Milestone 1: Stranger Things Light Walls

Jordan Alberico & Tim Duong
Rowan University

October 21, 2018

1 Design Overview

The goal of this milestone is to use the MSP430F5529 to control the brightness and color of an external RGB LED using PWM and UART. In this implementation, three NMOSes are used as low side switches to provide enough power to the RGB LED. The MSP430 controls the three NMOSes each connected to the Red, Green, and Blue pins of he RGB LED. The MSP430F5529 should be able to receive bytes over UART at 9600 BAUD, take the first four bytes, and transmit any excess bytes. The first byte is the packet length or the number of expected bytes. The MSP430 should receive this and transmit the packet length minus three. The next three bytes are set to control the PWM of the red, green, and blue LED. After transmitting the rest of the expected bytes, the MSP430 will expect the next byte to be packet length and reset once received.

1.1 Design Features

Design features include:

- PWM controlled RGB LED
- RGB LED operating at 5V through low side switches
- UART compatible at 9600 baud

1.2 Featured Applications

Applications include:

- PWM RGB LED color
- Interface with other boards/computer with UART

1.3 Design Resources

Github repository link: `https://goo.gl/7gj6Wy`
2N7000 Datasheet: `https://www.onsemi.com/pub/Collateral/2N7000-D.PDF`

1.4 Board Schematic

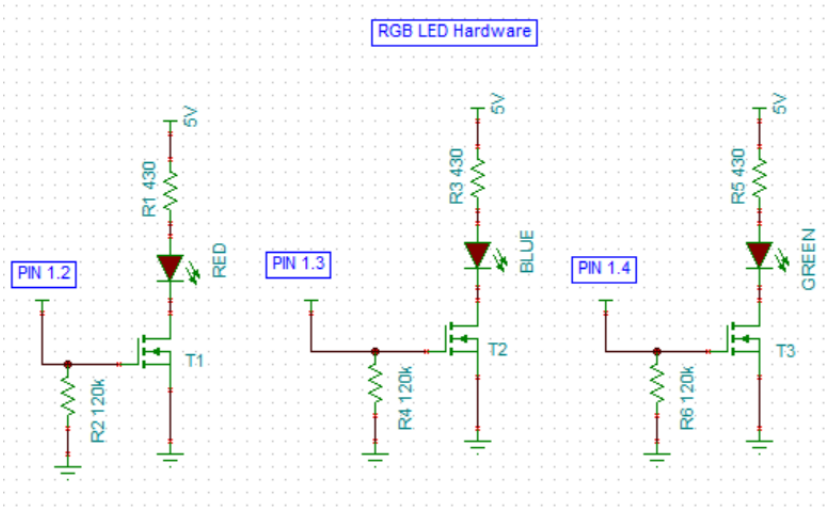

Figure 1: Schematic of the RGB LED

Figure 2 shows how the individual LEDs were connected in the RGB LED. The NMOS functions as a low side switch that turns on when the micro-controller pin is in the on state. This low side switch will be used to pulse the LED. Figure 1 shows the schematic of the RGB LED connected to the micro-controller.

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
SMCLK Frequency	1 MHz	Speed of the timer used for PWM
Baud Rate	9600	Speed of UART Communication
TA0CCR0	0xFF	Defines period of PWM
TA0CCR1	P1.2	Determines on time of red LED
TA0CCR2	P1.3	Determines on time of green LED
TA0CCR3	P1.4	Determines on time of blue LED

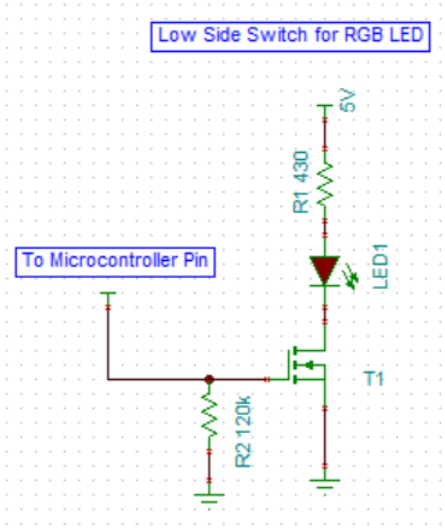

Figure 2: Low Side Switch for an LED

3 System Description

There are three main components in this system: the microprocessor, the low side swithces, and the RGB LED. The microprocessor handles receiving bytes, transmitting bytes, and controlling the PWM for the red, green, and blue LEDs. The low side switches is responsible for protecting the microprocessor from back flow and providing enough power for the RGB LED. The RGB LED is the load the system is trying to drive and control.

