Troubleshooting Guide for Beginner Engineers

1. Ensure safety and prevent expensive damage while changing as little as possible.
2. Describe the problem
3. Divide and measure

1. Ensure safety and prevent expensive damage while changing as little as possible

The first responsibility of an engineer is the health and safety of themselves, the public, and personnel. We all want to solve problems and investigate, but we must consider safety and damage first.

Possible hazards you may encounter in your troubleshooting career include:

- Fires that must be extinguished before it is safe to enter the area
- Enclosed environments that may need to be ventilated to ensure it is safe to enter.
- Valves and circuit breakers that need to be locked to prevent others from using them while people work on the system.
- Equipment that should not be turned off without a careful plan in place (cooling systems, water supply systems, etc.)

Once it is safe to work on a system, work methodically. Consider the state you find the system in to be evidence, and document as much as possible before you change anything.

2. Describe the problem

People can have strong emotional reactions when equipment doesn't perform the way they want it to, but as an engineer you are paid to be the careful, methodical thinker in the room. Management may say things like "Try replacing part XYZ because that fixed it last time the machine broke" but depending on the symptoms, you could be facing a completely different problem than "last time".

"It doesn't work" is an almost useless problem description. Describe what you expect to observe, and what you actually observe.

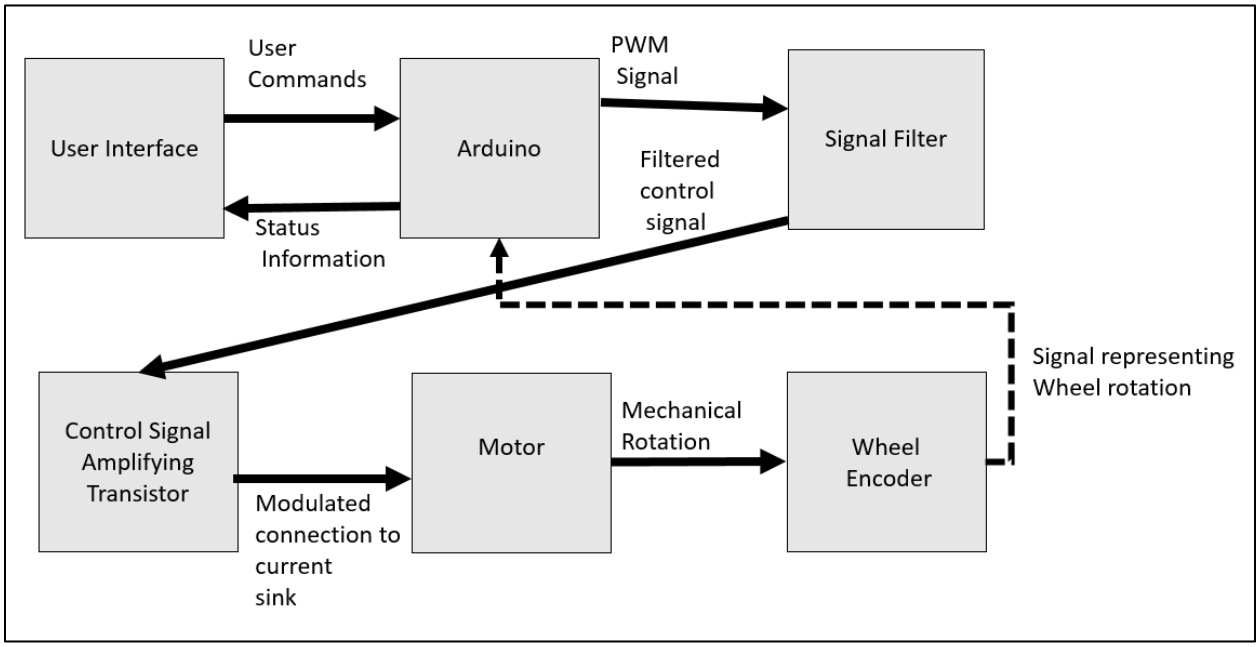
Bad Descriptions	**Good Descriptions**
"It doesn't work."	"We expect the LED to blink three times when we push the button, but it just stays constantly lit."
"We got too much water."	"When the system is running, this pump should be moving the water at a rate of 100 gallons per minute, but the flow sensor is measuring 250 gallons per minute."
"The transistor must be bad"	"We expected to see a signal change between 0V and 5 V but instead we see a signal between 0V and 2V."
"My computer hates me!"	"Smoke billows out from the computer casing when the computer is turned on, and there is

