Modular Electronics Learning (ModEL) PROJECT



TROUBLESHOOTING AND PROTOTYPING TIPS

© 2025 by Tony R. Kuphaldt – under the terms and conditions of the Creative Commons Attribution 4.0 International Public License

LAST UPDATE = 17 April 2025

This is a copyrighted work, but licensed under the Creative Commons Attribution 4.0 International Public License. A copy of this license is found in the last Appendix of this document. Alternatively, you may visit http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons: 171 Second Street, Suite 300, San Francisco, California, 94105, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

ii

Contents

1	Intro	oduction	3
2	Gene	eral circuit prototyping	5
3	Sold	erless breadboard caveats	7
4	Mult 4.1 4.2	General multimeter adviceA gallery of multimeter "tricks"4.2.1 Recording unattended measurements4.2.2 Avoiding "phantom" voltage readings4.2.3 Non-contact AC voltage detection4.2.4 Detecting AC power harmonics4.2.5 Identifying noise in DC signal paths4.2.6 Generating test signals4.2.7 Using the meter as a temporary jumper	 9 10 11 12 13 16 17 18 19 20
5	Linux 5.1 5.2 5.3 5.4 5.5 5.6	x computer operating system Linux utilities for command-line navigation Linux utilities for general file viewing and editing 5.2.1 Linux text editors 5.2.2 Linux text-viewing utilities Linux command-line plumbing Linux utilities for networking Linux utilities for programming Linux utilities for PostScript document files	 21 22 23 23 24 25 25 25
6	Digit 6.1 6.2 6.3 6.4	tal radio modules and software Software-defined radio receiver recommendations	27 27 29 31 33

CONTENTS

7	Programmable test equipment7.1Siglent SPD1305X power supply programming tips7.2NanoVNA vector network analyzer tips7.3Mcumall GQ-4x4 EPROM programmer tips	35 35 36 37		
8	??? 8.1 ??? tips 8.2 ??? tips 8.3 ??? tips 8.4 ??? tips	39 39 39 39 39		
A	Problem-Solving Strategies			
В	Instructional philosophy			
С	Tools used			
D	O Creative Commons License			
Е	References	61		
F	Version history	63		
Ine	ndex			

1

CONTENTS

Introduction

Developing new circuits is a process often fraught with challenges. This is especially true if certain components or sub-systems in your design are poorly documented or have known design flaws of their own. This document is a repository for general troubleshooting in electronic circuit development as well as a collection of tips and useful information on specific pieces of software and hardware. Expect this document to grow substantially over time!

General circuit prototyping

- Build and test complex systems in stages! Building the entirety of a complex system and expecting it to function properly on the first try is folly, due to the high likelihood of multiple problems. Instead, identify portions of your circuit or system that may be constructed and tested individually, then build and test them one-by-one before connecting them together to form a larger system. This strategy keeps the "problem space" as small as possible so it's easier to locate and remedy problems, whether they be faults or design flaws. Breaking apart a malfunctioning complex system and testing the separated sections works well for the same reason.
- When in doubt, collect more data! If you find yourself confused on why a circuit is not doing what you think it should, *take more measurements!* Test equipment will show you what your eyes cannot see, and tests done with different types of instruments (e.g. oscilloscope instead of multimeter) will show you things those other instruments might not reveal.
- Ensure stable source voltages Most electronic circuits absolutely require stable DC power supply voltages to function properly. The DC power supply "rails" should ideally exhibit unchanging voltage values well within the high and low allowable values for the circuit(s) being powered, sagging negligibly when loaded, and having little or no "noise" (e.g. ripple, etc.). Checking for proper power supply voltages using a voltmeter is always a good first step when troubleshooting a misbehaving circuit. Checking those same voltages using an *oscilloscope* will show transients that a multimeter will miss.
- Always decouple your supply rails Capacitors connected in parallel with the DC voltage source(s) act to stabilize voltage (see above) during moments of transient current demand. It's difficult to have too much of this "decoupling" capacitance, but it's also important that this capacitance have minimal equivalent series resistance (ESR) so as to not impede current when it needs to quickly source or sink it. Minimum decoupling capacitance will be based on the magnitude of expected transient current (I), the maximum tolerable voltage sag (dV), and the expected duration of the transient current (dt) in accordance with $I = C \frac{dV}{dt}$. NP0 ceramic capacitors are excellent for low ESR, while aluminum electrolytic and tantalum capacitors are terrible.

- Never let CMOS logic inputs float CMOS-based digital logic IC inputs must always have a definite connection to "high" (+V) or "low" (Ground). If left unconnected (i.e. "floating"), inputs will be susceptible to stray electric fields and may assume *any* logic state at random.
- Never let comparator or opamp inputs float Comparators and operational amplifier ICs also have high-resistance inputs and should not be "floated" for the same reason.
- Wire routing and placement matters at high frequency Parasitic capacitance and inductance values are negligible in DC and low-frequency AC circuits, but at high frequencies they can become problematic. This is how signals "couple" from one conductor to another. Keep "aggressor" conductors away from "victim" conductors, minimize wire lengths, etc.
- **Pulse integrity** A well-formed digital pulse signal must have steep rising and falling edges, with valid logic voltage levels between those edge events. Many circuits using flip-flops will behave strangely if clock pulses rise or fall too slowly, for example binary counter ICs generating incorrect count sequences!
- **Pulse timing** "Synchronous" digital circuits using clock pulses to coordinate actions between components require input signal voltages to be in stable states both before and after each clock pulse arrives in order to reliably read those input signals. This is known as *set-up time* and *hold time*, respectively.

Solderless breadboard caveats

Solderless breadboards are useful tools when prototyping circuits using through-hole components, but they have several important limitations every student of electronics needs to know.

- Voltage and current limitations Breadboards are not intended for high-power circuits. As such, both their voltage and current ratings are quite modest. A high-quality breaboard model at the time of this writing is the model BB830 by BusBoard Prototype Systems, and its datasheet states a maximum voltage of 36 Volts and a maximum current of 2 Amperes.
- Connection resistance Not only does each wire-to-board connection have substantial resistance (often many tenths of an Ohm), but this resistance varies considerably if the wire or component lead happens to move at all. This is a major source of trouble when your circuit design demands precision resistance values, and/or low resistance (e.g. less than 10 Ohms) values! In such cases it is better to use a terminal block or even solder components together rather than make those connections on a solderless breadboard.
- **Power rails** Many breadboards offer "power rail" connections spanning the length of the board, making it convenient to connect DC voltage source terminals at one end of the board and accessing those rails at any point along the length using short jumper wires. *However, not all breadboard power rails work the same!!* Some models, notably the legacy "Radio Shack" and "Archer" breadboards manufactured by Tandy, have *divided* power rails which only run half the length of the board, and are not clearly marked as such.
- Use the right wire Solid wire only, not stranded, of an appropriate AWG gauge number (22 AWG is standard). Wires that are too thin won't made good contact with the board's internal spring clips, and wires that are too thick may not fit in and may even damage the spring clip if forced!
- **Backing surface** Most breadboards require a solid backing surface when used, to provide physical support for the spring clips when wires are pushed through the connection holes. A tabletop works well, as does the metal backing plate usually provided with a new breadboard.

