Milestone 1: Communicating with Will Byers

Brad Anderson and Brandon Salamone
Rowan University

10/16/2017

1. <u>Abstract</u>

The intent of this project was to create a RGB LED capable of being connected in series with other RGB LEDs. The RGB LED will take a string of hexadecimal values over UART, utilize the three least significant bytes to set the color of the RGB, and pass the rest of the string to the other RGB LEDs. The three bytes the RGB LED uses will alter the Duty Cycle of the RGB which enables a particular color to be displayed on the LED.

2. <u>Introduction</u>

In this milestone, the task is to build addressable RGB LEDs. When successfully built, the LED should be able to connect in series with other RGB LEDs which can then be used to display patterns. The RGB node needs to be able to take in a string a Hexadecimal values over the UART RX Line, take the three least significant bytes to determine the RGB values for the given node, and then pass the remaining string to the next RGB node. The RGB values will need to be Duty Cycles corresponding to the individual light colors.

The RGB node is to be configured to wait in Low Power Mode upon startup while it waits for a message to be received over UART. When a message starts to be received, the first three bytes get stored in an RGB buffer to set the current node's color, and the remaining bytes will get stored in a UART TX buffer. When the message is completed received, the bytes in the TX buffer are to be sent to the next RGB node over the UART TX line.

3. <u>Background</u>

3.1 General Concept

In the movie Stranger Things, Will Byers is trapped in a parallel dimension and can only communicate with his mother via a series of christmas string lights on a wall. We will attempt a similar concept using RGB LEDS attached to different boards connected through UART.

3.2 Project Parameters

The end result of the project should behave like the diagram in Figure 1.

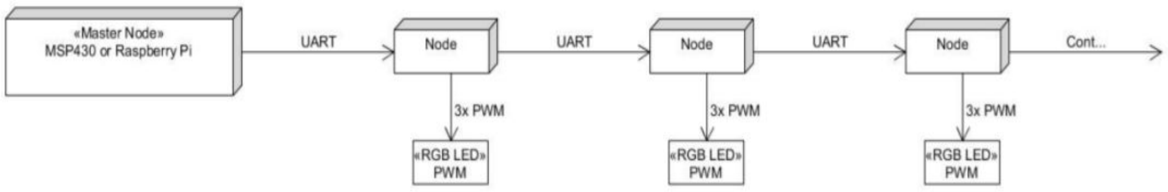


Figure 1. Node Sequence Diagram

As seen in Figure 1, the initial signal will come from the MSP430, through UART to the first RGB node. The least 3 significant bytes will be stored from the signals to control the first RGB LED and then through UART the remaining bytes will be sent to the next RGB LED and so on. The initial string will be 80 bytes. This concept is shown in further detail in the Figure 2.

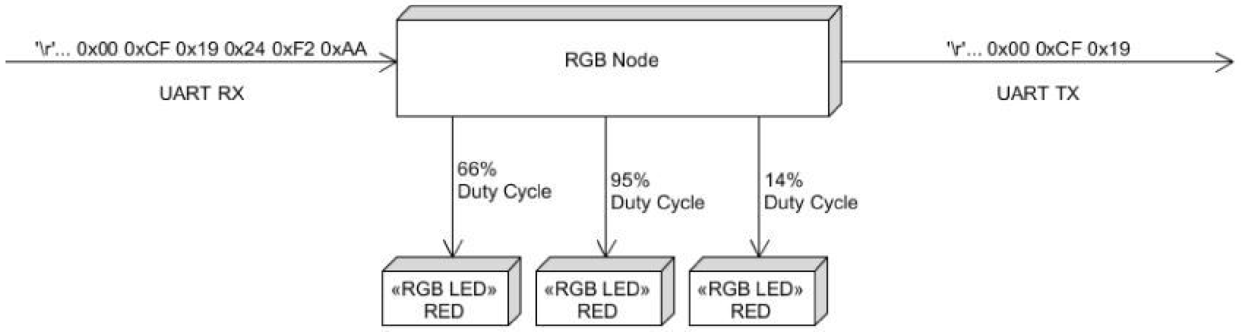
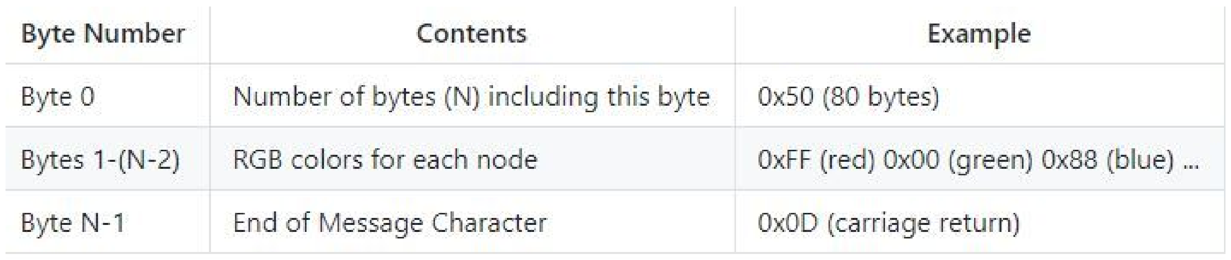