3.1 Detailed Block Diagram

Reference Figure 3 for detailed block diagram.

3.2 Highlighted Devices

- (1) MSP430F5529 - controls the PWM signals of the RGB LED.
- (3) 2N7000 Small Signal MOSFETs - to operate as low side switches in the circuit.
- (3) 120k ohm resistor - to use as a pull down resistor to prevent floating on the MSP pins.
- (3) 430 ohm resistor - to use in series with the RGB diodes.
- (1) RGB common anode diode.

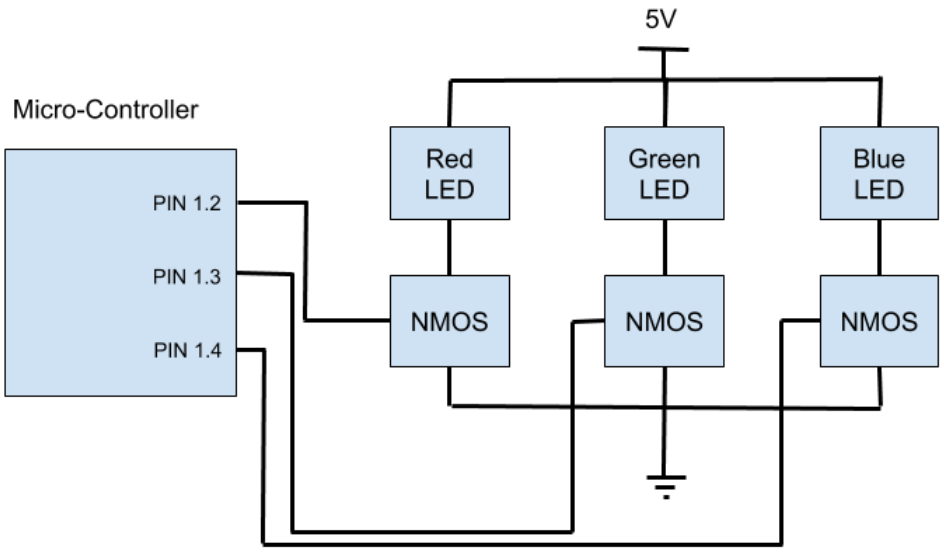

Figure 3: Micro-controller and RGB LED Block Diagram

3.3 Device/IC 1

The MSP board used for the Milestone was the MSP430F5529. The reason the F5529 was picked was for its timer implementation. The MSP430F5529 contains 7 capture/compare registers for Timer A. This is useful for PWM considering capture/compare register 0 can be set as the max period and capture/compare registers 1-3 can be used to assign PWM pulses for red, green, and blue accordingly. The group was familiar with hardware PWM coding on the F5529 considering it was one of the boards selected for the ”Hardware PWM” section of a previous lab.

3.4 Device/IC 2

One possible way to drive the RGB LED was to run the diodes at 3.3 volts, which is possible through the micro-controller. However, in order for the LED to be bright, it had to be ran at 5v. Since the micro-controller cannot output 5v at the pins, a different driving circuit was used for the RGB LED. Three NMOS devices were used as low side switches that could be pulsed with the micro-controller. There was a NMOS for each colored diode in the LED so the colors could be pulsed individually. Since the pins were connected to the gates of the NMOS devices, there was potential for them to have a floating value if the pins went from an on state to an off state. Therefore, as a second order design, a 120k ohm resistor was grounded at the gate of the NMOS. This prevents the pin from having a floating voltage.

4 SYSTEM DESIGN THEORY

4.1 Design Requirement 1

One of the design requirements is to pulse the red, green, and blue part of the RGB LED individually with the MSP430F5529. This requirement was achieved using TimerA and three capture compare registers. Each capture compare register will determine the duty cycle of a specific pin. In this case, CCR1 corresponds to P1.2, CCR2 corresponds to P1.3, and CCR3 corresponds to P1.4. CCR0 is used as the period and is set to 255 because 255 is the maximum decimal number a byte can represent. The timer is set using SMCLK in UPMode and output mode in reset/set. When the counter reaches CCR1, the output on P1.2 will be reset or 0. The same applies to the other CCRs. When the counter reaches CCR0, the output on P1.2-P1.4 will be set to 1 and the counter is reset. By changing the values stored in CCR1-3, the user can control the PWM of P1.2-P1.4.

