Extracted from:

Build a Binary Clock with Elixir and Nerves

Use Layering to Produce Better Embedded Systems

The Pragmatic Bookshelf

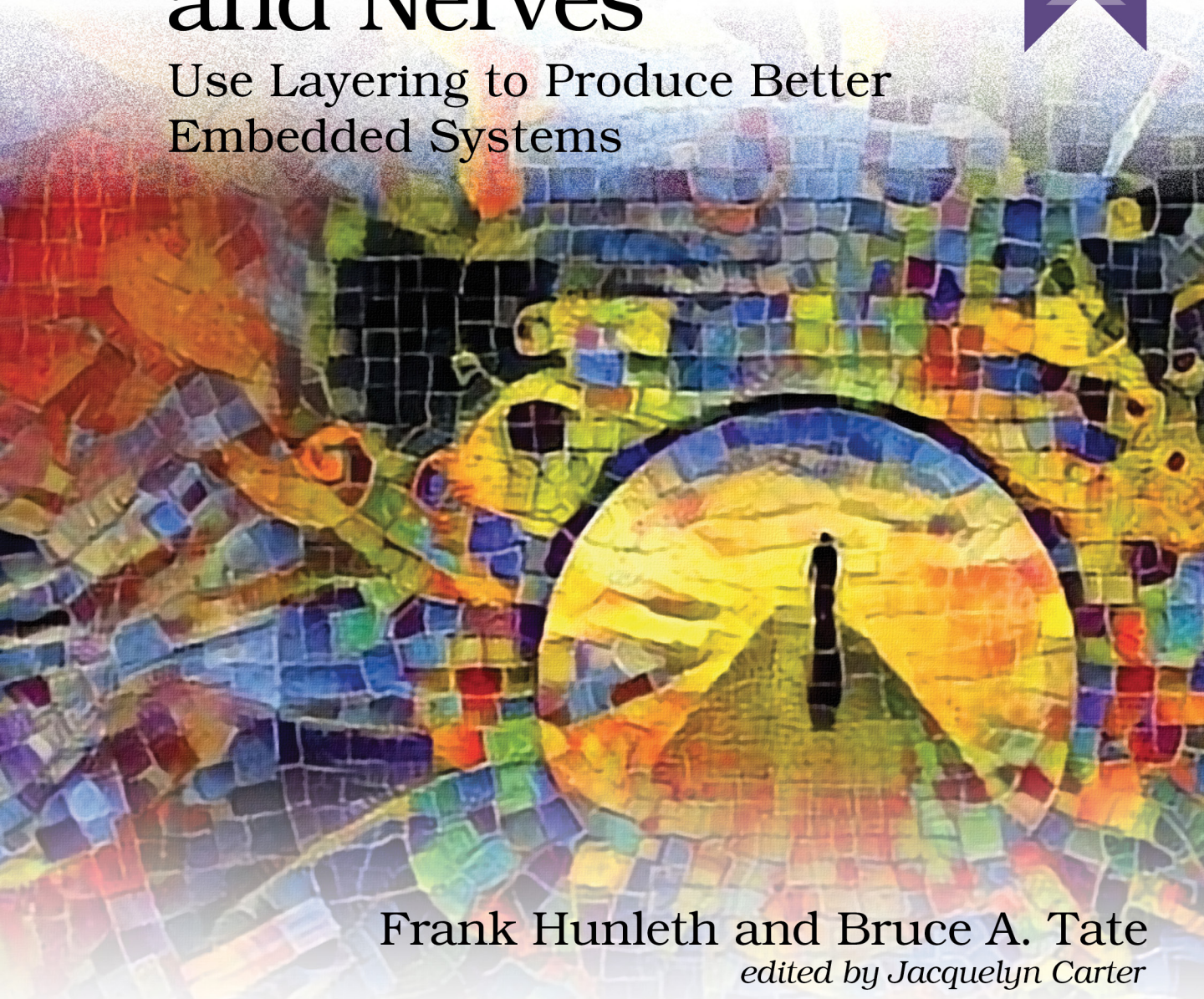
Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Build a Binary Clock with Elixir and Nerves

Use Layering to Produce Better Embedded Systems



Frank Hunleth and Bruce A. Tate

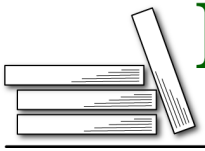
edited by Jacquelyn Carter

Build a Binary Clock with Elixir and Nerves

Use Layering to Produce Better Embedded Systems

Frank Hunleth
Bruce A. Tate

The Pragmatic Bookshelf
Raleigh, North Carolina



For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Jacquelyn Carter
Copy Editor: L. Sakhi MacMillan
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-923-6
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—August 2022

CHAPTER 5

Build the Clock's Circuit

This is the part of the project that all hardware lovers crave. You'll be planning and building the hardware that will make up the clock. It's the shortest part of the book but might take the longest to execute depending on how much experience you have.

For the first time, we'll use a constant current LED driver. Since that much is a mouthful, we'll sometimes refer to it as a constant current driver, or even driver. Here's what it does.

As you work in this chapter, you'll work inside a new `mix` project called `Clock`. It will have the implementation for the entire binary clock. Like software, hardware systems are built in layers. The interfaces between the layers make it easy to isolate services and let engineers focus on one bit of complexity at any given time. The TLC5947 chip we bought in Chapter 3, Build a Circuit, on page ? has the constant current driver that will take care of lighting the LEDs all at once, so all we need to do is attach the LEDs to the TLC5947 board and the TLC5947 board to the Pi.

