Milestone 1: Stranger Things Light Wall - Hardly Working

Cameron Korzeniowski & Jacob Okun
Rowan University

October 22, 2018

1 Design Overview

The intent of this project was to create a single RGB LED node using an MSP430 microcontroller. This RGB node also needed to work in series with other MSP430 RGB nodes communicating via UART. For this purpose, an MSP430F5529 was utilized. The F5529 RGB node created receives a string of hexadecimal values via UART, utilizes the three least significant bits to control the RGB color and brightness, and passes the remaining bits of the hexadecimal string onto the next RGB node in the chain.

1.1 Design Features

These are the design features:

- Able to receive and execute UART signals
- Able to pass on excess UART bits to subsequent RBG nodes
- Change RGB LED Color
- Change brightness of individual LED colors
- Reset button turns off RGB LED

1.2 Featured Applications

- Stranger Things LED Light Wall
- Decorative variable color lighting

1.3 Design Resources

Here's a link to our code.

1.4 Block Diagram

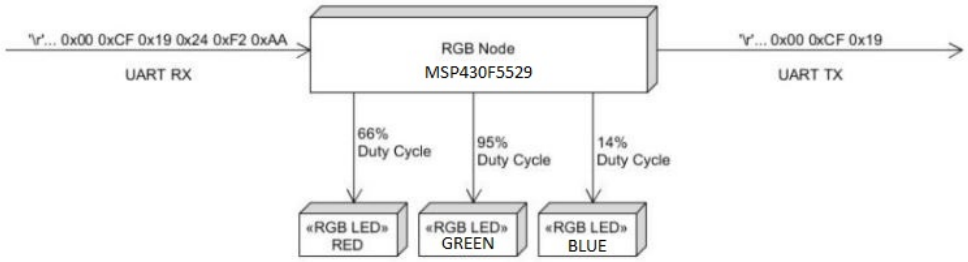

Figure 1: System Block Diagram

1.5 Board Image

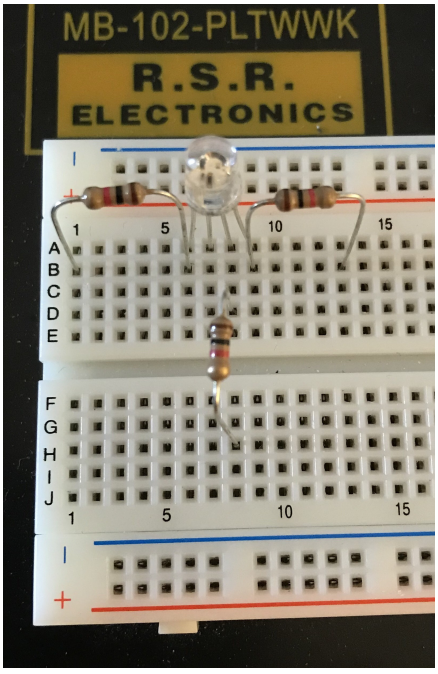

Figure 2: RGB LED with 1000 ohm resistors

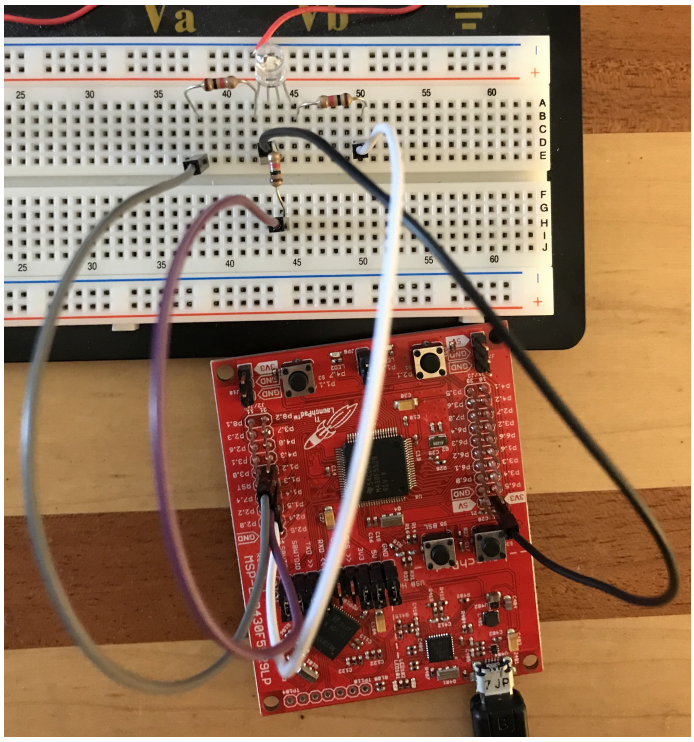

Figure 3: RGB LED connected to MSP430F5529

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Byte Size	0xXX	This sets the byte size of the message sent over UART, changing XX to 03 sends a 3 byte message
Color Code	0xFF 0x00 0x00	Illuminates the LED red
Color Code	0x00 0xFF 0x00	Illuminates the LED green
Color Code	0x00 0x00 0xFF	Illuminates the LED blue
Color Code	0xFF 0xFF 0xFF	Illuminates the LED white
Color Code	0x00 0xFF 0xFF	Illuminates the LED teal

3 System Description

For this project, a program was created that could be loaded on to an MSP430 processor to create an RGB LED node. For this purpose, timers, PWM, and UART protocol

were utilized. The signal sent over UART includes a length byte, three bytes to control the red, green, and blue of the RGB LED, and an indeterminate number of bytes to be passed onto the next RGB node. An example of this is 0x06 0xFF 0x00 0x00 0x00 0xFF 0x00 which would send a 6 byte message, turning on red, and leaving green and blue off on the first node, and passing a green-only signal to the next node. The board that the program was implemented for was the MSP430F5529.

