Milestone 1: Stranger Things

Thomas Haskell, Rajinder Parhar
Introduction to Embedded Systems: Section I
October 18, 2018
haskellt8@students.rowan.edu; parharr4@students.rowan.edu

I. Abstract

In this project we attempt to recreate an inter-dimensional communication. In "Stranger Things" they strung up lights with letters corresponding to each light, one light would blink at a time to display a message over a period of time. This project explores the idea of replicating this idea by using a microprocessor to control an addressable LED that will blink accordingly when sent the proper hex code. In this instance a MSPF5529 micro-controller will be used and connected in series with other microprocessors to perform the same task demonstrated in the show.

II. Introduction/Background

The main concern with this lab is to be able to create a program that can be launched on a MSP430 processor to control a RGB LED based on some input. Core concepts used in this project include Universal Asynchronous Receiver/Transmitter (UART), TIMERS, Pulse with Modulation (PWM), and General Purpose Input/Output (GPIO).The "Master Node" is responsible for sending out the signal in HEX values to the first node as shown in figure 1. However depending on the amount of nodes within the system the node can receive up to three bytes times the amount of nodes plus two addition bytes to equal the total amount bytes from the master node. Each node will reduce the size of the packet by 3 bytes, and each node will have the responsibility to confirm that they took three bytes by editing the second to last set of bytes. For the last byte the program should just keep track of the remaining bytes in the program. Than transfer the set of bytes of the UART TX line to the next node where it will repeat the process.

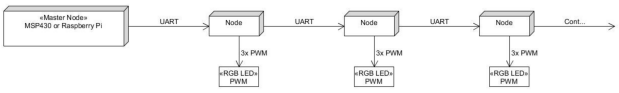

Fig. 1. Network of Devices

To control the RGB LED node a 3 byte message should be decoded as seen in figure 2. This node should take the 3 least significant bits of data which tell the different RGB values and form a specific color at the corresponding duty cycle.

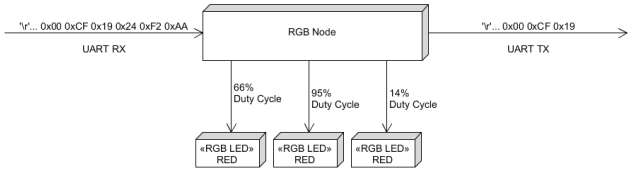

Fig. 2. Enhanced view of the RGB network

Shown in figure 3 it can be demonstrated that within the start up process all the timer and peripherals need to be launched and enter lower power mode(LPM). While in LPM the node waits for a message that would be received over UART RX line. After the receiving the message it should run the program and form a buffer for the UART TX line that is waiting to send the set of information to the next microprocessor.

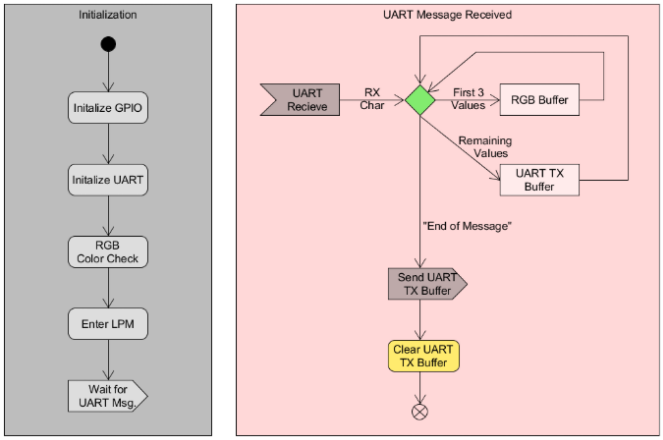

Fig. 3. Initialization process of the RGB Node

III. Implementation

A. Processor Selection

The processor selected was the MSP430F5529 due it having five Capture Compare Register(CCR) that can be used. The more CCRs available made it much more easier to implement the LED duty cycles and only timer needs to be made for the PWM. The MSP430g2553 and MSP430fr2311 weren't used due to the fact it only has 3 CCRs meaning they wouldn't allow the user to implement using two timers instead of one

making it more difficult to program. The MSP430p401r wasn't chosen not to due to the fact of availability of CCRs or other issues but in fact was a preference choice.

B. Pulse with Modulation (PWM)

Pulse with modulation is used to control the duty cycles for the red, blue, and green diodes to produce a specific colors and brightness. Having the MSP430F5529 allows us to only need to implement one clock since each clock contains enough CCRs to implement the three different colors and the software PWM. The clock chosen was the SMCLK in up/down mode with an input of divider of 2, which we can determine the result to be 0xFF or 255.

```
void PWMSetup(void)
{
    TA0CCR0 = 0xFF; // reset register
    TA0CCR1 = 0xFF; // red
    TA0CCR2 = 0xFF; // green
    TA0CCR3 = 0xFF; // blue

    TA0CCTL1 = OUTMOD_7; // set/reset mode
    TA0CCTL2 = OUTMOD_7;
    TA0CCTL3 = OUTMOD_7;

    TA0CTL = MC_3 | TASSEL_2 | ID_1; // up down | SMCLK | 2^1 division
}
```

