Debugging: How To

Notes:

Learning Objectives

- List three ***prerequisite*** skills need for becoming a strong hardware debugger.
- List three habits that will aid in building ***safe and intelligible*** circuits.
- Learn a procedure for ***efficient*** debugging.

Preliminaries

What is Debugging

Debugging (aka **troubleshooting**) is *the process of identifying and removing errors from computer hardware or software*. In the ECE110 laboratory, we construct prototype circuits on a breadboard that often do not initially perform as expected. The debugging process is a procedure that we can use to ultimately get to the root cause of the error so that it can be corrected.

This document is divided into three key sections on debugging. The first will describe the prerequisite knowledge for **becoming a strong hardware debugger**. As this is an introductory course in electrical engineering, we understand that forming this knowledge will remain a "work-in-progress" throughout the semester, but it is important for you to understand your own responsibility for due diligence in meeting the Learning Objectives from previous exercises and lectures. Attend office hours for more guidance as needed.

The second section on debugging will focus on three **safe and intelligible approaches to building circuits** that will a) make it less likely to commit common mistakes and b) make it easier to debug when mistakes do occur. Through this careful approach, you are also less likely to cause physical harm (perhaps small burns) to yourself or your teammates.

The third section gets into a specific **debugging procedure** for when a circuit is recognized to have an error. In that section, we examine general domain-level common problems, specific application-level common problems, and then a procedure for locating less-predictable hardware problems.

Notes:

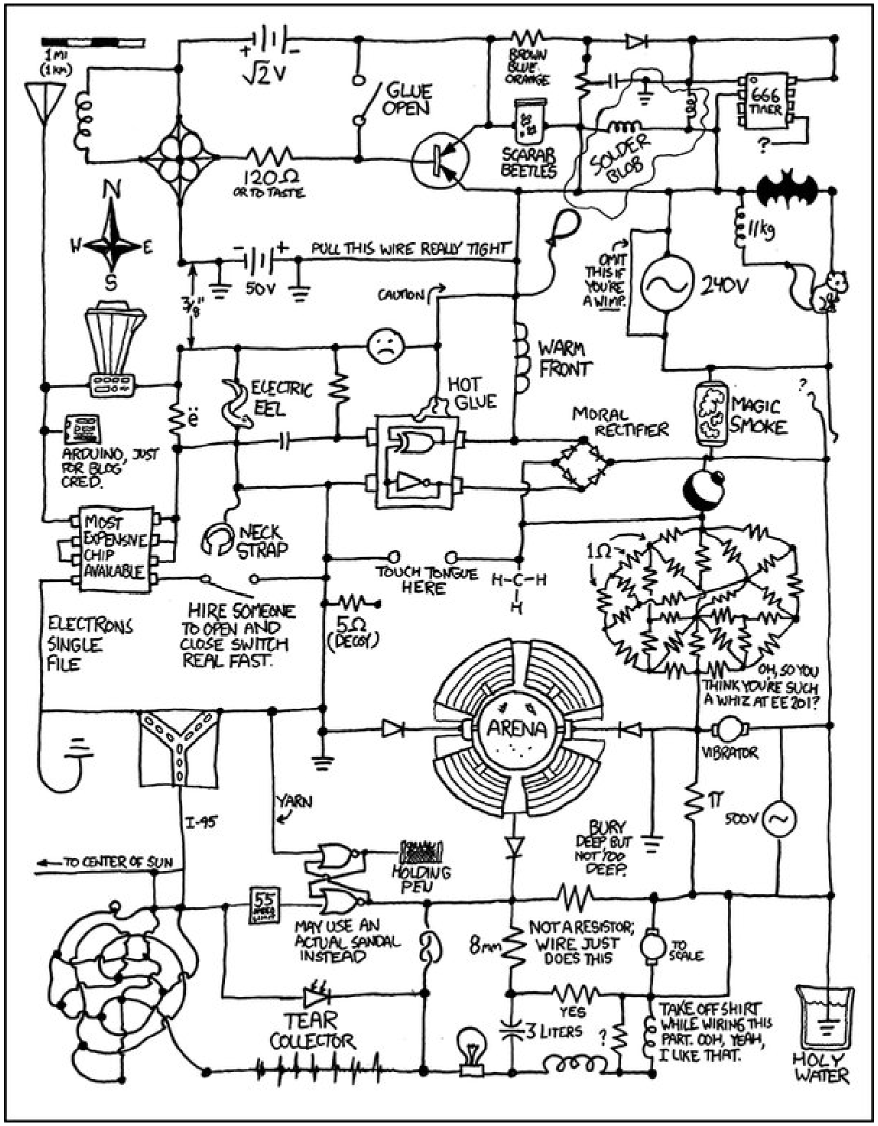


Figure 1: Debug this, my friends. (humor)

