Designing and Prototyping the Hardware and Software for a Holiday Lights Controller

James Palmer and Dr. James A. Jerkins

Department of Computer Science and Information Systems

UNA COMPUTER SCIENCE and INFORMATION SYSTEMS

Sanders College of Business and Technology
University of NORTH ALABAMA

Abstract

A custom controller was developed using an RP2040 microcontroller to manage RGB LEDs that rely on the WS2811 protocol, which existing code from the manufacturer did not support. The design employed two PIO state machines to control two strings of LEDs, progressing through three prototypes: a breadboard with through-hole components, a PCB printed using a Voltera printer, and a commercially fabricated PCB with surface mount components. The final version included programmable LED displays for American holidays, selectable via a pushbutton.

Introduction

Common RGB LEDs use a non-standard, unique protocol to control the color and intensity. Since no microcontrollers speak the proprietary protocol to control these LEDs, an original controller was designed and implemented using a RP2040 microcontroller. The design uses two programmed I/O (PIO) state machines to manage two strings of fifty RGB LEDs connected to the microcontroller's GPIO pins.

Three versions of the controller were designed and prototyped. The initial hardware prototype used a breadboard with through-hole components and was manually assembled. After verifying this design, a second prototype was created. This version was designed as a printed circuit board (PCB) with surface mount components and fabricated using a Voltera PCB printer at UNA's Generator. For the third version of the controller, a PCB was designed using surface mount components and then commercially fabricated. The components for the final prototype were attached using a reflow oven. The third version had different LED displays for various American holidays which were selected using a pushbutton on the controller.

WS2811 Protocol Support

The first step to creating the holiday lights controller was to write code that could support the WS2811 protocol the LED strings use. The WS2811 data sheet indicates that for "high speed mode" data transfer, bits can be sent at a rate of 800 KHz. Therefore, one bit is sent from the microcontroller's PIO state machine every 1.25 microseconds. Based on code provided by Raspberry Pi for the WS2812 protocol, the state machine's clock rate was set to 10 cycles per bit, or 8 MHz.

Below describes how the clock divider was calculated for the 8 MHz clock rate:

$$div = ClockFrequency/(BitRate * CyclesPerBit)$$
$$div = 125\text{MHz}/(0.8\text{MHz} * 10)$$
$$= 15.625$$
$$freq = 125\text{MHZ}/15.625\text{MHz} = 8\text{MHz}$$

The WS2811 protocol requires that for a 0 bit to be sent, the signal must be high for 0.25 microseconds and then low for 1 microsecond. For a 1 bit, the signal must be high for 0.6 microseconds and then low for 0.65 microseconds. With the state machine's clock period of 0.125 microseconds, it can be determined that the signal will always be high for 2 cycles, either high or low for the next 3 cycles, and then always low for the last 5 cycles. Figure 1 shows oscilloscope output captured during WS2811 program execution.

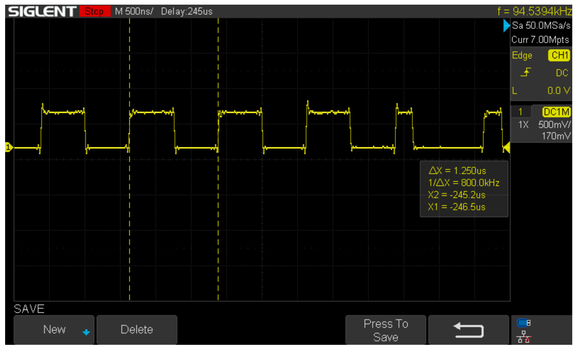

Figure 1:Oscilloscope output from WS2811 program

Bread Board Prototype

Once functioning support for WS2811 was completed, the first prototype for the controller was created on a bread board. The components needed were a barrel connector for an external power supply and a level shifter. The level shifter converted the 3.3V PIO output from the RP2040 to 5V, the expected voltage by the LED strings. Figure 2 shows the actual bread board prototype with no LEDs connected.