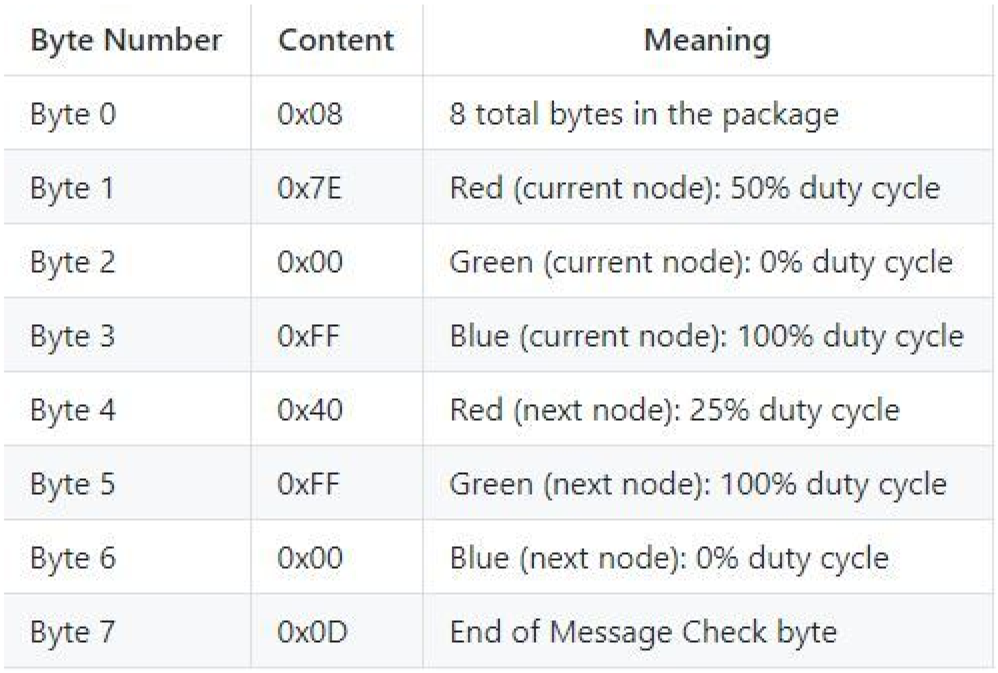
Figure 2. UART Signal Transfer

The bytes that are received will be formatted in the following way: the first byte tells the program how many bytes are left in the sequence, then the next three give the RGB LED information on the proper PWM settings, then the rest of the information will be passed on to the next board, as shown in the following figures.



Byte Number	Contents	Example
Byte 0	Number of bytes (N) including this byte	0x50 (80 bytes)
Bytes 1-(N-2)	RGB colors for each node	0xFF (red) 0x00 (green) 0x88 (blue) ...
Byte N-1	End of Message Character	0x0D (carriage return)

Figure 3. General Byte Index



Byte Number	Content	Meaning
Byte 0	0x08	8 total bytes in the package
Byte 1	0x7E	Red (current node): 50% duty cycle
Byte 2	0x00	Green (current node): 0% duty cycle
Byte 3	0xFF	Blue (current node): 100% duty cycle
Byte 4	0x40	Red (next node): 25% duty cycle
Byte 5	0xFF	Green (next node): 100% duty cycle
Byte 6	0x00	Blue (next node): 0% duty cycle
Byte 7	0x0D	End of Message Check byte

Figure 4. Specific 8 Byte Sequence Index

4 <u>Evaluation and Results</u>

The guidelines for this milestone did not state a specific board to use for the RGB sequencing. Instead the freedom to choose a specific board was given. Using a common cathode RGB LED meant that our circuit needed the common pin of the LED to be grounded while specific voltage inputs were set to each pin. Current limiting resistors were used for safety measures.

