P17221 Problem Solving Worksheet

Identifying and Selecting Problem

PSP Step 1

What is “the problem” you are trying to solve?

- Commands sent from the Arduino are not producing the expected results, i.e. the motor is not spinning.
- The motor driver (MA860H) is faulting on startup.

Analyzing the Problem

PSP Step 2

What are the possible causes of “the problem”.

- The driver could be receiving voltage outside of it’s operating range, either on the high end or on the low end.
- The driver could be receiving unstable voltage.
- The control signals could be faulty.
- The driver could be broken.

Check out the above possible causes and ask “why”.
Are there other possible causes to get to the “root cause”?

- The power supply could be broken rather than the driver.
- The Arduino’s could be sending faulty signals, which the driver may be unable to interpret.

What is the “root cause”?

- There is no further level of abstraction.

Generating Potential Solutions

PSP Step 3

Brainstorm Possible Solutions

- Debug the driver.
- Debug the power supply.

- Debug the Arduino.
- Purchase a new power supply.
- Purchase a new driver.

Selecting and Planning Solution

PSP Step 4

First we plan to debug possible issues. We suspect that the problem may be with the driver. During testing, we noted a slight burning smell. Upon detecting this, we powered down the system. Due to the proximity of the burning smell with the driver, we suspect that the driver is the problem unit in our subsystem.

The first step for debugging is to isolate the individual components. The first component that we will isolate is the power supply. We will power on the power supply and test to make sure the voltage is steady at 36V ± 10%. This will be tested on a multimeter and an oscilloscope to make voltage is at the appropriate level and stable. If no problems are detected, the power supply is likely not a problem.

The second component that we must debug is the driver. We will input 36V into the driver to make sure that the driver is still faulting on startup. If no further information can be found in the manual, we will have to take the driver apart. Based on smelling a component that was burning, we will visually inspect the board for signs of damaged components.

The last component that we will check is the output coming out of the Arduino. We want to make sure that the output matches the output that we expected. Extraneous signals have the potential to cause problems.

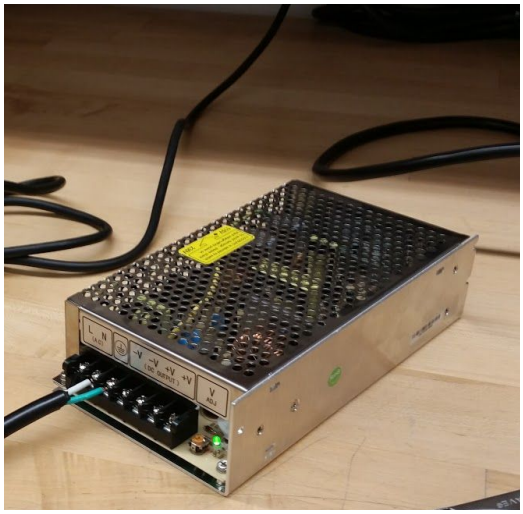
If all the components are working. We will reassemble the subsystem and make sure all of the components are wired together correctly. After powering on the subsystem, we will remeasure voltages and signals that were previously measured individually to make sure that the subsystem is working properly when assembled.

Of course, we will also consult Professor Slack to cover any gaps in our debugging, or if we are unable to determine the cause of the problem after debugging.