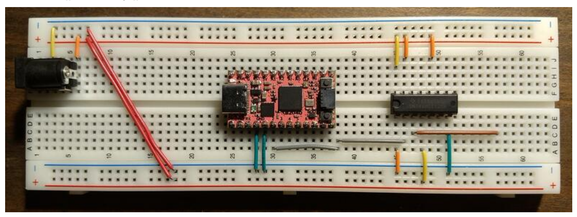

Figure 2:Bread board prototype

PCB Using Voltera Printer

The Voltera V-One circuit board printer allowed for rapid prototyping and testing of PCB designs. The prototype created with the V-One introduced a few more components: a USB-C connector wired to the barrel connector, a pushbutton and resistor that allowed for user input, and wiring for two strings of LEDs to be controlled at the same time. Figure 3 shows the PCB during printing as well as the assembled prototype.

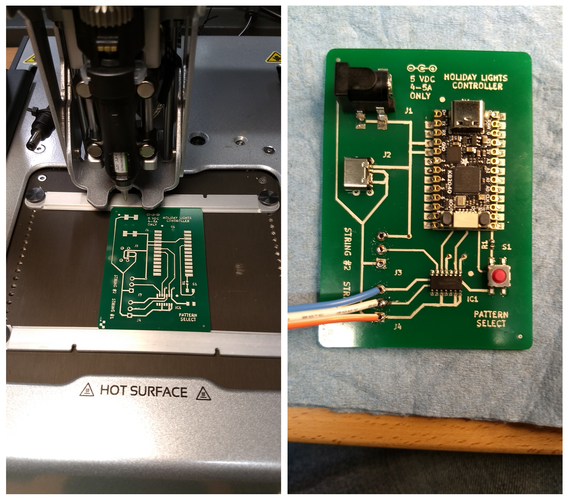

Figure 3:The prototype during printing (left) and the prototype with its surface-mount parts (right)

Final Prototype

The prototype created with the V-One printer had a few issues that the final prototype aimed to correct. First, the USB-C connector was placed too far away from the edge of the board, making it difficult to plug a cable into. Second, The solder paste used by the V-One printer was somewhat weak, leading to the barrel connector breaking off of the board shortly after its first use. Lastly, the traces printed on the board were not incredibly reliable and could easily be scratched or melted off of the board. Using more newly-manufactured paste would correct most of these issues, but a commercially fabricated board was a less costly solution both in time and money. Figure 4 shows the final prototype design with its improved USB-C and barrel connector locations. Figure 5 shows the assembled final prototype, including the USB-C cable that supplies power to the RP2040 from the power supply.

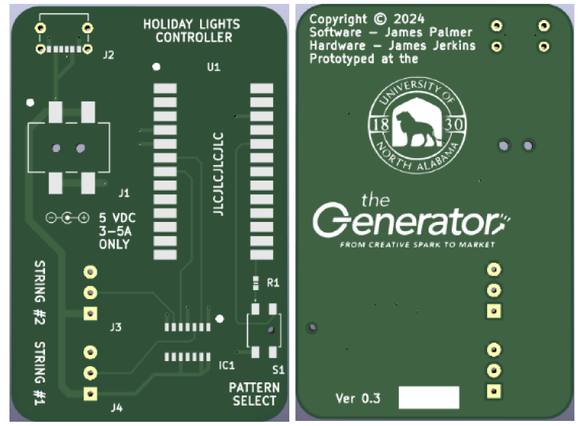

Figure 4:Image of the final PCB design

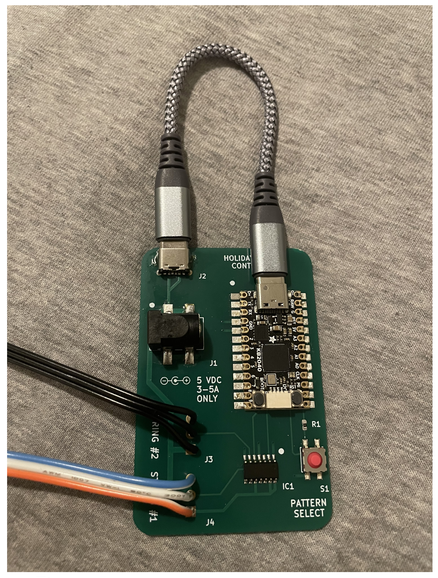

Figure 5:Image of the assembled final prototype

References

- RP2040 Datasheet: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
- WS2811 Datasheet: https://cdn-shop.adafruit.com/datasheets/WS2811.pdf