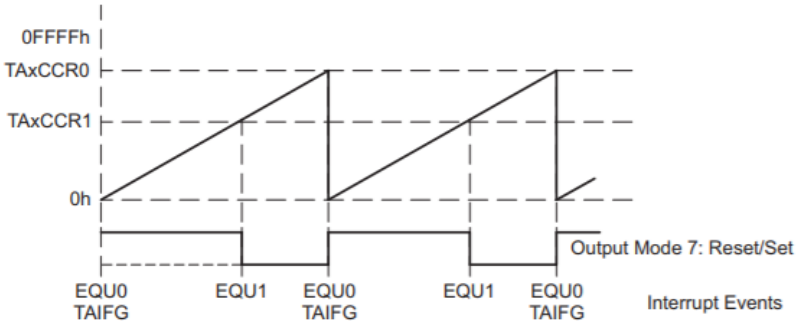

Figure 4: PWM of Output Mode 7

4.2 Design Requirement 2

The addition of the low side switches for the RGB LED were not necessary to the design requirements. However, the low side switches provide the RGB LED with more voltage and current than can be supplied with the micro-controller. The additional hardware connected to the RGB LED shows how to improve this design. Realistically, the micro-controller would not only be controlling one RGB LED. It could be controlling a whole strip of LEDs. If this were the case, the micro-controller does not have enough voltage and current to drive all the LEDs in a strip. Therefore, additional hardware including the amplifying NMOS devices were added.

5 Getting Started/How to use the device

Once the code has been uploaded to the device, verify the hardware has been wired properly. Once the correct pins have been connected to the MOS devices, connect the MSP 430 to your computer through USB. Run the software 'Realterm' (explained in section 6.2) and set up the port and baud rate properly. After that has been completed, bytes can be sent through the software to the MSP. The first byte the MSP will receive is the length bit, then the next three bytes will determine the red, green, and blue lights respectively.

In order to daisy chain the MSP device to other boards, the Rx pins of one board need to be connected to the Tx pin of another board. This will allow packets of data to be received by the first board, modified, then sent to the next board in the line.

6 Getting Started Software/Firmware

6.1 Hierarchy Chart

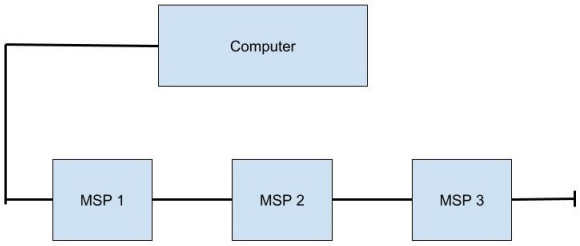

Figure 5: Hierarchy of MSP Devices

Figure 5 shows the connections of MSP boards when they are daisy chained. The connections enter in one board's Rx pin and exit through the boards Tx pin. This allows the board to continually transmit packages of data through the chain. The computer sends the initail package of data.

6.2 Communicating with the Device

The MSP430F5529 can communicate with other devices using the RXB and TXB pins. By default, these pins are jumped so that the user can communicate with the microprocessor through USB. The user can use a program called RealTerm to send bytes to the microprocessor. Ensure the BAUD rate and the ports match up in set up to ensure the two can communicate with one another. If the user wants to daisy chain multiple processors together, they would need to remove the jumper from the RXB and TXB pin. The user would then need to connect their RXB to the TXB pin of

another board and the TXB pin of their board to the RXB pin of another board. Ensure all the microprocessors share a common ground.

7 Test Setup

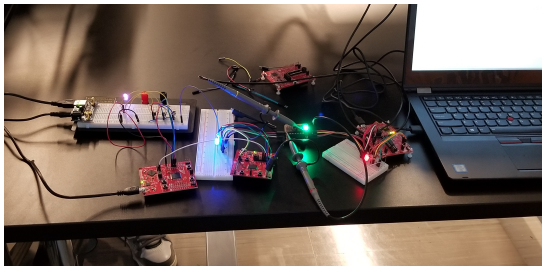

Figure 6: Connected MSP Devices

Figure 6 shows the MSP devices of four groups daisy chained together. The sequence of bytes sent from the computer intended to set the first LED to red, second LED to green, third LED to blue and the last LED to magenta. The light were set correctly by the computer. However, when the computer sent the next package of data, the lights did not respond properly. This meant at some point in the chain one of the boards incorrectly processed the receive data and transmited the improper data.