ECE Embedded Systems: Milestone

Milestone 1: Communicating with Will Byers

Mathew Philippou, *Simon Bubbles* and *Michael Lonetto*
Rowan University

1 Abstract

The goal of this milestone is to generate an RGB node using a MSP430 microcontroller and linked it up in series with other nodes to generate different LED patterns. The RGB nodes will take in code through UART which is sent from a master node, then pass the code along to the rest of the node chain.

2 Introduction

One might ask, what would be the purpose of lighting up an RGB node chain? If you have ever seen the scene from Netflix's "Stranger Things", where Will Byers uses Christmas lights to communicate with his mother you would understand. The mother sets up the Christmas lights like a Ouija Board to communicate back and forth with Will Byers who is stuck in a parallel dimension. This scene can essentially be replicated using the RGB nodes generated in this milestone.

The end goal of the milestone can be seen in Figure 1. The "Master Node" will send a string of Hex values to the first node in the chain. The node will then take the 3 least significant bytes of the string and then transmit the remainder of the string over the UART TX line to the next node in the chain. The chain of bytes will continue to move down the chain until there are no more bytes for the nodes to read.

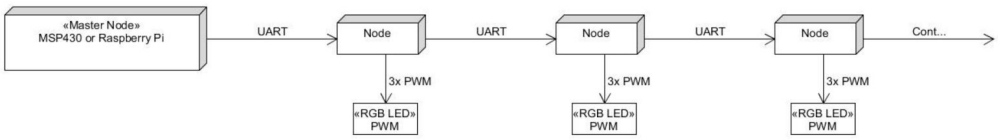

Figure 1: Network of Devices

Taking a closer look at each individual node in Figure 2, after the node receives the 3 least significant bytes it uses these bytes as its own RGB values. In order to replicate a color, these RGB values need to be converted and transformed into Duty Cycles that correspond to a specific color.

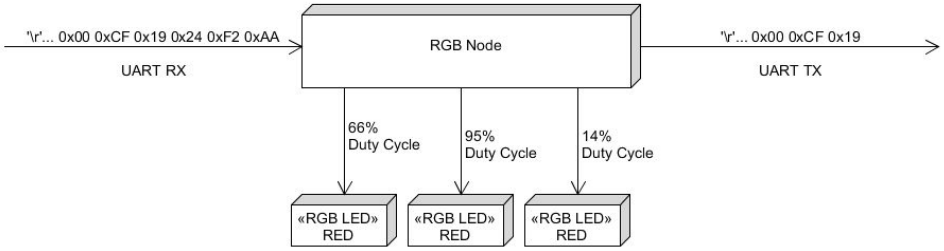

Figure 2: RGB Node Inputs and Outputs

Upon startup of the RGB Node the initialization process seen in Figure 3 needs to take place. During this stage the node needs to initialize all its timers and peripherals and enter Low Power Mode while it waits for a message over UART. Once a message is received, the 3 least significant bytes should get stored in a RGB Buffer which will be later used to set the duty cycles for the RGB LED. The rest of the message is sent to

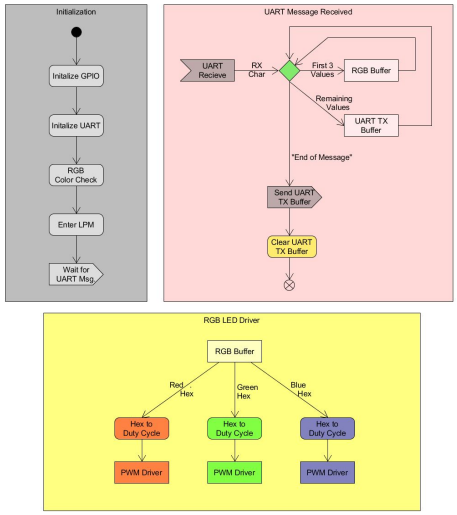

Figure 3: Flow of Code

UART TX Buffer, which transmits it over to the next node. The 3 bytes in the RGB Buffer converts the Hex values to a Duty Cycle for each of the three colors (Red ,Blue ,Green), driving the PWM and emitting the desired color.

3 Background

Why the MSP4305529?

We chose the MSP430F5529 to implement the milestone project because it had three CCR registers to hold the three PWM duty cycles we needed at the same time. None of the other boards would have been as convenient to implement.