	an eerie screeching coming from the hard drive. Neither the smoke nor the screeching are expected conditions."
"The oscilloscope is broken"	"The oscilloscope turns on, but shows a flat line even when we attach the probes to a part of the circuit that we expect to see a sinusoidal voltage signal at."

3. Divide and measure

The systems you will work on in your career will not be monolithic, single-component "things." They will be made up of multiple components interacting with each other. A key part of troubleshooting is identifying what parts of the system are known *not* to cause the problem, then focus your attention on what remains.

For example, consider the system below:

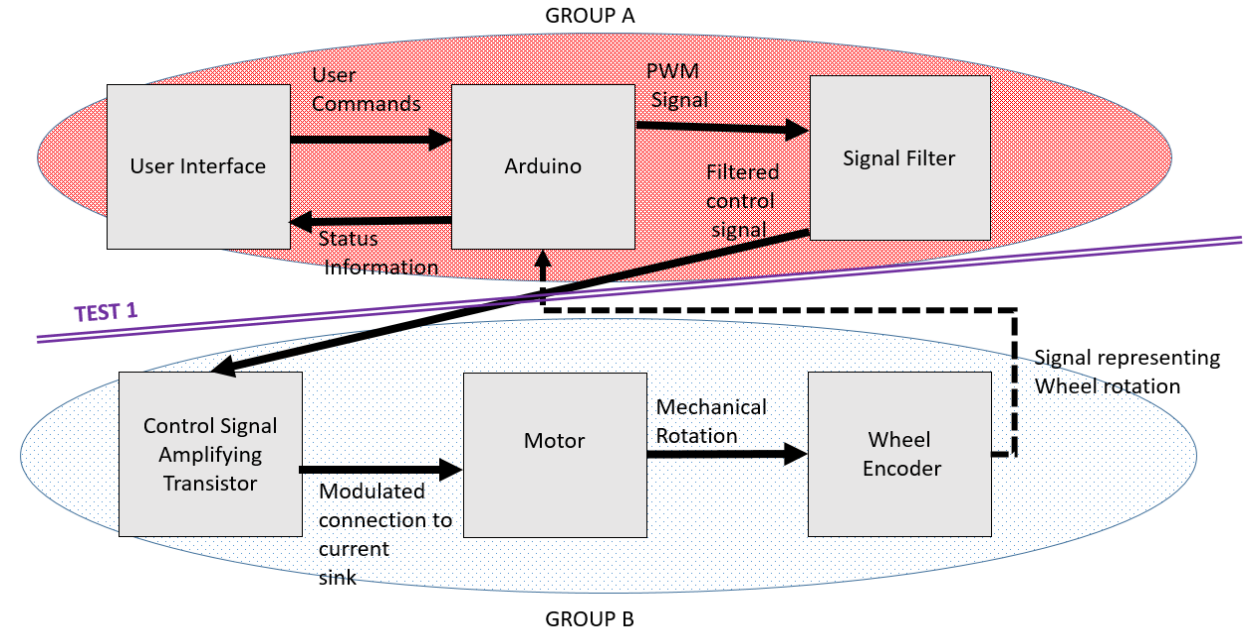


We have described a system as a collection of component circuits, and each component circuit has inputs and outputs. Suppose we have a problem that is described as:

PROBLEM STATEMENT 1: ***"The motor does not spin even though the user interface specifies it should spin at 2,000 RPM."***

How could we troubleshoot this? Well, to start, we will divide the circuit into two pieces. Pick a convenient point to test. In this case, we will ask

Test 1: ***"Does the filtered control signal look like we expect it to when the user interface specifies the motor should spin at 2,000 RPM?"***