Notes:

Prerequisite Skills

To become an efficient hardware debugger, you should have enough prerequisite knowledge to correctly anticipate outputs. The use of a voltmeter or an oscilloscope, for example, to monitor the voltage at a particular node is only effective if you have a strong idea of what that node voltage should be and how to attach the meter. You should be familiar with fundamental circuit analysis and its terms, for example, node voltage is the voltage measured at the node of interest *relative to* a common (ground) node. The prerequisites for an efficient debugger are

1. Knowledge and application of **fundamental circuit analysis**.

   If you don't know what to expect, you are already in trouble. If you can't predict the errors that could cause your non-ideal observations, you cannot make the necessary corrections.

2. Knowledge of **circuit element models** and the limits of these models.

   The models (expected behavior) of your circuit devices are based on how you are expected to use them. If you incorrectly build your circuit, a circuit element may behave differently than that simplified model. Knowing something about device model limitations will assist in debugging.

3. Knowledge of **equipment** used in the lab as well as their behaviors/models **and their limitations**.

   Test-and-measurement equipment like the multimeter, voltage source, function generator, and oscilloscope also have physical characteristics (models) that can either be changed through the settings (like 50 Ohm and 1 MOhm options on the oscilloscope channel input) or may fall short of "ideal" in some settings (for example, a voltmeter has a large resistance but doesn't have infinite resistance, so this will affect the circuit when measuring voltages across devices with a resistance on the order of MOhms).

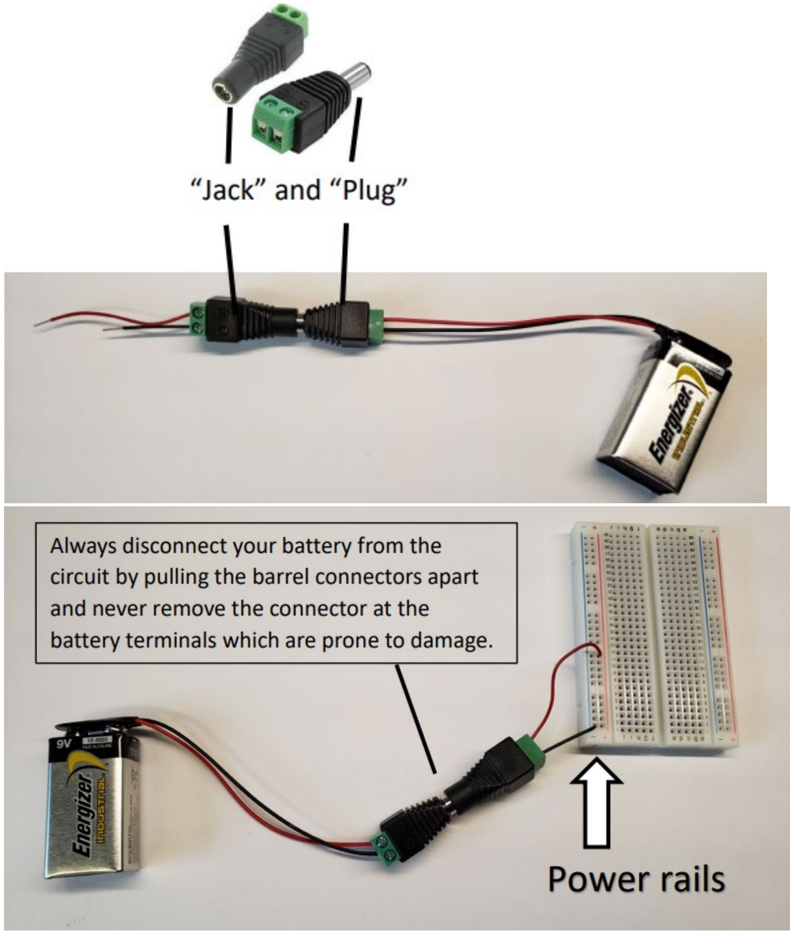
In ECE 110, all these are addressed in the learning objectives of earlier exercises or provided in the details of additional resources made available to the students. Again, since this is an introductory course, many exercises are completed in the physical laboratory where teaching assistants are available to assist you while you are still acquiring these prerequisite skills.