Here's what the clock face of our project looks like:

The only user interface the user will see is a series of LEDs. We'll mount them in a cabinet by drilling eighteen 5/16-inch holes. Seventeen of the holes will hold live LEDs and one will be a dead LED that we present for symmetry. The

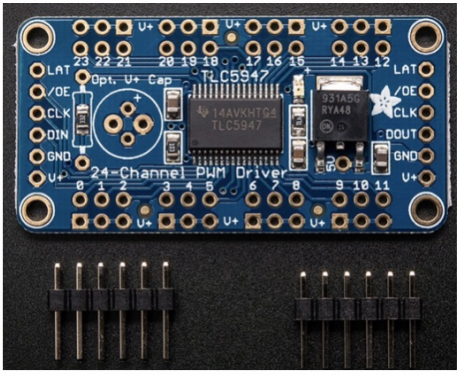
clock will tell time using a binary number system.[1] We'll use one of those LEDs to represent an AM/PM indicator, six bits for the second and minute digits, and four bits for the hour.

When we're done, the user will see only the tips of the LEDs, and we'll connect those wires to our device. Let's plan our attack.

Plan the Hardware

You might be tempted to try to control all of your LEDs from the GPIO pins from the Pi. The problems with that approach are twofold. First, there might be too much power for our project to accommodate. Second, the brightness would probably be uneven, and you might even see some flickering. For this reason, our days of simple GPIO pins are over for this project.

Instead of hooking up resistors and LEDs to individual pins, we'll invite an intermediary onto the scene to control things, like a traffic cop. The Pi will give the intermediary specific instructions for turning on and off groups of LEDs and leave it to the traffic cop to carry out the instructions. Our traffic cop is the standardized interface you first encountered in Chapter 3, Build a Circuit, on page ?. This chip uses a standardized hardware interface called SPI, for *Serial Peripheral Interface,* shown in the following figure.



Each of the holes in the previous figure is a potential connection. The two rows with two holes each, labeled 0–23, are potential devices. We'll connect LEDs to seventeen of them. The two rows of holes on each end represent the input and output connections. We'll connect the inputs to the Pi. If you

1. https://www.mathsisfun.com/binary-number-system.html

wanted to connect more than the 24 LEDs this project requires, you'd chain the outputs from this chip to the inputs of another.

Since we only need one chip, we'll hook up the constant current driver to the Pi, and the LEDs to the driver, and we'll tell the driver to do the work through a library called Circuits.SPI.

That means the hardware side of this project is tedious, but manageable. First, you'll prepare the constant current driver that will serve as the traffic cop. You'll solder headers to the chip so that you can easily make connections with simple jumper wires. When you're done, you'll have an interface board that's ready to accept LEDs and the individual connections to the Raspberry Pi.

Next, you'll connect the constant current driver to the Pi. This step will go quickly because the SPI interface our driver uses is a common hardware interface, so the connections between the chips are well defined. We'll just follow a known schematic that tells us precisely which pins on each chip to connect with jumper wires.

Frank says:

The Value of Datasheets

Working with this constant current driver was great. We didn't have an exact chip in mind. We just went to AdaFruit and searched for a constant current driver for LEDs. Then we downloaded the TLV5947 datasheet, which documented the messages we needed to make in Elixir. I just had to tell Bruce where to put the standard connections on the Raspberry Pi and point him at the Circuits.SPI documentation, and everything worked right out of the box.

Next, you'll build four individual LED groups representing hours, minutes, seconds, and AM/PM bits. You'll use jumper ribbon wires so we can exert just a little control over the inevitable rat's nest of wires. It's a tedious build because each LED has two wires, and seventeen LEDs means we're soldering 34 joints. You'll plug each LED strip onto the header pins you added to the constant current driver.

Finally, before you build a cabinet and install the LEDs, you'll test the hardware. When you're done, you'll have a completed clock that lacks only working firmware. Let's start the hardware assembly with the constant current driver.

Prepare the Constant Current Driver

The TLC5947 will need several connections. There are many ways to solder together persistent connections. You might decide to solder individual components right onto the TLC5947, but that process is error prone, leading to hard-to-find bugs. Instead, you'll use a more forgiving approach. You'll solder on permanent header pins. That way, you can slide temporary jumper wires onto those headers to make connections. The compromise is a temporary connection, but one that's easy to correct should you make mistakes.

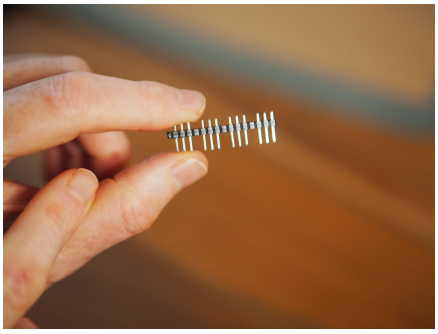
Solder on the Headers

You will need to solder five headers to the board. One six-pin header will cover the inputs, and two long headers will cover each row of LED connections. When you're done, your headers should have long pins protruding from the side of the board that has the chips.

If you've never soldered before, you might want to practice a bit. Try watching a video[2] to get the basics. Make sure you don't have any extra solder between the pins that might cause a short.

Start with the short six-pin header. The board comes with two short headers. You only need one of them. Insert the short pins through the top of the board. Then, use some masking tape or scotch tape to temporarily hold them in place while you turn the board over and solder them up. Alternatively, place the header pins in breadboard, long pins down. Then place the chip upside down over the header pins. Gravity will hold it all in place as you solder it up.

Next, you'll solder on the long boards. The long headers will be too long and will have too many pins. Cut off fifteen or sixteen pins, and remove every fourth pin to leave four groups of three pins, as in the following figure.



2. https://learn.adafruit.com/how-to-solder-headers

Next, insert each of the modified headers into the top of the board, tape them in place, and solder them up. When you're done, you should have four rows of headers running left to right in groups of three, and six pins on your left.

Once you've done that much, you're ready to solder the LEDs to the jumper ribbons.