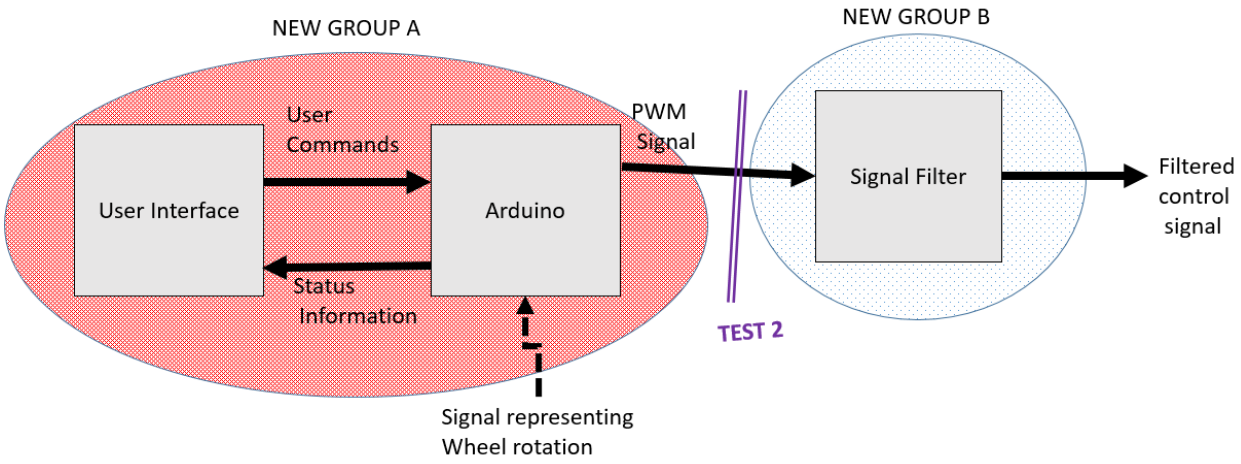
Let us call the User Interface, Arduino, and Signal Filter "Group A." We will also call the Transistor, Motor, and Wheel Encoder "Group B." We check the signal going from Group A to Group B – in this case, this means we use the user interface to request a speed of 2,000 RPM and use our oscilloscope to measure the filtered control signal.

If Test 1 finds that the signal is the expected value, then we continue to troubleshoot Group B, because Group A is working as expected.

If Test 1 finds that the signal is NOT the expected value, then we know there is no reason to look at Group B any longer, because Group A is not working as expected, and we cannot expect Group B to work properly until we give it a valid input.

In this example, let us suppose we measure a bad filtered control signal. Specifically, imagine that we measure a constant 0V signal. What next?

DIVIDE AND MEASURE!



Test 2: ***"Does the PWM signal look like we expect it to when the user interface specifies the motor should spin at 2,000 RPM?"***

We give the user interface a command to request the speed to be set to 2,000 RPM, and we measure the PWM signal from the Arduino to the signal filter. Notice that a new division "Test 2" has been drawn on the diagram and that "Group B" has been removed from consideration.

If the PWM signal is what we expect it to be in this situation, then the problem must be to the *right* of the TEST 2 division. If the PWM signal is NOT what we expect it to be, then the problem must be to the *left* of the TEST 2 division.

For this example, let us suppose that we measure the PWM signal and find that it is exactly what we expect it to be. This means that we do not bother troubleshooting the user interface, and we do not bother troubleshooting our code. We already know that, given an expected input, the "Signal Filter" block gives an unexpected output, so there MUST be at least one problem in the Signal Filter block.

This does NOT mean that there are no other problems in the system. However, you should not change anything else in the system until you fix the section you KNOW is bad.

So, for this example, let us assume you found out that you wired your capacitor incorrectly in your signal filter. You correct the problem, and the motor starts spinning at the correct speed. Congratulations! You have fixed a problem in a complex system with six components and seven interconnections, and you only had to take two measurements and examine one component to do it!

Meanwhile, another group who has the same problem you have, but has not read this guide, has replaced their motor once, their transistor three times, and is currently digging through their Arduino code looking for a bug that does not exist. This is the power of thoughtful, methodical troubleshooting.