Safe and Intelligible Builds: *Cautious Construction PLUS Cautious Power-up*

Haste makes waste. It is easy to get in a hurry with the hope that everything will work on your first try. For circuits consisting of, say, three elements, this may be true. But as our circuits become more complex, the likelihood of error grows quickly, and your project will benefit from slowing down and validating your construction as you go. By conditioning yourself to follow certain habits, you will build more error-free circuits and circuits that are also much faster to debug if an error should exist.

1. Cautious Build
   a. **Plan the layout** on your breadboard before you start. How much room will you need? Do you want certain devices closer together than others? Do you want the physical build on your breadboard to strongly match the abstract draft of your circuit schematic?
   b. Let the **color** of your wires hold meaning. Use red only for connection to nodes that are directly connected to the positive side of your battery/power supply. Use black only for nodes connected directly to the negative side. Use, say, yellow wires for key nodes that have labeled voltages that you will want to observe later.
   c. Consider the use of small colored wires to mark other "**test points**" on a larger circuit so that you can easily locate them during a presentation or demo. Consider using a small piece of masking tape to put a written node name or other comment directly into your circuit (much like comment lines in a software code!).
   d. As you build each portion of your circuit, test it individually to ensure it is functioning properly *before* connecting it to the previous portion of the build. If you **test the smaller subcircuits one-by-one**, you will have a better chance of catching an error early.
2. Cautious Power-up. Protect the circuit and protect yourself.
   a. Before connecting the power supply to the power rails…**trace each circuit loop** of your circuit schematic on the physical build to validate correctness. Devices often suffer permanent damage when a mis-wired circuit is powered on.
   b. Always connect the **battery** as the **last step** before monitoring your circuit for correct behavior. Always disconnect your battery as the **first step** before debugging a problem. Damage may be avoidable. Have a voltmeter or oscilloscope in place to watch for signs of success or failure as you plug in the battery and be ready for a quick power-down!
3. Quick Power-down.
   a. Limiting the exposure of a mis-wired circuit to the power supply can also reduce changes of device damage or injury to the student. An overheated IC or resistor can splinter and send ceramic material flying into the air.
   b. If your circuit smells as if a component is burning, **disconnect the power source immediately**! Do **not** search for the burning element with your finger or you will get a burn! Wait for your circuit to cool and initiate your debugging procedure.
   c. Disconnecting at the barrel connector can help prevent burns to your fingers if your board has a power-dissipating short circuit!

Notes:



Always disconnect your battery from the circuit by pulling the barrel connectors apart and never remove the connector at the battery terminals which are prone to damage.

Figure 2: By making the barrel jack-and-plug the last thing you connect and the first thing you disconnect, you can be ready to protect your circuit from accidental damage.

Notes:

Notes:

Efficient Debugging

Efficient debugging suggests getting to the root cause of any failure quickly. We want to identify the location of the fault as quickly as possible to save valuable time. Finding the fault is like the game of *Twenty Questions* or Hasbro's *Guess Who?* game or Spin Master Games *Hedbanz*. In all these games, you want to find the answer in as few questions as possible. Therefore, you ask questions to eliminate possibilities while approaching the correct answer. Statistically, in our debugging procedure we should eliminate the most-common errors first. Do you know the most common circuit-construction errors? Don't worry, your instructors and teaching assistants do! We'll highlight a few here, starting with the most general circuit-domain specific errors:

1. Check your **Supply Power** and power rails
    a. Did you forget to attach the power to the breadboard or turn it on (if appropriate)? Oops. 😊
    b. Is your battery dead/dying? Disconnect it from the circuit to check the voltage using a voltmeter. Is it lower than expected? Connect it to your circuit and measure it "under load." Did the voltage fall drastically? If so, there are still two possibilities:
        i. your battery is weak and doesn't perform well "under load" or
        ii. your battery is fine, but your circuit is mis-wired and perhaps has a short circuit (a difficult load!).
2. Check for **short circuits**.