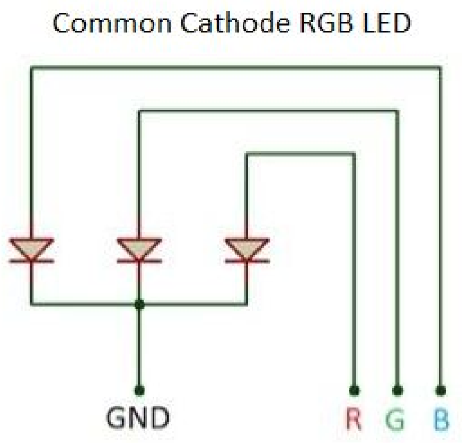


Figure 5. Common Cathode RGB LED Diagram

4.1 Choosing the Correct Board

Since the RGB LED has three different color values to set, the signal needs to PWM three times, one for each value; red, green, blue. This means at the very least, the board chosen must have at least three capture compare registers, not including CCR0.

4.1.1 MSP430G2553

The Msp430G2553 could not be used for this project because it only supports CCR0, CCR1 and CCR2, while our implementation required that CCR3 be used to store information from UART.

4.1.2 MSP430FR6989

MSP430FR6989 board was not considered for this project due to its high cost and unnecessary added features such as the LCD display.

4.1.3 MSP430FR5994

The MSP430FR5994 board was also not considered for the project due to the fact that it too is an expensive board and has unnecessary features such as the super-capacitor.

4.1.4 MSP430FR2311

We chose not to use this processor because like the MSP430G2553 it does not support usage of the CCR3 register, which is critical to making our implementation of the UART code work.

4.1.5 MSP430F5229

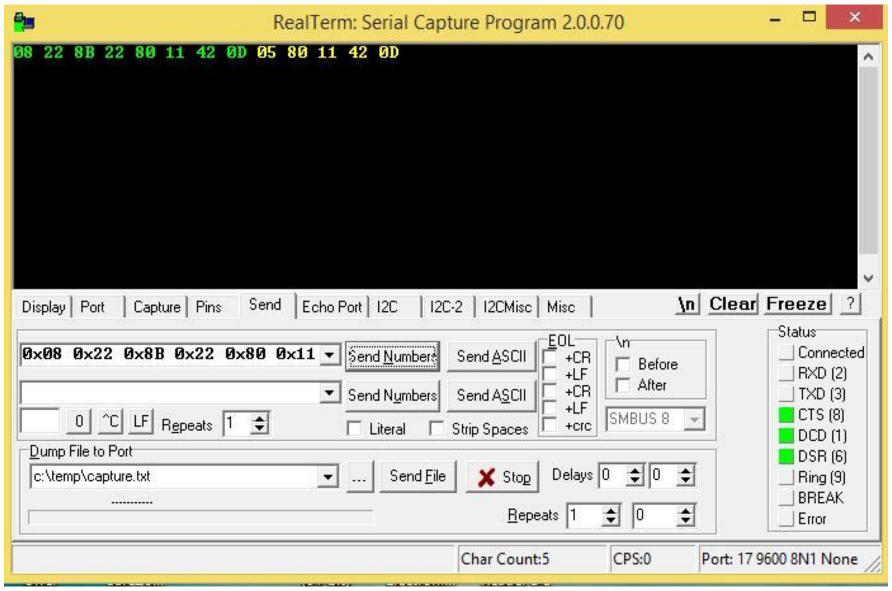
The MSP430F5229 was used since it has the appropriate number of CCR registers and the hardware PWM was easy to implement and use to control the RGB LED. There were also no unnecessary features that would drive up the cost of the board.

4.2.1 UART Communication (Code)

In order to communicate with the UART pins on the F5529, there are several UART-specific registers that need to be set in order to initialize the UART peripheral. Luckily, the resource library in TI provided an example as to how to set each register. Once the UART peripheral is initialized, an ISR needs to be coded to handle the receiving and transmitting of data through UART. Our interrupt service routine begins by setting the number of bytes in the packet equal to the first hexadecimal number received through UART and recording that one byte was received. Next, three more hex values are taken in and used to set the proper duty cycle on the red, green, and blue LEDs in the RGB LED. Using a case statement to select the proper byte for each color, the hex values corresponding to the duty cycle of each LED are stored in CCR registers CCR1, CCR2, and CCR3. Each value is then divided by the period of 255 Hz stored in CCR0 and the end result is a color that matches or is very close to the color that was intended. Next, the remaining hex values in the sequence are transmitted to the next board until the last hex value in the sequence is reached. When and if the last value in the sequence, 0x0D, is reached, the number of bytes received resets to zero and the program terminates.