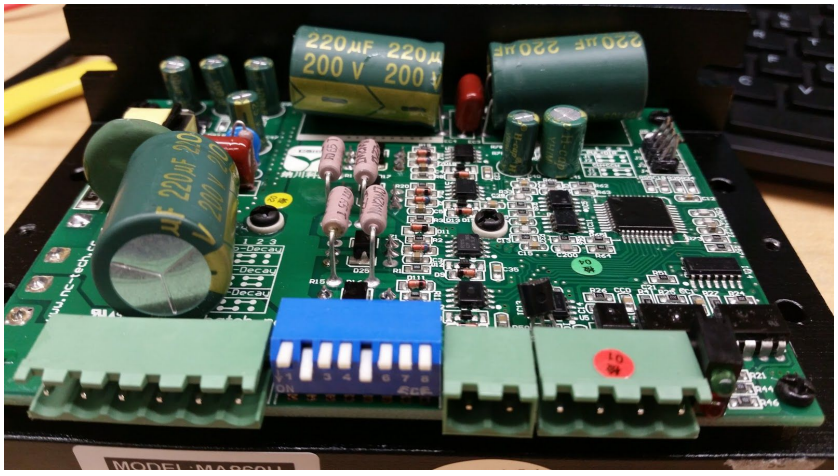
Implementing the Solution

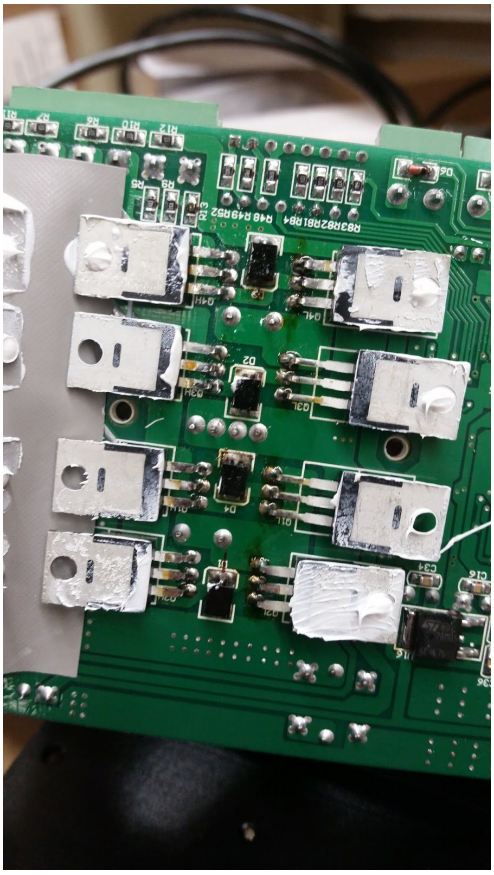
PSP Step 5

We followed our debugging plan fairly closely, checking out all of the individual components. The power supply checked out okay.



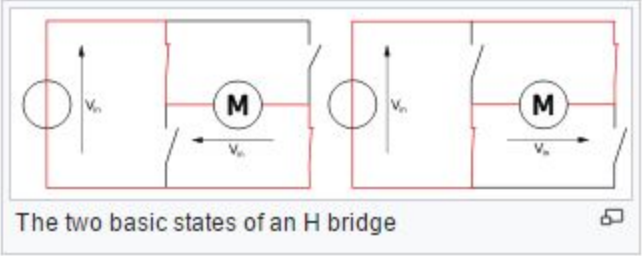
The driver was still faulting on startup. So, we took apart the driver. The top of the board appeared okay. The bottom of the board did not. There were some burn marks near the transistors that make up the H-bridge. It seems that the transistors were shorted.





The question now becomes, why were the transistors shorted? After looking at various sources online and documentation for the MA860H, it seems that the MA860H was expecting active low pulses while the Arduino was providing active high pulses. This is a problem.

Below is a diagram for a generic H-bridge driver.



The two basic states of an H bridge

Note that the driver works by toggling the transistors between open and closed in different combinations. There are certain combinations that are not valid. For example, if the two right-most transistors are both closed, the voltage (5V in this case) would be shorted through the transistors. If the driver expects active low pulses and the Arduino is providing active high pulses, there can be problems if there is not protection circuit. Evidently, there was no protection circuit on this driver. While the input did not go outside the bounds of the expected voltage, it

still caused the driver to close the two left-most transistors and/or the two right-most transistors at the same time, resulting in a 5V short through the transistors.

In terms of implementing a solution, we have two chief options. First, we could try to identify the burned out transistors are replace them on the board. The control signals from the Arduino can be adjusted such that the signals are active low. Secondly, we could purchase a new driver.

Personally, I lean towards purchasing a driver that uses active high control signals. There are multiple reasons for this. First, buying new transistors and soldering them on is costly in terms of time, and is not even guaranteed to work. Second, with the wiring we have, the 5V rail on the Arduino will come up before the control signals are initialized. This means that the driver could be getting a 5V reference with a control input that is floating. This could result is the same issue that we just debugged. With an active high style driver, the driver gets a ground reference with a floating control signal. This will not be a problem because the floating control signal will not register as high in the driver.

Evaluating the Solution

PSP Step 6

Purchasing a new driver of the same model seems to have worked. It seems to be clear now that the inputs are in fact differential. Thus, we can use active high control signals without fear of reprisal.