3.1 Detailed Block Diagram

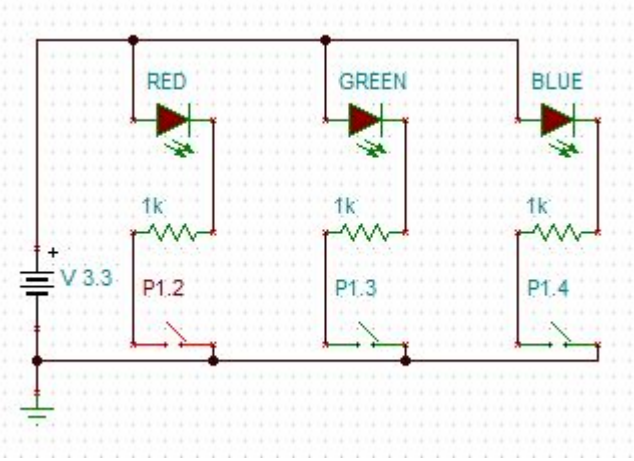

Figure 4: RGB LED Diagram

3.2 Highlighted Devices

- Computer with Realterm - Generates and sends original UART signal to the first MSP430 RGB node
- MSP430F5529 - Receives UART signal, illuminates RGB LED accordingly, passes on remaining UART bits.

3.3 MSP430F5529 Processor

The MSP430F5529 microprocessor is the heart of the RGB LED node. It is responsible for receiving, processing, and sending the UART signal that determines the RGB LED color and brightness. The RGB LEDs are configured as common anode, so the microprocessor provides switched ground to the red, green, and blue LEDs of the RGB. The specific pins used are P1.2 for red, P1.3 for green, and P1.4 for blue. These three pins were selected because they are by default PWM controlled inputs/outputs on the MSP430F5529. The rate at which these PWM pins modulate is determined by the values stored in capture/compare registers 1 (for red), 2 (for green), and 3 (for

blue) of Timer A0. These capture/compare registers receive their values from the 2nd, 3rd, and 4th bytes of the UART signal received at RX. Once the processor has read the 3 bits necessary to determine its node's LED color and brightness, it sends the remaining UART bits (less the 3 it used) out to the next node via TX.

3.4 RGB LED and Circuitry

Our Embedded Systems class as a whole decided to designate all RGB LEDs as common anode. Common anode means that the red, green, and blue LEDs of the RGB are supplied with a constant power (in our case 3.3 V from the MSP430F5529 development board), and modulated on the ground side. This modulation is performed by the MSP430F5529 processor as described above. The MSP430 processor can only handle small currents through its I/O pins, and since LEDs have little to no resistance, current regulation was necessary. This current regulation was accomplished by placing 1k ohm resistors in series between the red, green, and blue LED cathodes and the processor pins. The 1k resistors limit the current through each processor pin to 3.3 mA, which is well below their limits while still maintaining sufficient LED brightness.