Holding a breadboard in mid-air when inserting wires into it is a bad idea, as is adhering it to a soft surface with no hard sheet or plate in between the board and that soft surface.

• Not all breadboards are made the same! – Quality varies widely in breadboard design and construction. For example, some breadboards have unreasonably high insertion forces, others will not accept square post terminals, etc.

Multimeter tips

4.1 General multimeter advice

- Meters are loads Voltmeters have high input resistance, but they are not perfect "opens". Ammeters have low input resistance, but they are not perfect "shorts". All meters, to some extent, place an energy burden on the circuits they monitor.
- Ammeter fusing Quality multimeters always provide a fuse in-line with the red test lead which blows if ever excessive current passes. These fuses have no effect on voltage measurement, and cannot protect against excessive voltage!
- Sign indicates polarity Whenever a meter indicates a positive quantity it means the red test lead is at a higher electrical potential than the black test lead. When measuring voltage this means red = + and black = -; when measuring current it means conventional flow into the red and out of the black.
- Resistance versus diode-check modes Most digital multimeters offer different modes for sensing continuity, and these usually include *resistance* and *diode check*. Both modes generally work by sourcing a current (conventional flow exiting the red lead and entering the black) and sensing voltage. In the case of resistance-measuring mode the meter outputs a constant current and infers resistance by the amount of voltage dropped, the amount of current sourced depending on the range of the resistance measurement. In the case of diode-check mode the meter outputs a constant current and directly registers the voltage dropped by the PN junction. One important distinction between these two modes is compliance voltage (i.e. the maximum voltage output by the meter as it attempts to drive a constant current): in resistance-measuring mode the meter limits its compliance voltage to a level low enough that a PN junction should not be forward-biased, in order to avoid false measurements if a PN junction happens to be connected in parallel with the resistance you intend to measure.
- AC/DC discrimination Modern digital multimeters generally do an excellent job of discriminating between AC and DC quantities in those modes. For example, a digital voltmeter set to DC mode will register a DC power supply's voltage while ignoring ripple, while the same multimeter set to AC mode will register the ripple voltage while ignoring the DC voltage level. This is an excellent feature to use when one does not have an oscilloscope to check for ripple on DC power supply rails.
- Min/Max mode Arguably one of the most important features of a digital multimeter, the ability to record the minimum and maximum quantities since engaging the Min/Max feature is extremely helpful for troubleshooting intermittent problems. Note that Min/Max mode should be engaged only *after* the signal is acquired by the meter, or else the lack of measurement (zero) will be recorded as Minimum. Also, Min/Max mode usually locks in the meter's range, if the meter is auto-ranging.
- Auto-ranging Speaking of auto-ranging, this feature is where the meter chooses its own range value in acquiring a new signal, in order to give the user the best resolution. However, this feature can also become annoying if the signal in question rises and falls enough to

4.2. A GALLERY OF MULTIMETER "TRICKS"

provoke repeated "range-changes" on the meter's part, much like an automobile's automatic transmission refusing to remain in one gear despite a constant load. Auto-ranging may always be overridden by the user!

- Sample rate Every digital measuring device works by periodically sampling the signal rather than continuously reporting it as an analog instrument does, and the rate at which this sampling occurs is important to know when the signal in question varies rapidly. Learn the sample rate of your meter, knowing that the bare-minimum rate must be *twice the frequency* of the signal of interest in order for any aspect of that signal to be reliably captured. Some digital meters offer a "Fast Min/Max" mode where the sample rate is higher than normal in order to better capture transients, usually gaining speed by sacrificing resolution.
- **Resolution** Speaking of resolution, some digital multimeters offer a "precision" resolution mode displaying extra digit(s) for the user. This high-resolution mode works by sacrificing sample rate, displaying the sampled quantity at greater intervals.

4.2 A gallery of multimeter "tricks"

The digital multimeter (DMM) is quite possibly the most useful tool in the electronic technician's collection¹. This one piece of test equipment, properly wielded, yields valuable insight into the status and operation of many electrical and electronic systems. Not only is a good-quality multimeter capable of precisely indicating electrical voltage, current, and resistance, but it is also useful for more advanced tests. The subject of this section is how to use a digital multimeter for some of these advanced tests².

For all these tests, I suggest the use of a top-quality field multimeter. I am personally a great fan of *Fluke* brand meters, having used this particular brand for nearly my whole professional career. The ability of these multimeters to accurately measure true RMS amplitude, discriminate between AC and DC signals, measure AC signals over a wide frequency range, and survive abuse both mechanical and electrical, is outstanding.

¹As a child, I often watched episodes of the American science-fiction television show *Star Trek*, in which the characters made frequent use of a diagnostic tool called a *tricorder*. Week after week the protagonists of this show would avoid trouble and solve problems using this nifty device. The *sonic screwdriver* was a similar tool in the British science-fiction television show *Doctor Who*. Little did I realize while growing up that my career would make just as frequent use of another diagnostic tool: the electrical multimeter.

 $^{^{2}}$ I honestly considered naming this section "Stupid Multimeter Tricks," but changed my mind when I realized how confusing this could be for some of my readers not familiar with colloquial American English.

4.2.1 Recording unattended measurements

Many modern multimeters have a feature that records the highest and lowest measurements sensed during the duration of a test. On Fluke brand multimeters, this is called the *Min/Max* function. This feature is extremely useful when diagnosing intermittent problems, where the relevant voltages or currents indicating or causing the problem are not persistent, but rather come and go. Many times I have used this feature to monitor a signal with an intermittent "glitch," while I attended to other tasks.

The most basic high-low capture function on a multimeter only tells you what the highest and lowest measured readings were during the test interval (and that only within the meter's scan time – it is possible for a very brief transient signal to go undetected by the meter if its duration is less than the meter's scan time). More advanced multimeters actually log the *time* when an event occurs, which is obviously a more useful feature. If your tool budget can support a digital multimeter with "logging" capability, spend the extra money and take the time to learn how this feature works!