UART

In order to control the LED using the 5529, a package needs to be received through UART. A UART connection was set up inside code composer through the software by initializing UCA1 or

the second available UART connection on the 5529. Below is the code used to Initialize Uart that was supplied to us and altered slightly.

```
volatile unsigned int total_bytes = 0;


void initializeUART(void)       // from Lab 1 example code
{
    P4SEL |= BIT4;          // UART TX
    P4SEL |= BIT5;          // UART RX
    UCA1CTL1 |= UCSWRST;    // Resets state machine
    UCA1CTL1 |= UCSSEL_2;   // SMCLK
    UCA1BR0 = 6;            // 9600 Baud Rate
    UCA1BR1 = 0;            // 9600 Baud Rate
    UCA1MCTL |= UCBRS_0;    // Modulation
    UCA1MCTL |= UCBRF_13;   // Modulation
    UCA1MCTL |= UCOS16;     // Modulation
    UCA1CTL1 &= ~UCSWRST;   // Initializes the state machine
    UCA1IE |= UCRXIE;       // Enables USCI_A0 RX Interrupt

}
```

Figure 4: Code to Initialize Uart

The UART TX connection was set to P4.4 and the RX connection was set to P4.5. These are the connection that could be used if a UART cable was available. Instead, in our case, the headers for the TXD and RXD connections on the 5529 were just removed so that they may be used for connection.

PWM

Pulse Width Modulation was used to implement different colors using the RGB LED. Pulse width modulation is controlling the widths of a pulse using a duty cycle. The duty cycle is how long the signal will remain high for the given period of the pulse. By manipulating the percentages of red, green, and blue in the LED we can make the LED virtually any color.

```
void Set_PWM(void){
    TA0CCR0 |= 225;
    TA0CCR1 |= 0;
    TA0CCR2 |= 0;
    TA0CCR3 |= 0;
    TA0CCTL1 |= OUTMOD_7;
    TA0CCTL2 |= OUTMOD_7;
    TA0CCTL3 |= OUTMOD_7;
    TA0CTL |= MC_1 + TACLR + TASSEL_2;
}
```

Figure 5: Code to Initialize PWM

The CCR control registers were set into reset/set mode in order to hold the duty cycle values for each color in the RGB LED. We set the period to 225 using the line "TA0CCR0 = 225;". All that was left to do was set the clock to 1MHz using the SMCLK and in up mode to finish implementing.

RGB LED

An RGB LED is actually 3 LEDs, red, green, and blue inside one package. For this milestone a common anode RGB is used which can be seen in Figure 6. The RGB has four pins one for each color and a common anode connected to 5V VCC. The common anode RGB works by turning an LED on if a low signal is received and turns off if a high signal is received. Using three individual PWMs, the duty cycle of each LED can be changed to produce any color desired.

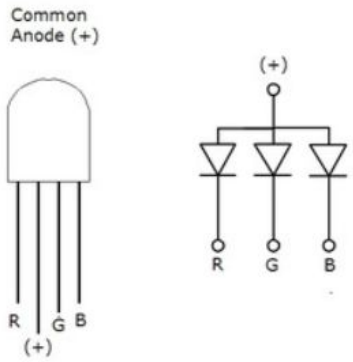

Figure 6: Common Anode LED

The wiring for the RGB LED can be seen in Figure 7. Each color of the LED is connected to an NMOS with a 1k Ohm resistor which limits the current. From the NMOS each color is connected to a different pin on the the MSP430F5529. The red LED is connected to P1.2, green LED is connected to P1.3, and the blue LED is connected to P1.4.