4 SYSTEM DESIGN THEORY

The MSP430F5529 was picked over the other boards for multiple reasons. The F5529 was chosen due to it making UART communication easier by using a USB cable, and simpler implementation of the PWM thanks to pins P1.2, P1.3, P1.4 and their corresponding CCRs. The G2 doesn't have the 3 CCR registers necessary for an RGB LED, while the F5529 has five CCR's. The FR2311 only has timer B, and a UART USB back channel would have to be used.

4.1 PWM

PWM, or pulse width modulation, is used to control the duty cycles of the red, blue and green LEDs. By using PWM, the brightness can be changed for each LED to combine the colors together to produce a variety of colors. Because the MSP430F5529 was used, an additional timer module was not needed as Timer A0 has four capture/compare registers built into it. The timer clock was set to SMCLK in UP mode at 1 MHz. This sets the clock period to full cycle, and the additional capture/compare registers are set and reset for each color using OUTMOD_3. Because of this when the timer reaches a value that is set in one of the capture/compare registers it will be set high. The timer will go until it reaches the full cycle clock period of 256. When this happens the pin is reset due to overflow.

4.2 UART

UART stands for Universal Asynchronous Receiver/Transmitter. UART is needed to send and receive signals from/to the device. The values received by the processor

are bytes in hex value. To use UART, the pins associated with it must be enabled. On the MSP430F5529, those are pins 4.4 for transmit and 4.5 for receive. Next, the state machine is initially set to reset so it is not used till it is needed. Then the BAUD rate was set to 9600. The modulation is set after and these values are stored in corresponding registers. Once this has all been done, the state machine is ready to be initialized. The RX interrupt is enabled and the interrupt flags are cleared.

4.3 State Machine

The state machine is used to control the LED's PWM and pushes information to the processor. There are a total of five cases, four of them being cases 0-3 and one default case. The first case is used to set the total equal to the first byte of the UART signal. Then the next three cases all set the capture/control register to their respective byte of the UART signal. The last case also checks to make sure the TX buffer is ready and sets the first byte of the transmitted UART signal to the previous total minus 3. The default case is set to check that the TX buffer is ready and sets the TX buffer to the remaining bytes stored in the RX buffer. After each case happens the byte is incremented by 1, which selects the next case.

5 Getting Started/How to use the device

To use this program an MSP430F5529 must be obtained. The code is designed for this board and will not work with other boards due to different pin assignments and timer modules. To use this RGB node, multiple programs must be downloaded and installed and a basic understanding of using them is needed. A breadboard, resistors and an RGB LED are needed to build the circuit to be connected to the MSP430F5529. Once the circuit is constructed and connected UART can be used to send signals to the processor which will change the LED color and brightness.

5.1 Software needed

Texas Instruments' Code Composer Studio software is needed to load the firmware onto the MSP430F5529. This can be downloaded from TI's website. Once downloaded, follow the install wizard and be sure to install the package for the MSP430. Now the code can be input into Code Composer Studio and flashed to the board.

The other piece of software needed is Realterm. Realterm is a serial capture program that can be used for UART. Once this is downloaded and installed you are ready to connect to the MSP430F5529. First plug the board into the computer using a USB cable. The port that the device is connected to must be determined, which can be done by going into Device Manager and checking Ports (COM & LPT). The one needed is MSP Application UART1 (PortXX), in our case it was port 6. Once this information is obtained, go to the port tab of RealTerm, set the BAUD rate to 9600, select your port, set it to open, and click change. Once done, the processor is ready to receive messages over UART.

5.2 Circuit Setup

A circuit must be built on a breadboard for the RGB LED. A breadboard, three 1000 ohm resistors, an RGB LED, and connecting cables are needed. Figure 4 in Section 3.1 shows the block diagram for the circuit that must be built. Power and ground will come from the MSP430F5529 as shown in figure 3.

5.3 Communicating with the Device

To communicate with the LED, the MSP430F5529 processor must be connected to the RGB LED circuit. P1.2 of the development board is used for red, P1.3 is used for green, P1.4 is used for blue, and 3.3 V will go to the anode pin of the LED. Once wires are installed connecting these pins to the RGB LED, messages sent over UART will instruct the processor to interact with the LED. To send these messages, go to the ”Send” tab in Realterm. There is a box next to ”Send Numbers” where you will type the hex bytes for the message to be sent. The first bye is the message size, with the next three bytes determining the LED color and brightness, in order of red, green, blue. For example: 0x03 0xFF 0xFF 0x00 will send a three byte message, enabling the red LED at full brightness, the green LED at full brightness, and the blue LED off. To communicate with multiple boards chained together, the byte size of the message must be changed accordingly. If one board is added, a 6 byte message would be necessary, so the first byte would be 0x06. To daisy-chain boards, the transfer pin from the main board must be connected to the receive pin of the following board. The two boards must also share a common ground. An additional LED circuit must be constructed and connected to the new board in the same manner. The steps for sending color changes are the same, but with more bytes, for example: 0x06 0xFF 0x00 0x00 0x00 0x00 0xFF will set the first node’s RGB LED to red and the second node’s RGB LED to blue.