4.2.2 Avoiding "phantom" voltage readings

My first "trick" is not a feature of a high-quality DMM so much as it is a solution to a common problem *caused* by the use of a high-quality DMM. Most digital multimeters exhibit very high input impedance in their voltage-measuring modes. This is commendable, as an ideal voltmeter should have infinite input impedance (so as to not "load" the voltage signal it measures). However, in industrial applications, this high input impedance may cause the meter to register the presence of voltage where none should rightfully appear.

Consider the case of testing for the absence of AC voltage on an isolated power conductor that happens to lie near other (energized) AC power conductors within a long run of conduit:



With the power switch feeding wire 5 in the open state, there should be no AC voltage measured between wire 5 and neutral (L2), yet the voltmeter registers slightly over 10 volts AC. This "phantom voltage" is due to capacitive coupling between wire 5 and wire 8 (still energized) throughout the length of their mutual paths within the conduit.

Such phantom voltages may be very misleading if the technician encounters them while troubleshooting a faulty electrical system. Phantom voltages give the impression of connection (or at least high-resistance connection) where no continuity actually exists. The example shown, where the phantom voltage is 10.3 volts compared to the source voltage value of 120 volts, is actually quite modest. With increased stray capacitance between the conductors (longer wire runs in close proximity, and/or more than one energized "neighboring" wire), phantom voltage magnitude begins to approach that of the source voltage³.

The equivalent circuit is shown here, with the DMM modeled as a 10 M Ω resistance:



An analog voltmeter would never have registered 10.3 volts under the same conditions, due to its substantially lower input impedance. Thus, "phantom voltage" readings are a product of modern test equipment more than anything else.

14

 $^{^{3}\}mathrm{I}$ have personally measured "phantom" voltages in excess of 100 volts AC, in systems where the source voltage was 120 volts AC.

4.2. A GALLERY OF MULTIMETER "TRICKS"

The obvious solution to this problem is to use a different voltmeter – one with a much lesser input impedance. But what is a technician to do if their only voltmeter is a high-impedance DMM? Connect a modest resistance in parallel with the meter input terminals, of course! Fluke happens to market just this type of accessory⁴, the SV225 "Stray Voltage Adapter" for the purpose of eliminating stray voltage readings on a high-impedance DMM:



With the voltmeter's input impedance artificially decreased by the application of this accessory, the capacitive coupling is insufficient to produce any substantial voltage dropped across the voltmeter's input terminals, thus eliminating. The technician may now proceed to test for the presence of AC control signal (or power) voltages with confidence.

⁴Before there was such an accessory available, I used a 20 k Ω high-power resistor network connected in parallel with my DMM's input terminals, which I fabricated myself. It was ugly and cumbersome, but it worked well. When I made this, I took great care in selecting resistors with power ratings high enough that accidental contact with a truly "live" AC power source (up to 600 volts) would not cause damage to them. A pre-manufactured device such as the Fluke SV225, however, is a much better option.

4.2.3 Non-contact AC voltage detection

While the last multimeter "trick" was the elimination of a parasitic effect, this trick is the exploitation of that same effect: "phantom voltage" readings obtained through capacitive coupling of a highimpedance voltmeter to a conductor energized with AC voltage (with respect to ground). You may use a high-impedance AC voltmeter to perform qualitative measurements of ground-referenced AC power voltage by setting the meter to the most sensitive AC range possible, grounding one test lead, and simply touching the other test lead to the insulation of the conductor under test. The presence of voltage (usually in the range of millivolts AC) upon close proximity to the energized conductor will indicate the energization of that conductor.

This trick is useful for determining whether or not particular AC power or control wires are energized in a location where the only access you have to those wires is their insulating sheaths. An example of where you might encounter this situation is where you have removed the cover from a conduit elbow or other fitting to gain access to a wire bundle, and you find those wires labeled for easy identification, but the wires do not terminate to any exposed metal terminals for you to contact with your multimeter's probe tips. In this case, you may firmly connect one probe to the metal conduit fitting body, while individually touching the other probe tip to the desired conductors (one at a time), watching the meter's indication in AC millivolts.

Several significant caveats limit the utility of this "trick:"

- The impossibility of quantitative measurement
- The potential for "false negative" readings (failure to detect a voltage that is present)
- The potential for "false positive" readings (detection of a "phantom voltage" from an adjacent conductor)
- The exclusive applicability to AC voltages of significant magnitude (≥ 100 VAC)

Being a qualitative test only, the millivoltage indication displayed by the high-impedance voltmeter tells you nothing about the actual magnitude of AC voltage between the conductor and ground. Although the meter's input impedance is quite constant, the parasitic capacitance formed by the surface area of the test probe tip and the thickness (and dielectric constant) of the conductor insulation is quite variable. However, in conditions where the validity of the measurement may be established (e.g. cases where you can touch the probe tip to a conductor known to be energized in order to establish a "baseline" millivoltage signal), the technique is useful for quickly checking the energization status of conductors where ohmic (metal-to-metal) contact is impossible.

For the same reason of wildly variable parasitic capacitance, this technique should *never* be used to establish the de-energization of a conductor for safety purposes. The only time you should trust a voltmeter's non-indication of line voltage is when that same meter is validated against a known source of similar voltage in close proximity, and when the test is performed with direct metal-tometal (probe tip to wire) contact. A non-indicating voltmeter *may* indicate the absence of dangerous voltage, or it may indicate an insensitive meter.

4.2.4 Detecting AC power harmonics

The presence of *harmonic* voltages⁵ in an AC power system may cause many elusive problems. Power-quality instruments exist for the purpose of measuring harmonic content in a power system, but a surprisingly good qualitative check for harmonics may be performed using a multimeter with a frequency-measuring function.

Setting a multimeter to read AC voltage (or AC current, if that is the quantity of interest) and then activating the "frequency" measurement function should produce a measurement of exactly 60.0 Hz in a properly functioning power system (50.0 Hz in Europe and some other parts of the world). The only way the meter should ever read anything significantly different from the base frequency is if there is significant harmonic content in the circuit. For example, if you set your multimeter to read frequency of AC voltage, then obtained a measurement of 60 Hz that intermittently jumped up to some higher value (say 78 Hz) and then back down to 60 Hz, it would suggest your meter was detecting harmonic voltages of sufficient amplitude to make it difficult for your meter to "lock on" to the fundamental frequency.