C. State Machine

Within the interrupt we establish a switch statement that would control the behavior of the program for each individual case of HEX codes that are received. The first thing received by the processor is the byte length which is handled in the first state. In the first state we take the byte length and subtract 3 than push it along. The reason no operation is implemented in this case is to prevent the processor from missing data while it is sending data. The second to fourth state we have the three different LEDs being handled by the incoming bytes. In the final state which transmits the bytes for the boards down the line.

```
            switch(byteCount)
            {
                case 0:
                    while(!(UCA0IFG & UCTXIFG)); // wait while transmitting
                    byteLn = UCA0RXBUF; // byte length
                    UCA0TXBUF = byteLn - 0x03; // transmit all but first 3 bytes
                    __no_operation();
                    break;
                case 1:
                TA0CCR1 = (0xFF ^ UCA0RXBUF); // red
                    break;
                case 2:
                TA0CCR2 = (0xFF ^ UCA0RXBUF); // green
                    break;
                case 3:
                TA0CCR3 = (0xFF ^ UCA0RXBUF); // blue
                    break;
                default:
                    while(!(UCA0IFG & UCTXIFG));
                    UCA0TXBUF = UCA0RXBUF; // Just transmit the incoming byte on to the next board
                    __no_operation();
                    break;
            }
```

D. Universal Asynchronous Reciever/Transmitter (UART)

The UART design is mainly based of the code given to us in Lab0. In this instance we re-purposed the code given to us by mainly readjusting the baud rate and modulation that was in the sample code. The baud rate was adjusted to match the speed at which data is transferred on other boards. 9600 Baud rate is translated to 9600 bits per second transferred.

```
void UARTSetup(void)
{
    P3SEL |= (BIT4|BIT3); //RX|TX peripheral mode
    UCA0CTL1 |= UCSWRST; // reset state machine
    UCA0CTL1 |= UCSSEL_2; // SMCLK
    UCA0BR0 = 6; // 9600 baud
    UCA0BR1 = 0;
    UCA0MCTL |= UCBRS_0|UCBRF_13|UCOS16; // modulation UCBRSx=0, UCBRFx=0
    UCA0CTL1 &= ~UCSWRST; // initialize state machine
    UCA0IE |= UCRXIE; // enable UART interrupt
}
```

E. Off-Board

This schematic what should be designed off board for the RGB LED, each output to an LED would be connected to each correlating cathode as shown in the in figure 4 or 5. This can be varied depending on the type of RGB led that is used but for the purpose of this, a common anode RGB LED is used. Apart from the LED, we use three 2N6907

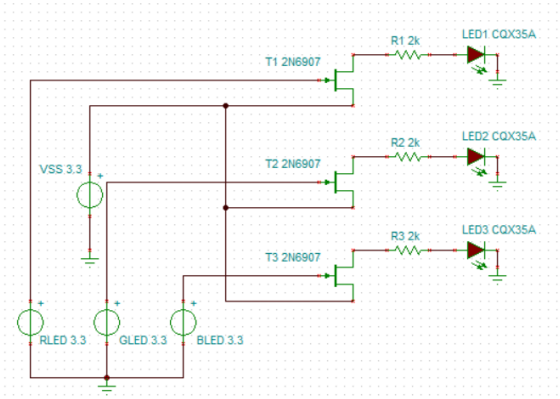

Fig. 4. Schematic design for the off board LED

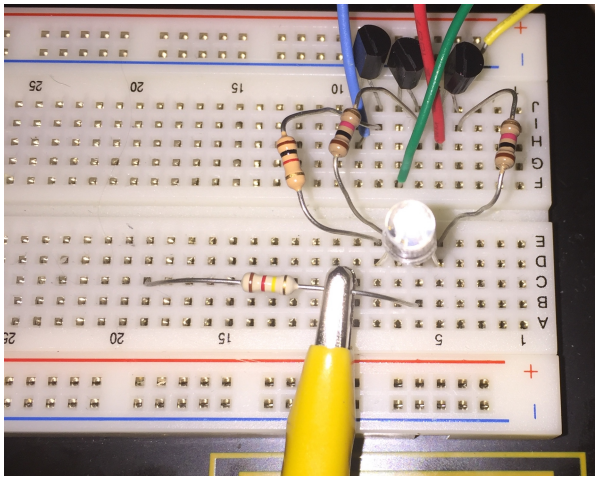

Fig. 5. Breadboard RGB LED Prototype layout

Next would be connecting the point for red, green, and blue with the MSP430F5529. Red correlates with P1.2 out, green is connected with P2.3 out, and red is connected with P1.4 out. Power can be pulled from the microprocessor board from the 3.3v out pin, and serial communication can be established by connecting the RXD pin to P3.4 and TXD pin to P3.3 on the micro processor board. All these connections are shown in figure 6. These are the steps to replicate the design once the code has been implemented.

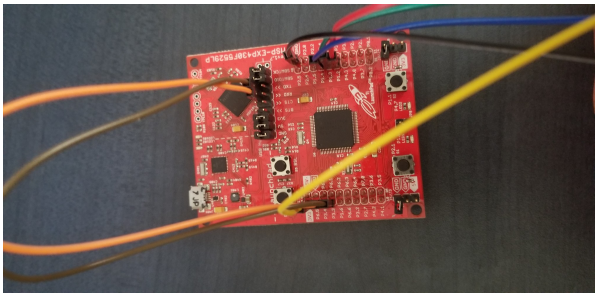

Fig. 6. Layout of the connections needed for the board to function.

IV. Conclusion

The end of result of the milestone was to generate a RGB node using a MSP430 microprocessor to be linked up in series with other microprocessor to form the whole alphabet. However this project wasn't put into actual practice by being linked in series with 25 other microprocessors, through the self conducted tests it was shown that the microprocessor would work with others processors in a series linked together as long as the code and process is consistent among all the other processors. However it is achievable it is not something that is practical considering the user would need to obtain 25 other microprocessors or potentially 13 total if the user were to connect 2 RGB Leds to the microprocessor. The real purpose of the project is to show how UART and PWMs can be used together to show one of the many applications.