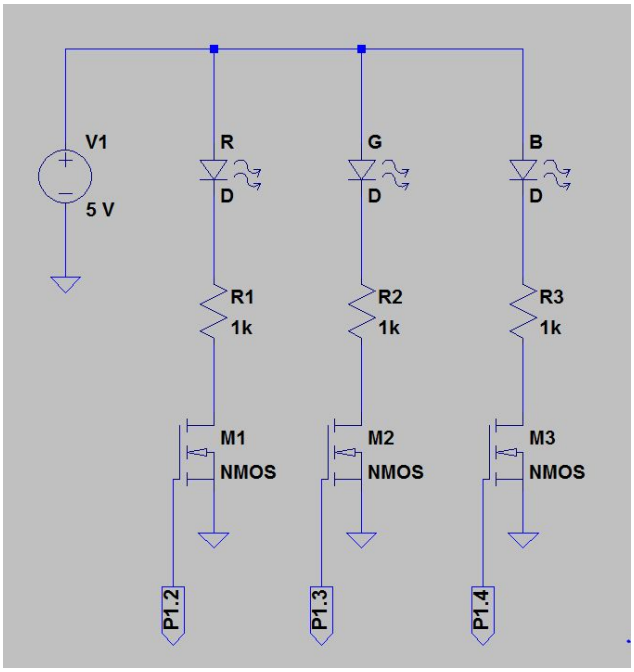

Figure 7: Schematic of Breadboard

4 Evaluation and Results

RGB LED

In order to ensure the code was working properly the program RealTerm was utilized. By entering hexadecimal characters into the command window of RealTerm, bytes were sent to the board, represented by the numbers shown in green. The first number denotes how many bytes are in the package. The next three numbers are the values being sent to the pins of the LED. Any numbers remaining, those in yellow, get sent back out (hypothetically to the next board), along with an initial number that will denote how many bytes are in the next package. The figures below show three different strings of numbers being sent to the 5529 and how the LED changed color for each.

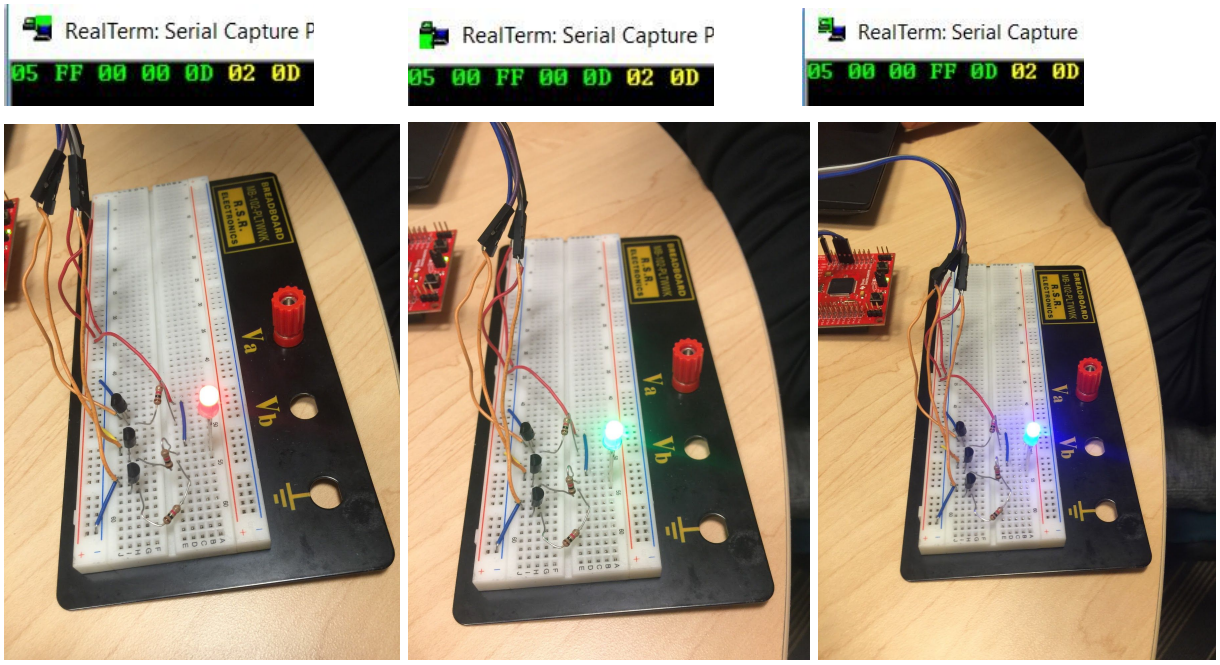

Figure 8: RealTerm Packages and Corresponding LED Outputs Below it

FF is the highest value that may be sent to an individual LED out pin of the 5529. By setting the first value to FF and the other two to zero we can effectively turn the duty cycle of the red component of the RGB LED up to 100% and the rest to 0% so that we may only see red. The same process was used for green and blue in the following two pictures.

5 Discussion and Conclusion

The goal of this milestone was to generate an RGB node using a MSP430 microcontroller and linked it up in series with other nodes to generate different LED patterns. To accomplish the goal an MSP430F5529 was chosen as the microcontroller of choice due to its convenience. Using the the microcontroller, it was able to receive a string of hex values to then convert over to duty cycles for each of the 3 LEDs. The duty cycles of each LED ultimately determine the final color of the RGB.