It is very important to note that this is a crude test of power system harmonics, and that measurements of "solid" base frequency do not guarantee the absence of harmonics. Certainly, if your multimeter produces unstable readings when set to measure frequency, it suggests the presence of strong harmonics in the circuit. However, the absence of such instability does not necessarily mean the circuit is free of harmonics. In other words, a stable reading for frequency is *inconclusive*: the circuit might be harmonic-free, or the harmonics may be weak enough that your multimeter ignores them and only displays the fundamental circuit frequency.

 $^{{}^{5}}$ These are AC voltages having frequencies that are integer-multiples of the fundamental powerline frequency. In the United States, where 60 Hz is standard, harmonic frequencies would be whole-number multiples of 60: 120 Hz, 180 Hz, 240 Hz, 300 Hz, etc.

4.2.5 Identifying noise in DC signal paths

An aggravating source of trouble in analog electronic circuits is the presence of AC "noise" voltage superimposed on DC signals. Such "noise" is immediately evident when the signal is displayed on an oscilloscope screen, but how many technicians carry a portable oscilloscope with them for troubleshooting?

A high-quality multimeter exhibiting good discrimination between AC and DC voltage measurement is very useful as a qualitative noise-detection instrument. Setting the multimeter to read AC voltage, and connecting it to an signal source where pure (unchanging) DC voltage is expected, should yield a reading of nearly zero millivolts. If noise is superimposed on this DC signal, it will reveal itself as an AC voltage, which your meter will display.

Not only is the AC voltage capability of a high-quality (discriminating) multimeter useful in detecting the presence of "noise" voltage superimposed on analog DC signals, it may also give clues as to the source of the noise. By activating the frequency-measuring function of the multimeter while measuring AC voltage (or AC millivoltage), you will be able to track the frequency of the noise to see its value and stability.

Once on a job I was diagnosing a problem in an analog power control system, where the control device was acting strangely. Suspecting that noise on the measurement signal line might be causing the problem, I set my Fluke multimeter to measure AC volts, and read a noise voltage of several tenths of a volt (superimposed on a DC signal a few volts in magnitude). This told me the noise was indeed a significant problem. Pressing the "Hz" button on my multimeter, I measured a noise frequency of 360 Hz, which happens to be the "ripple" frequency of a six-pulse (three-phase) AC-to-DC rectifier operating on a base frequency of 60 Hz. This told me where the likely source of the noise was, which led me to the physical location of the problem (a bad shield on a cable run near the rectified power output wiring).

4.2.6 Generating test signals

Modern digital multimeters are fantastically capable measurement tools, but did you know they are also capable of *generating* simple test signals? Although this is not the design purpose of the resistance and diode-check functions of a multimeter, the meter does output a low DC voltage in each of these settings.

This is useful when qualitatively testing certain measurement instruments such as data acquisition modules and other meters designed to input a low-voltage or low-current DC signal. By setting a multimeter to either the resistance (Ω) or diode check function and then connecting the test leads to the input terminals of the signal-sensing device, that device's response may be noted.

Of course, this is a *qualitative* test only, since multimeters are not designed to output any precise amount of voltage in either the resistance or diode-check modes. However, for testing the basic response of a process indicator, recorder, controller, data acquisition channel, DCS input, or any other DC-signal-receiving devices, it is convenient and useful. In every multimeter I have ever tried this with, the diode-check function outputs *more* voltage than the resistance measurement function⁶. This gives you two levels of "test signal" generation: a low level (resistance) and a high level (diode check). If you are interested in using your multimeter to generate test voltages, I recommend you take the time to connect your multimeter to a high-impedance voltmeter (such as another digital multimeter set to measure DC volts) and note just how much voltage your meter outputs in each mode. Knowing this will allow you to perform tests that are more quantitative than qualitative.

⁶There is a design reason for this. Most digital multimeters are designed to be used on semiconductor circuits, where the minimum "turn-on" voltage of a silicon PN junction is approximately 500 to 700 millivolts. The diode-check function must output more than that, in order to force a PN junction into forward conduction. However, it is useful to be able to check ohmic resistance in a circuit *without* activating any PN junctions, and so the resistance measurement function typically uses test voltages *less than 500 millivolts*.

4.2.7 Using the meter as a temporary jumper

Often in the course of diagnosing problems in electrical and electronic systems, there is a need to temporarily connect two or more points in a circuit together to force a response. This is called "jumpering," and the wires used to make these temporary connections are called *jumper wires*.

More than once I have found myself in a position where I needed to make a temporary "jumper" connection between two points in a circuit, but I did not have any wires with me to make that connection. In such cases, I learned that I could use my multimeter test leads while plugged into the *current-sensing* jacks of the meter. Most digital multimeters have a separate jack for the red test lead, internally connected to a low-resistance *shunt* leading to the common (black) test lead jack. With the red test lead plugged into this jack, the two test leads are effectively common to one another, and act as a single length of wire.



Touching the meter's test leads to two points in a circuit will now "jumper" those two points together, any current flowing through the shunt resistance of the multimeter. If desired, the meter may be turned on to monitor how much current goes through the "jumper" if this is diagnostically relevant.

An additional benefit to using a multimeter in the current-measuring mode as a test jumper is that this setting is usually current-protected by a fuse inside the meter. Applying jumper wires to a live circuit may harbor some danger if significant potential and current-sourcing capability exist between those two points: the moment a jumper wire bridges those points, a dangerous current may develop within the wire. Using the multimeter in this manner gives you a *fused* jumper wire: an added degree of safety in your diagnostic procedure.

Linux computer operating system

Linux is a powerful and open-source Unix work-alike operating system for a wide range of computing platforms. One of the appeals of Linux is its extensive array of "utility" programs that do useful tasks. This chapter outlines some of them.

5.1 Linux utilities for command-line navigation

The command-line user environment is crude in appearance compared to a point-and-click "graphical window" interface, but much more powerful in many respects.

To list the contents of whatever directory (folder) you happen to be "in" at any given time, simply use the "listing" command 1s. To see a listing of all files and subdirectories in full detail showing such things as date, size, etc. simply use the "long" option for 1s like this:

ls -1

To navigate downn one level into a sub-directory (sub-folder) with a given name, use the *change directory* command **cd** followed immediately by the name of the desired directory. For example, if there is a sub-directory named **src** you would like to descend into, type:

cd src

To navigate one level up in the directory (folder) structure, use the *change directory* command cd like this:

cd ../

Creating a new copy of a file is done using the "file copy" command cp. For example, taking a file named data.csv and copying it to the new name testresults.csv may be done as follows:

cp -vi data.csv testresults.csv

The -vi options invoked here for the cp command instruct it to be *verbose* (telling you what it's doing as it's doing it) and also to *interactively* prompt you in case the command might overwrite existing files.