    Often a student may forget the layout of the breadboard and accidentally make connections to the same node. A student might also get confused between series and parallel and place a wire or instrument in the wrong configuration. Sometimes a student will connect the nMOS drain directly to power and the source to the ground node…without some amount of resistance from power-to-ground, the nMOS can generate a short (and get very hot)!

3. Check your instrument's **horizontal and/or vertical scales**.

    Are you using your oscilloscope (or other test-and-measurement tool) with the wrong time or amplitude scale? It's easy to just adjust the horizontal and vertical scale without thinking about what order of magnitude to expect for those signals. Often a student will see some voltage variation and assume it is correct without stopping to think that it should be on the order of, say, 5 volts and not 20 mV!

Additionally, within any specific experiment, there can be devices or instruments which are commonly misused. While these errors can vary significantly from one experiment to another, we can list a few of the most common issues here.

1. The LED is **burnt**.

If you were expecting an LED to light up and it doesn't, you may have inserted it backwards or damaged it. Often, damage is caused by the failure to use a current-limiting resistor in series with it. The damage may have occurred the last circuit in which you used the LED and investigation of the current circuit may not reveal that mistake. To test an LED, use a series resistance of perhaps 1000 Ohms and connect it briefly to your battery in its own circuit. If it is illuminated, you are good.

2. The IC is inserted **backwards and/or damaged**.

   It is easy to overlook the location of pin 1 and put your IC in the wrong direction. If this happens, you might have also reversed the "power-and-ground" pins. Unfortunately, this mistake can damage the IC beyond use. To determine if your IC is undamaged, you should move it onto a separate breadboard for independent testing (without the complexities of your possibly troubled circuit).

3. The **wrong** IC is inserted.

   We don't have many ICs in the kit, but several do look very similar. The safest way to know you have the correct IC is to read the label.

4. There is a **design** flaw.

   Yes, you cannot trust everything to be flawless. This ties back to knowing what to expect and having the analytical capability to detect and correct an error.

While we have given you several "common" errors, there are an infinite number of other mistakes that can be made. Is there a procedure to follow to isolate an error that hasn't yet been identified? Yes! Going back to the game of *Twenty Questions*, the most efficient way to win (locate the error in our case) is to *systematically cut the possible error locations in half*! This is often called Divide-and-Conquer.

Notes:

Notes:

Divide-and-Conquer

We can explain this divide-and-conquer approach best in an example.

The Circuit: Suppose the following circuit is intended to run the motor at full speed when a room is darkened and stop entirely when in light. The user finds that the motor runs in both a dark room and a bright room. The design also has a red LED indicator light intended to illuminate brightly when the motor turns off and turn off entirely when the motor is running.

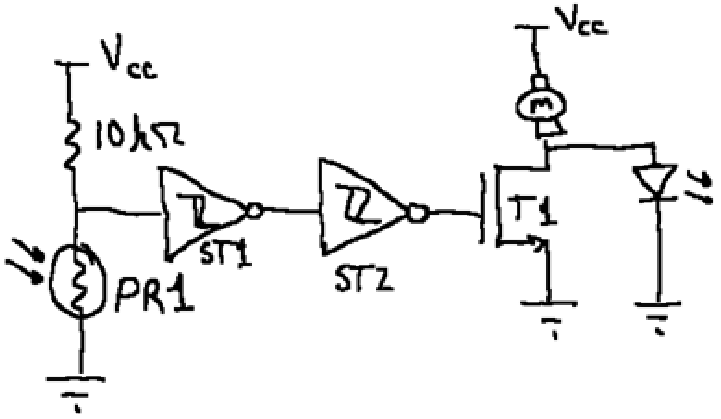


Figure 3: Example motor-and-LED control circuit.

1. Check the “output” with a knowledge of what to expect. If the output is as expected, run the system through some other parameters to make sure its overall functionality is also as expected so you can be sure it is fully functional.
2. If the output is not as expected, power the circuit down and consider the 6 common mistakes provided above and any other common mistakes as indicated by your teaching assistants. If none of these reconcile the problem(s), begin a divide-and-conquer approach.
3. Using the oscilloscope, probe all the power connections in your circuit (relative to ground). They should all be near the power supply voltage and without variation. Probe as close to the device as you can to eliminate the possibility of a poor contact point not being identified.