4.22 UART Communication (Realterm)

Using a UART cable and the software package Realterm, we were able to successfully demonstrate that our code worked as designed. The first test involved confirming that the program successfully removed the bytes necessary to create one color and then sent the remaining bytes on to the next board with an additional byte that says the correct number of bytes being transmitted. A confirmation of this is shown in the figure below. Note that green lettering in Realterm indicates signals received and yellow lettering indicates signals transmitted. As the figure shows, eight bytes were taken in, while only 5 were sent out. The missing three bytes were successfully used to power the RGB LED.


RealTerm: Serial Capture Program 2.0.0.70

```
08 22 8B 22 80 11 42 0D 05 80 11 42 0D
```

Send: 0x08 0x22 0x8B 0x22 0x80 0x11

Dump File to Port: c:\temp\capture.txt

Repeats 1 · Delays 0 0 · Repeats 1 0

Char Count:5 · CPS:0 · Port: 17 9600 8N1 None

Figure 6. Realterm Communication Sequence

Now that the code has been confirmed to work as designed, several colors were produced on the RGB LED starting with a violet color given by 0x33 0x00 and 0x66.

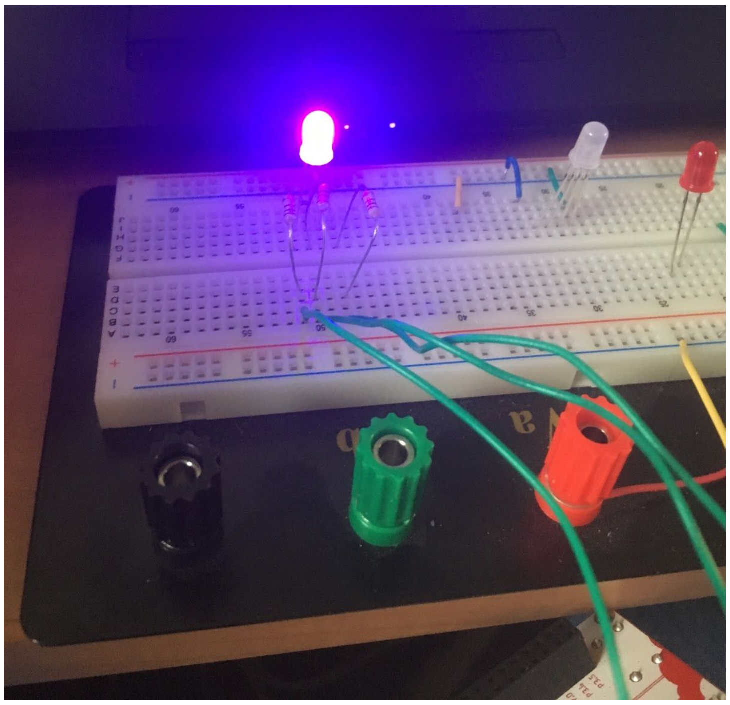


Figure 7. Violet LED Color

Next, a green color given by 0x4C 0x99 0x00:

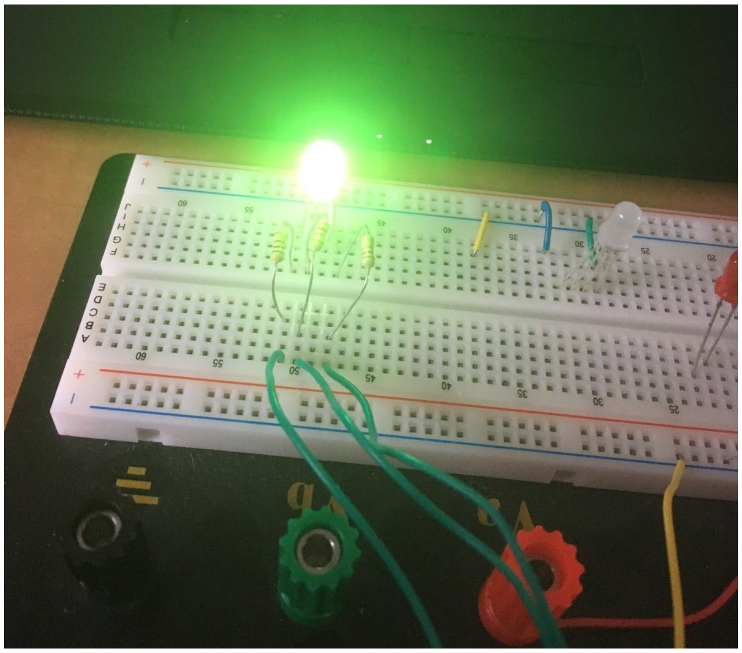


Figure 8. Spring Green Color

And finally a magenta color given by 0xCC 0x00 0x66:

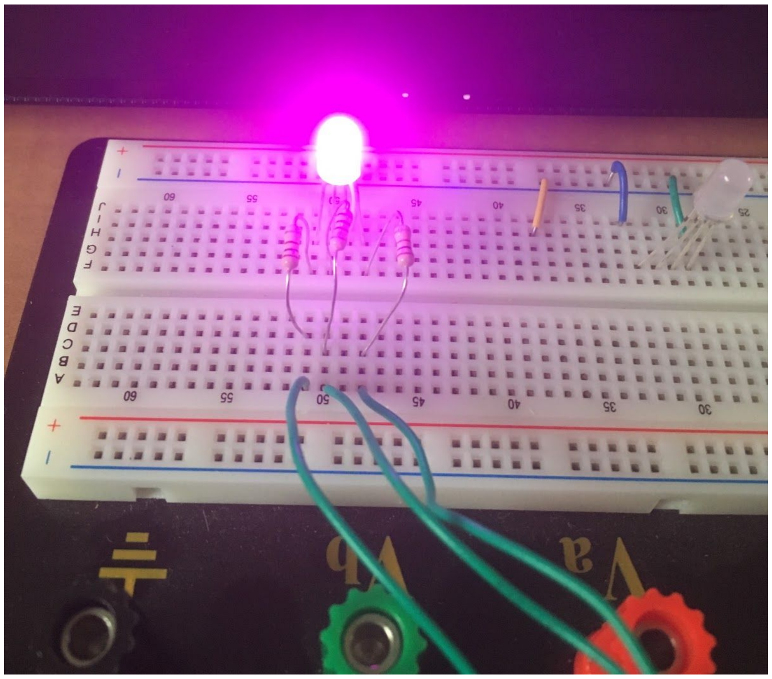


Figure 9. Magenta Color LED

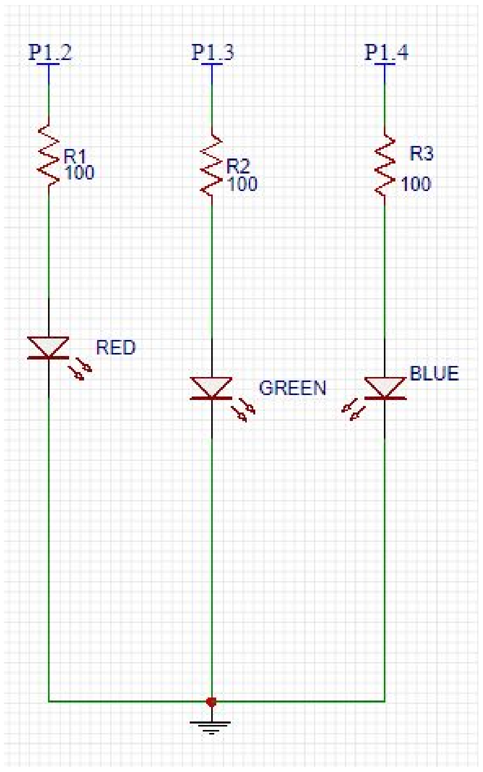


Figure 10. Circuit Diagram of RGB LED

Figure 10 shows design for the RGB LED. The Pins on the MSP430F5529 are specified at the top representing the specific input signal. The signal then passes through a current limiting resistor of 100 Ohms and then powers the individual LED colors. The longest pin on the RGB LED gets connected directly to ground.

5 Conclusion

A common cathode RGB LED allows for different pins on the MSP430F5229 that relate to the PWM outputs to be wired to the input pin of each color on the LED. The longest pin, the common pin, gets connected to ground. To limit the current going through the LED, a 100 Ohm resistor was put in series between each LED and the input from the F5229. After setting up our RGB LED, UART was tested to successfully send and a receive a signal to the LED.