The "file move" command (mv) works similarly to cp, the major difference being the original filename will cease to exist and only the new one will remain after this command executes. For example, to *rename* the file data.csv to testresults.csv, use the "move" command as follows, again the use of the options -vi being highly recommended so you can know and confirm what will happen:

mv -vi data.csv testresults.csv

Failing to include the -vi options for either command will result in it executing without notice and without asking your permission if a file is to be over-written.

To create a new sub-directory called My_new_directory, use the "make directory" command:

mkdir My_new_directory

To destroy the same, use the "remove directory" command:

rmdir My_new_directory

5.2 Linux utilities for general file viewing and editing

5.2.1 Linux text editors

At some point you will need to use a *text editor* to view and/or edit text-based files, as much of the Linux operating system is controlled by these files. An easy-to-use text editor is nano, using Control-key sequences to do things like save files and exit. A much harder-to-use editor is vim which despite its idiosyncratic behavior is quite popular and found on virtually every Linux system.

Example: using nano and then vim to view a file named myfile.txt:

nano myfile.txt
vim myfile.txt

5.2.2 Linux text-viewing utilities

A useful utility for read-only viewing of text files is cat, which simply prints the contents of a text file to the screen. For larger text files containing multiple pages' worth of text, a useful utility is **less** which gives you up- and down-control of the screen.

Examples:

cat myfile.txt
less myfile.txt

If the file you wish to view isn't plain-text but contains "binary" data, a better tool is hexdump which will output a hexadecimal listing of every byte contained in that file. For example, to view the bytes contained within a graphic JPEG image file named *photo.jpg*, do the following (using the -C option to show it in "canonical" form complete with ASCII interpretation of the bytes):

hexdump -C photo.jpg

A very powerful utility used to scan text files for instances of specific text strings is grep. For example, to scan for lines of text containing the word "computer" inside the file myfile.txt, type:

```
grep computer myfile.txt
```

To scan for text strings containing space characters, the string must be enclosed in quotation marks:

```
grep "computer virus" myfile.txt
```

To report any lines *not* containing a specific text string, use the -v option:

```
grep -v computer myfile.txt
```

5.3 Linux command-line plumbing

One of the powerful features of the command-line environment within Linux is the ability to chain multiple utilities together by means of additional characters. This is referred to as "plumbing" by Linux enthusiasts because it is analogous to piping the discharge of one process into the intake of another.

Sometimes we might wish to take the screen-output of one command and "plumb" it as the input to another command. For example, suppose we wish to scan all running processes on the computer using the ps -e command but only display those lines of text containing the word "server", such as in a case where we want to know how many server programs are currently running. To do this we may plumb together the ps -e and grep commands using the "pipe" symbol (1) like this:

ps -e | grep server

Similarly, if we wish to record the output of any command-line utility to a text file rather than print it to the screen, we may use the "redirection" (>) symbol between the command and the name of the new text file. For example, taking a detailed listing of all files in the present directory and saving that list to a file named file_log.txt:

ls -l > file_log.txt

If we later wish to append the contents of the file listing.txt to our log file, we may do so using the append redirection symbol (>>) as follows:

cat listing.txt >> file_log.txt

If a particular utility requires text data to be sent to it upon invocation, we may do so using another redirection command (<). Legacy versions of SPICE required this to read netlists, as shown in the example below:

spice < netlist.cir</pre>

5.4 Linux utilities for networking

To remotely log into a computer running Linux, you may use the "secure shell" utility (ssh). The Linux target machine must be running an instance of ssh server in order for a remote client to connect. On modern Windows operating systems ssh is a built-in utility, but you may also use third-party apps such as Bitvise.

Invoke **ssh** as follows, assuming a login name of "Tony" on the target Linux computer having an IP address of 169.254.10.1:

ssh -l Tony 169.254.10.1

You will be prompted by a request for the correct password. Once logged in, you have the same access and level of control as any user typing on a keyboard directly connected to that target Linux machine!

A very powerful networking utility useful for sending messages and other text-based data across TCP/IP connections is "netcat" or nc. This is a client/server toolset which means someone must start up a netcat server on one end and a client on another end of the network. Typically a server session is invoked on one computer to "listen" for incoming messages like this, specifying a TCP port number in the last argument:

nc -lv 1234

Then a client session is started on another computer, and directed to the first computer's IP address (e.g. 169.254.10.3) and TCP port number (e.g. 1234) like this:

nc -v 169.254.10.3 1234

Once this connection is established, people at both computers will be able to type messages back and forth to each other! If we wish to send text-content files from one computer to the other, we may use the "zero" option for the client end and redirection operators as follows:

Server end

nc -lv 1234 < source_file.txt</pre>

Client end

nc -zv 169.254.10.3 1234 > dest_file.txt

5.5 Linux utilities for programming

5.6 Linux utilities for PostScript document files

Digital radio modules and software

6.1 Software-defined radio receiver recommendations

Of all the software-defined radio (SDR) hardware units I've tested, the *RTL-SDR* unit sold by RTL-SDR Blog is by far the best financial value. Other SDRs may have better resolution or have capabilities such as transmission, but the extremely low cost and wide frequency range of the RTL-SDR receiver makes it an outstanding value for students. It receives all frequencies between 500 kHz and 1766 MHz, and also offers 4.5 Volt DC "bias tee" capability for powering such devices as Low-Noise Amplifier (LNA) modules with DC power impressed on the same coaxial connection as the RF signal:



The single most important piece of advice I have for using the RTL-SDR module is to utilize a USB extension cable when connecting it to your computer's USB port, as the device itself is rather long and heavy and will place mechanical strain on that connector if directly plugged in.

If you don't mind a narrower frequency range, the Airspy HF+ Discovery receiver unit is an outstanding performer with greater signal sensitivity than the less expensive RTL-SDR receiver. It receives lower-frequency signals between 0.5 kHz and 31 MHz, and higher-frequency signals between 64 MHz and 260 MHz. Note how low the frequency range goes (down to 500 Hz!) but that there is a "dead" zone between 31 MHz and 64 MHz, and it tops out at only 260 MHz compared to the RTL-SDR's maximum frequency of nearly 1.8 GHz:



Airspy provides a simple-to-use server for sending the receiver's I-Q data stream over a TCP/IP network, which makes it particularly useful for distributed applications with a remote-mounted receiver.

