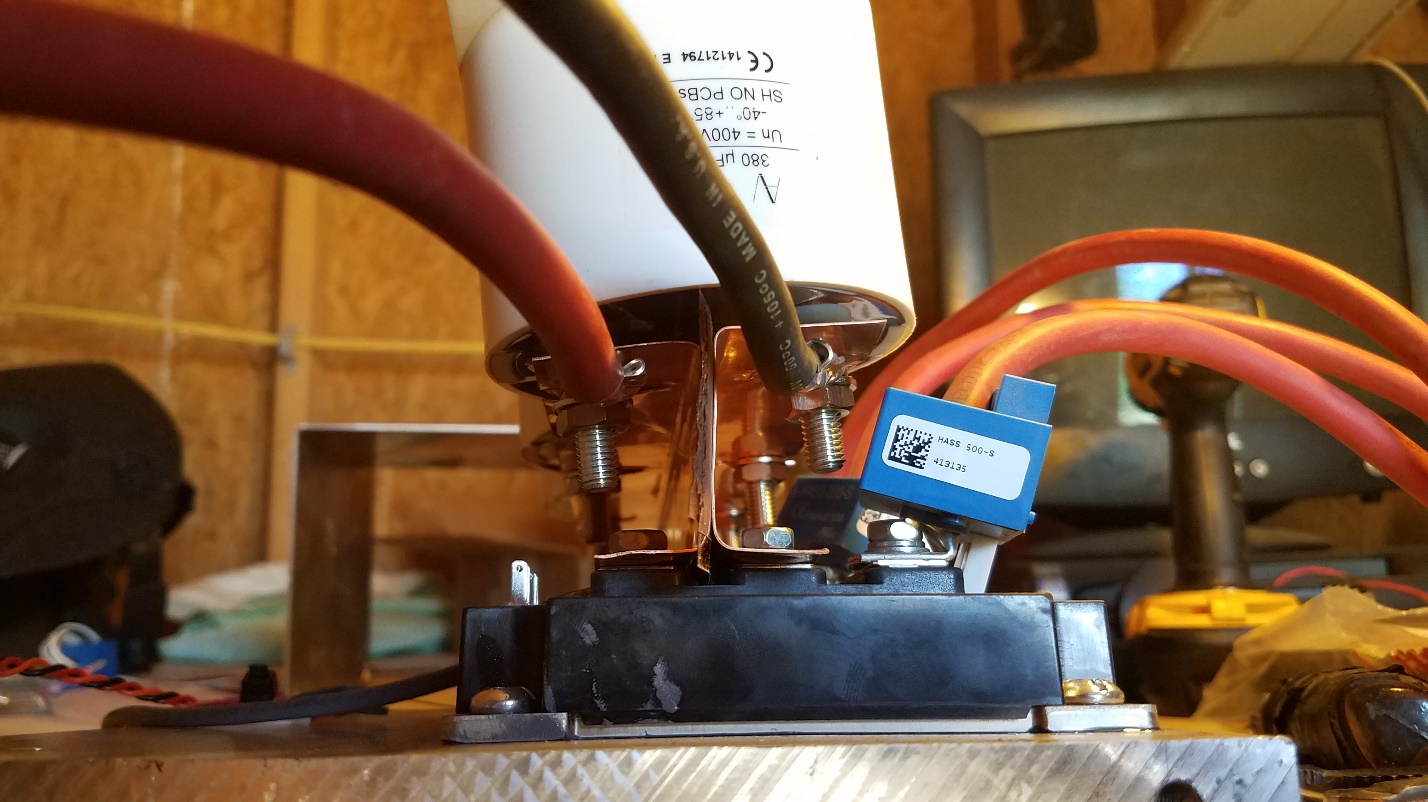
That would set the precharge time to 4.7 seconds. Let’s say you choose a precharge resistor that is 470 Ohms, and your bus capacitance is 1000 microFarad. For your precharge time, it would be:

5\*R\*C = 5 \* 500 \* 1000\*10^-6 = 2.5 seconds. So, you would set the precharge time to 2.5 seconds in that case.



This is one possible way of connecting the bus capacitor(s) to the IGBTs. You can do it however you want. All that matters is, the “loop area” from B+ on the IGBTs to the capacitor + to the capacitor – to the B- on the IGBTs is as small as possible. That’s why copper sheets are used, and why they are sandwiched together like that.





Notice that the current sensors go on each of the 3 phase cables (U/V/W) to the motor. **NO! YOU CANNOT USE 2 OF THEM WITHOUT MODIFYING THE CONTROL/DRIVER BOARD! 😊**



Notice in the above picture how the high voltage B+ connects to the 3 points on the control/driver board.

Programming:

Reprogramming the dsPIC30F4011 microcontroller using MPLab X:

If you want to change the flash memory of the microcontroller, do the following:

1. Download MPLab X. Click on File -> Import -> Hex/Elf… (Prebuilt) file
2. Choose your .hex file and select “Pickit 3” from the hardware tool section. Click Next…
3. Set the active project. Click finish.
4. Plug the pickit 3 into the control/driver board at the 6x1 pin vertical connector that just has 6 bare pins sticking out next to the microcontroller. Make sure that the white triangle on the pickit 3 matches up with the white dot on the control/driver board (pin 1). Also plug the pickit 3 into the computer.
5. Turn on the 24v power to the control/driver board.
6. Click on the icon that says “Make and Program Device Main Project” when you hover the mouse over it.

Here’s a helpful demonstration with pictures, and a video:

http://jeffeblack.com/writing-software-to-the-microprocessor-in-the-diy-leaf-control-board-from-ps/