Notes:

4. Probe the key node at the “output” of your circuit. It is safe to assume it is not as expected or there would be no issue, but at least now, you know what it is. Now, probe a key node near the center of your circuit. If it doesn’t look correct, disconnect it from the “downstream” portion of your circuit to isolate it. If this results in a correct signal, your issue is to the right (downstream). If not, you can reconnect it and move to a node further to the left (upstream) and try again.
5. Watch for both errors in your build as well as design errors.

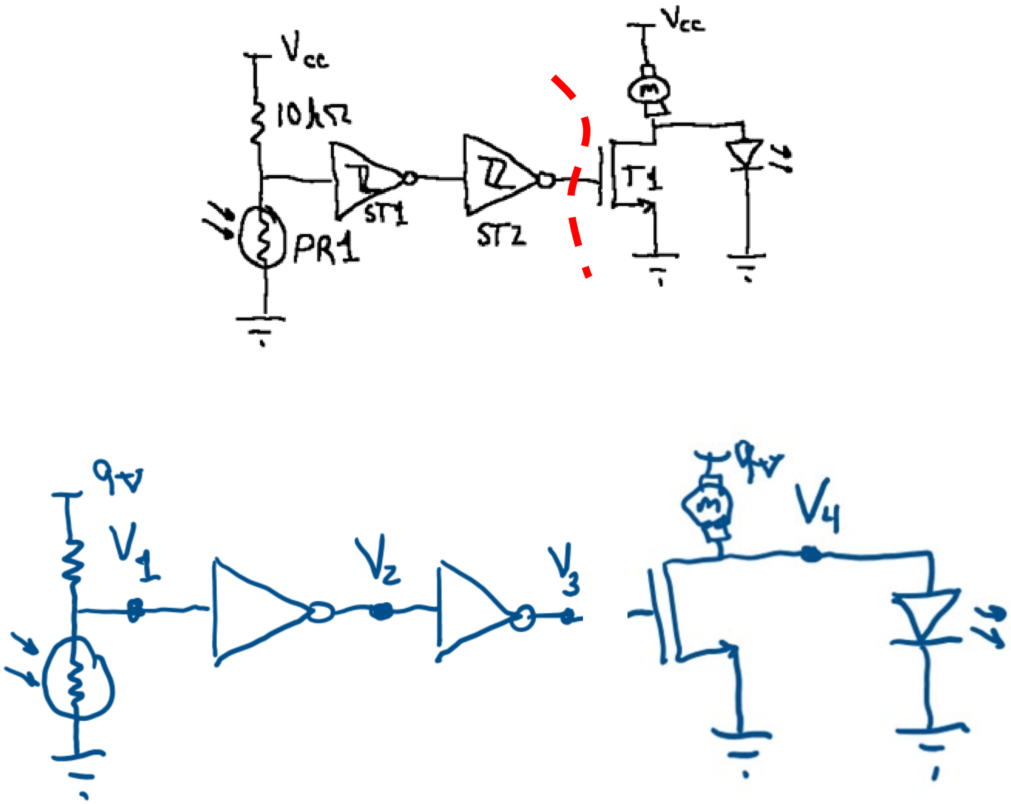


Figure 4: Divide and Conquer. If the circuit isn’t working, physically remove a connection between sub-circuits. Above, we disconnect V_3 from the MOSFET and see it the voltage V_3 behaves as expected.

Notes:

Review

In this exercise you have been presented with a three-tiered method to efficiently debug circuit hardware. Careful planning and layout reduce the chance of error and improve the ability to "see" the key test points of the circuit. Knowledge of the most-common issues provide an opportunity to eliminate the most probable errors first. Finally, a systematic divide-and-conquer breakdown of your circuit at certain test points (while removing and then re-attaching the downstream circuitry) can help you home in on the problematic portion(s) quickly.

While it is good practice to work to avoid circuit failures, you should also understand that failures are key to the learning experience. Each time you pause to examine your circuit or debug an issue, you will also be taking time to think through what you know and expect. The debugging process is a natural mechanism for making learning "sticky" so that you don't merely go through the motions of learning but retain the concepts in your mind long-term.

Question 1: **Generate a flowchart** that highlights a procedure for debugging. You do not need to include every detail presented in this document, but that flowchart should be functional enough that it would assist you or your lab partners while building and debugging a circuit.

NOTE: If you are unfamiliar with flowcharts, you may find benefit from this short tutorial at https://courses.engr.illinois.edu/ece110/fa2023/content/labs/Experiments/Skill_Flowcharting.pdf.