All SDR receivers have substantial power requirements, the RTL-SDR drawing nearly 300 mA from its USB (5 Volt) cable and from whatever computer it's plugged into. This is important to note especially if you are powering the SDR receiver from a battery-powered computer, and/or one with limited USB power-sourcing capability such as a single-board PC.

6.2 Airspy spyserver software tips

Airspy manufactures easy-to-use and powerful software defined radio (SDR) receiver units. A software application called **spyserver** may be used to stream data from one of these SDR receivers onto a network so that multiple client applications can seek RF signals from the receiver. The following instructions are for building the libairspy USB driver from source code on a Raspberry Pi computer running Debian-based Linux. This building-from-source is likely necessary in order for spyserver to actually connect with the AirspyHF+ radio unit via USB. I am indebted to John's Tech Blog (https://hagensieker.com/2019/03/16/airspyhf-spyserver-on-raspberry-pi/) for details on how to build and install the driver from source code, the Linux command-line instructions for this task being John's words verbatim.

First, use the apt utility in Debian Linux to install all the necessary software packages for this purpose, running as root ("sudo") for super-user privileges:

sudo apt-get install build-essential cmake libusb-1.0-0-dev pkg-config

Next, it is advisable to install some more dependencies in order to avoid having spyserver just connect and then disconnect upon invocation:

sudo apt-get install airspy libairspy0 libairspy-dev

Having these dependencies allows you to easily check your serial number for the conf file below by issuing the airspy_info command:

airspyhf_info

Issue the following commands in order to build and install the driver on the Raspberry Pi computer:

```
mkdir airspy
cd airspy
wget https://github.com/airspy/airspyhf/archive/master.zip
unzip master.zip
cd airspyhf-master
mkdir build
cd build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

Once the driver has been built from source code and installed, here is how to run the **spyserver** software on the Raspberry Pi computer. The spyserver file is an executable ready-to-run, but must be run as root:

sudo ./spyserver &

Leave the plain-text spyserver.config file as downloaded, unless you want to change some of the details. The bind_host parameter should be left at 0.0.0.0 and if the bind_port is left at the 5555-6666 range it means the spyserver program will randomly grab one TCP port from within this range when invoked.

When running an SDR client application such as SDR# on a computer remote to the Raspberry Pi, you set the "Source" to "AIRSPY Server Network" and then simply specify the Raspberry Pi's IP address and whatever TCP port number the spyserver grabs when started. For example, sdr://169.254.173.16:5555 if spyserver happened to use TCP port 5555 on a Raspberry Pi with 169.254.173.16 as its IP address.

If you run spyserver in the background (i.e. ./spyserver &) then it is possible you may eventually end up with multiple instances running simultaneous which is no good. To check the running process list on the Raspberry Pi, run the ps command with the -e ("show everything") option and the pipe that to grep to display only those lines containing the word "spy" like this:

ps -e | grep spy

6.3 GNU Radio Companion software tips

GNU Radio is a free and open-source software package for performing digital signal processing of the type often necessary for software-defined radio systems. GNU Radio Companion is a graphical user interface to GNU Radio allowing one to build digital signal processing chains using drag-and-drop function blocks connected together with arrows rather than writing Python or C++ code as is necessary with GNU Radio by itself.

In making a graphical function-block diagram (called a *flowgraph*), every function block inserting a stream of data into GNU Radio from an external device is called a *source*, while every function block sending a data stream to some external device is called a *sink*. For example, a software-defined radio (SDR) receiver sending I-Q data to GNU Radio would do so via a source function block. Conversely, an SDR transmitter would need a sink block to get data to it from GNU Radio. Visual displays to the computer's monitor such as spectrum displays and oscilloscopes also use sink blocks.

Unfortunately, the folder organization for all the function blocks within GNU Radio Companion leaves something to be desired. While a "magnifying glass" search tool exists in the software, it is a literal text search which means it doesn't work unless your text string is 100% correct. The following is a list of function blocks I've found useful for creating flowgraphs:

- Core/Soapy/Source/ this folder contains source blocks for commonly available softwaredefined radio (SDR) receiver hardware including RTL-SDR, AirspyHF, HackRF, PLUTO, and others!
- Core/Soapy/Sink/ this folder contains source blocks for commonly available software-defined radio (SDR) transmitter hardware including HackRF, PLUTO, and others!
- Core/Waveform Generators/ this folder contains waveform source blocks such as Signal Source useful for digitally synthesizing waveform signals such as cosine waves.
- Core/Math Operators/ this folder contains a wide range of mathematical function blocks. Some, like Multiply take in multiple inputs and perform the stated function on those inputs to generate an output. Others like Multiply Const take in a single signal and perform the stated function with a variable which may be sourced by a Variable block.
- Core/Variables/ this folder contains multiple function blocks useful for establishing values of variables used in a flowgraph.
- Core/GUI Widgets/QT/ this folder contains a wide range of function blocks for user-interface functions such as dials, sliders, etc.
- Core/Instrumentation/QT/ this folder contains a set of "sink" function blocks useful for user-interface indicators such as spectrum displays (QT GUI Frequency Sink) and oscilloscopes (QT GUI Time Sink) and I-Q constellation diagrams (QT GUI Constellation Sink).
- Core/Modulators/ this folder contains a wide range of function blocks useful for modulating

and demodulating different types of signals, including AM, FM (narrow- and wide-band), GFSK, etc.

- Core/Resamplers/ this folder contains resampling function blocks useful for interpolation and/or decimation. The Rational Resampler does both as a ratio of interpolation to decimation, hence the name.
- Core/Audio/ this folder contains function blocks for encoding, decoding, sourcing, and sinking audio signals. The Audio Sink function block is needed to send audio signals to the computer's sound card so you can hear the output of radio receiver flowgraphs. The Audio Source block takes data in from the computer's microphone input which makes GNU Radio capable of doing audio signal processing!
- OsmoSDR/ this folder contains some SDR source and sink function blocks that I've found less useful than the Soapy blocks mentioned previously.

6.4 DIGI XBee PRO digital radio transceiver and DIGI XCTU configuration software tips

These tips refer to the XBee PRO S3B 900 MHz (unlicensed) radio transceivers, product family XBP9B-DM.

- Network ID numbers all radio units must be programmed with the same Network ID in order to communicate with each other. The numerical range is 0x0000 to 0x7FFF which gives 32768 unique network identifiers! RF transmissions from other units are still received by an XBee unit even if the Network IDs are different. In other words, the modules "hear" all RF broadcasts, and differentiate modules in and out of network by analyzing the Network ID value sent in the data packets. This means the "Rx" LED of a module will blink if it "hears" data broadcast by any neighboring XBee module, even modules with different Network IDs! I strongly recommend accessing each module using XCTU software in a "radio quiet" environment.
- **Transparent Mode** when set to Transparent Mode, the modules function as a broadcast half-duplex serial network: any serial data transmitted by one is received by all others programmed with the same Network ID value.
- Loopback jumper beware of the loopback jumper on the DIGI development board, which when in place connects Tx (Pin 2) to Rx (Pin 3) and will mess up any attempt to drive the Rx pin from an external source!
- **RF module discovery** an interesting bug is that when you are "discovering" a local RF module using XCTU software, the serial discovery process may be hampered if there is incoming communication from other RF modules on the same Network ID. A good diagnostic indicator is to check the "Rx" LED on the motherboard to see if it's blinking, which indicates incoming RF data. You'll notice that these same LEDs blink whether indicating RF data or serial (USB cable) data, which suggests a common channel where those data streams (i.e. received RF data from other modules, and "discovery" query data from the PC hosting XCTU) might collide. Another interesting bug is that the "Add a radio module" feature seems more reliable than the "Discover radio devices". If a module does have difficulty connecting and needs to be reset, this reset seems to be done automatically when the "Add a radio module" button is clicked, but not when the "Discover radio devices" button is clicked.
- Serial console a related quirk is that module parameters should not be updated while the serial console is connected. Either shut down the serial console or "close" the open connection prior to making any parameter changes! You must "disconnect" the serial monitor before making any parameter changes to the radio unit, or else those parameter changes may not actually take!
- Serial buffer overflow serial buffers in the DIGI modules can overflow if you send data to them too quickly! For example, when sending ASCII text data from a microcontroller to a

DIGI radio configured for Transparent Mode it is advisable to program the microcontroller to pause between successive transmissions.

• **Power consumption** – these radio units draw a significant amount of current on the 3.3 Volt power pin when starting up, more than what some microcontrollers such as the Arduino Nano can source from their regulated +3.3 V power pins! Make sure these are well-decoupled to be able to source that start-up current to the DIGI modules!

Programmable test equipment

7.1 Siglent SPD1305X power supply programming tips

Important tips for communicating SCPI commands to this instrument for the purpose of automated testing:

- Note that the Teledyne-LeCroy model T3PS16081P is actually a Siglent brand power supply in disguise!
- Set my_PS.query_delay = 0.25 in order to avoid errors. By default the delay time for a SCPI "query" command is set for 0.0 seconds which causes time-out errors for this power supply. A delay of one-quarter of a second seems more than sufficient.
- Set my_PS.write_termination = '\n' in order to avoid errors. By default the line-terminating character sequence for all "write" SCPI commands is \r\n which is a "carriage-return" character followed by a "linefeed" character. To avoid this we explicitly re-initialize this attribute to simply be a linefeed character.
- The user manual is replete with typographical errors in its command summary, in the form of extra "space" characters shown between data fields of the SCPI commands. For example, the SCPI command to turn on the power supply's channel 1 output is shown as OUTPut CH1, ON in the manual, but the only way this command would actually work is if I omitted the "space" character between the comma and the letter "O" in "ON" like this: OUTPut CH1, ON.
- The Python command dir() is extremely useful for identifying attributes (i.e. variables) within methods (i.e. functions). For example, issuing the Python command dir(my_PS) lists all the attributes found within the method instantiated as my_PS in the working example first shown in this Case Tutorial section. This is how I was able to find the write_termination attribute which instructs Python how to terminate each SCPI command line, as well as the query_delay attribute specifying how long to wait after issuing a SCPI query before declaring a time-out error.

7.2 NanoVNA vector network analyzer tips

These notes apply to the model NanoVNA-H, hardware version 3.4 manufactured in 2007.

Usage instructions:

- Use a plastic guitar pick or the blunt end of a plastic ball-point pen as a stylus, not your finger!
- Set calibration frequency range with STIMULUS (set to 50 kHz to 900 MHz for calibration!) ; e.g. START = 50k and STOP = 900M
- Go to CAL menu and then CALIBRATE sub-menu ; calibrate by pressing OPEN (the next option will color green after doing this), calibrate by connecting shorting cap and pressing SHORT, calibrate by connecting load caps and pressing LOAD, press DONE, save calibration as "SAVE 0" or "SAVE 1", etc.
- Check calibration by connecting short/load/open and monitoring the display ; 50 Ohm load should put marker at center of Smith chart, open should put marker at right side of Smith chart, short should put marker at left side of Smith chart
- Set testing frequency range with STIMULUS (e.g. frequency range for testing CB antenna shown below); e.g. START = 26.965 MHz and STOP = 27.405 MHZ
- DISPLAY menu and then FORMAT sub-menu ; LOGMAG = S11 Return Loss in dB, SWR = S11 VSWR, LINEAR = same as LOGMAG but ratio instead of dB, SMITH = shows a Smith chart for complex impedance normalized to 50 Ohms, POLAR = same as SMITH but with markers shown as complex numbers

NanoVNASaver PC software

- First thing to do after starting software and plugging NanoVNA into the computer using a USB cable is to select the appropriate Serial port ; click on "Rescan" until it shows the NanoVNA option
- Click on "Connect to device"
- Must do the calibration procedure from the NanoVNA itself, not through the NanoVNASaver software!
- "Sweep settings" defaults to Single Sweep where you have to click "Sweep" every time to take a measurement! Switch to Continuous Sweep to do what the NanoVNA does by default when not connected to the NanoVNASaver software.
- Click on "Display setup" button to change screen colors (I prefer "Show lines" and "Dark mode" options turned on); within that pop-up window, click on the "Settings" button to choose what parameters are displayed for each of the markers
- Right-clicking on any graph gives a menu of options, including log/linear axes, saving image to a .png file, etc.
- The "Analysis" button opens up a pop-up window where you can select a variety of analyses to do, including minimum/maximum search for R or X, etc.

7.3 Mcumall GQ-4x4 EPROM programmer tips

A file named devices.txt is necessary for this program to function. You may need to manually create a directory and save this file to that directory when installing this software on your computer.

The programming software must be run as "Adminstrator" in Microsoft Windows in order to write data to a device. If run with regular user privileges, you can read data from the IC but not write to it!

???

- 8.1 ??? tips
- 8.2 ??? tips
- 8.3 ??? tips
- 8.4 ??? tips

CHAPTER 8. ???

40

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- <u>Study principles, not procedures.</u> Don't be satisfied with merely knowing how to compute solutions learn *why* those solutions work.
- <u>Identify</u> what it is you need to solve, <u>identify</u> all relevant data, <u>identify</u> all units of measurement, <u>identify</u> any general principles or formulae linking the given information to the solution, and then <u>identify</u> any "missing pieces" to a solution. <u>Annotate</u> all diagrams with this data.
- <u>Sketch a diagram</u> to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- <u>Perform "thought experiments"</u> to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- <u>Simplify the problem</u> until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- <u>Check for exceptions</u> to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- <u>Work "backward"</u> from a hypothetical solution to a new set of given conditions.
- <u>Add quantities</u> to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- <u>Sketch graphs</u> illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- <u>Treat quantitative problems as qualitative</u> in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- <u>Consider limiting cases.</u> This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- <u>Check your work.</u> This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment¹ where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic² dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity³ through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

¹In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

²Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

³This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, the fundamental goal of education is for each student to learn to think clearly and independently. This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.
- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.
- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers the goal is to practice the articulation and defense of one's own reasoning.
- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.
- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.
- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied⁴ effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge⁵ one another.

To high standards of education,

Tony R. Kuphaldt

⁴As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

 $^{^{5}}$ Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

Appendix C Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The GNU/Linux computer operating system

There is so much to be said about Linus Torvalds' Linux and Richard Stallman's GNU project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of Linux back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient Unix applications and scripting languages (e.g. shell scripts, Makefiles, sed, awk) developed over many decades. Linux not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's Vim text editor

Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer Vim because it operates very similarly to vi which is ubiquitous on Unix/Linux operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's T_{EX} typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus The Art of Computer Programming, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TFX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TFX is a programmer's approach to word processing. Since T_{FX} is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of T_FX makes it relatively easy to learn how other people have created their own T_FX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft Word suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is All You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, T_EX is inherently extensible. So, years after the release of T_EX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LAT_EX , which is the markup language used to create all ModEL module documents. You could say that T_EX is to LAT_EX as C is to C++. This means it is permissible to use any and all T_EX commands within LAT_EX source code, and it all still works. Some of the features offered by LAT_EX that would be challenging to implement in T_EX include automatic index and table-of-content creation.

Tim Edwards' Xcircuit drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for Xcircuit, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

Gimp graphic image manipulation program

Essentially an open-source clone of Adobe's PhotoShop, I use Gimp to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although Gimp does offer its own scripting language (called Script-Fu), I have never had occasion to use it. Thus, my utilization of Gimp to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

SPICE circuit simulation program

SPICE is to circuit analysis as T_{EX} is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer SPICE for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of SPICE, version 2g6 being my "go to" application when I only require text-based output. NGSPICE (version 26), which is based on Berkeley SPICE version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all SPICE example netlists I strive to use coding conventions compatible with all SPICE versions.

Andrew D. Hwang's ePiX mathematical visualization programming library

This amazing project is a C++ library you may link to any C/C++ code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as Mathematica or Maple to do. It should be said that ePiX is not a Computer Algebra System like Mathematica or Maple, but merely a mathematical visualization tool. In other words, it won't determine integrals for you (you'll have to implement that in your own C/C++ code!), but it can graph the results, and it does so beautifully. What I really admire about ePiX is that it is a C++ programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a C++ library to do the same thing he accomplished something much greater. gnuplot mathematical visualization software

Another open-source tool for mathematical visualization is gnuplot. Interestingly, this tool is not part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" not be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my gnuplot output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of Gimp to do rudimentary image editing, my use of gnuplot only scratches the surface of its capabilities, but the important points are that it's free and that it works well.

Python programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type from math import * you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (from cmath import *). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. Licensor means the individual(s) or entity(ies) granting rights under this Public License.

i. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority. Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.

Appendix E

References

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

17 April 2025 – substantial additions to the "Digital radio modules and software" chapter including some section title changes.

11 April 2025 – substantial additions to the "Multimeter tips" chapter.

4-5 April 2025 – added content to the "Solderless breadboard caveats" chapter.

26 March 2025 – added content to the "Multimeter tips" and "Solderless breadboard caveats" and "Linux computer operating system" chapters.

21 March **2025** – added notes on using **spyserver** software to stream software-defined radio data on a network via a Raspberry Pi 4 single-board computer running Debian Linux OS.

18 March 2025 – document first created.

Index

Adding quantities to a qualitative problem, 42 Annotating diagrams, 41

Checking for exceptions, 42 Checking your work, 42 Code, computer, 49

Digital multimeter, 11 Dimensional analysis, 41 DMM, 11

Edwards, Tim, 50

Fluke brand multimeters, 11 Fluke SV225 stray voltage adapter, 15

Graph values to solve a problem, 42

Harmonic frequency, 17 How to teach with these modules, 44 Hwang, Andrew D., 51

Identify given data, 41 Identify relevant principles, 41 Instructions for projects and experiments, 45 Intermediate results, 41 Inverted instruction, 44

Jumper wire, 20

Knuth, Donald, 50

Lamport, Leslie, 50 Limiting cases, 42

Moolenaar, Bram, 49

Open-source, 49

Pipe, Linux, 24, 30

Problem-solving: annotate diagrams, 41 Problem-solving: check for exceptions, 42 Problem-solving: checking work, 42 Problem-solving: dimensional analysis, 41 Problem-solving: graph values, 42 Problem-solving: identify given data, 41 Problem-solving: identify relevant principles, 41 Problem-solving: interpret intermediate results, 41 Problem-solving: limiting cases, 42 Problem-solving: qualitative to quantitative, 42 Problem-solving: quantitative to qualitative, 42 Problem-solving: reductio ad absurdum, 42 Problem-solving: simplify the system, 41 Problem-solving: thought experiment, 41 Problem-solving: track units of measurement, 41 Problem-solving: visually represent the system, 41 Problem-solving: work in reverse, 42 Qualitatively approaching \mathbf{a} quantitative problem, 42Redirection, Linux, 24 Reductio ad absurdum, 42-44 Simplifying a system, 41 Socrates, 43 Socratic dialogue, 44 Stallman, Richard, 49 Star Trek, 11 Thought experiment, 41 Torvalds, Linus, 49 Units of measurement, 41 Visualizing a system, 41

INDEX

Wire, jumper, $\frac{20}{20}$ Work in reverse to solve a problem, $\frac{42}{49}$ WYSIWYG, $\frac{49}{50}$