6 Test Setup

To setup testing of the program, the code must first be flashed to the MSP430 processor. For this purpose we used Code Composer, and the specific board that was flashed was the MSP430F5529. Once the code is flashed to the board it is ready to be interfaced with.

To communicate with the board, UART is used. To accomplish this the program Realterm was used. Once Realterm is opened go into the port tab and set the port to the port the MSP430 board is on. For our case it was port 16. The BAUD rate must be set to 9600 and the port must be set to open. Once this is done we are ready to send information to the board.

To send information to the board to change the color of the LED we must go to the Send tab. To change the LED color we need to send a 3 byte message. The first byte we would send is ”0x03” which is the size of the message, in this case 3 bytes, then the color for red, green, and blue respectively. An example message is ”0x03 0xFF

0x00 0x00", which sets red to full brightness, and green and blue to zero. Once send is pressed on Realterm the LED should change to whatever it is set to.

6.1 Test Data

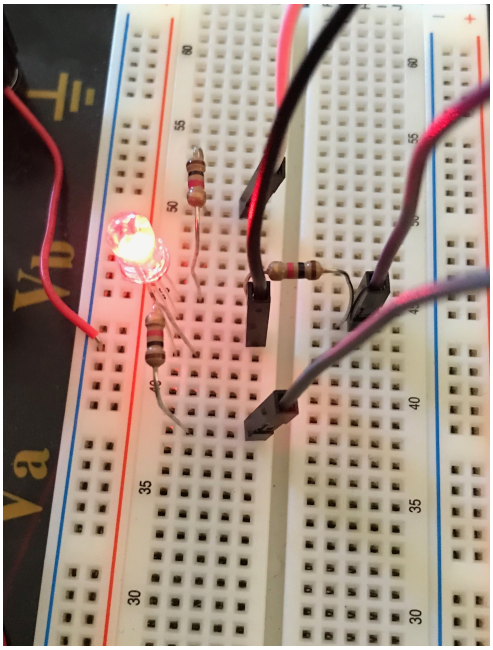

Figure 5: RGB LED Illuminated Red

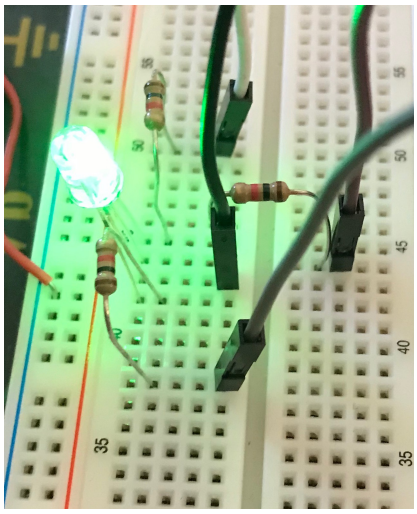

Figure 6: RGB LED Illuminated Green

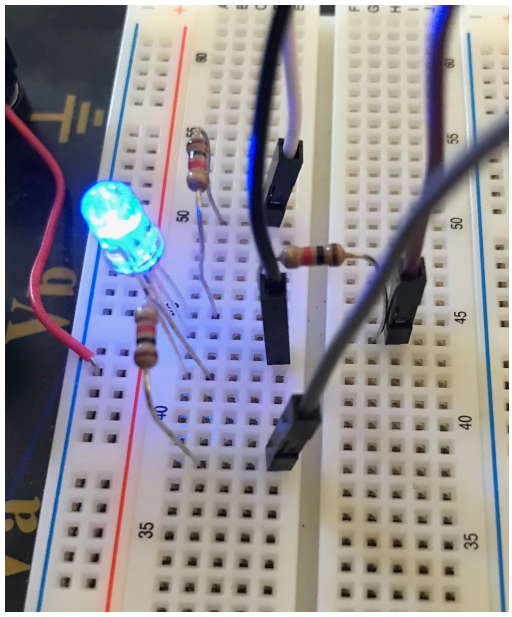

Figure 7: RGB LED Illuminated Blue

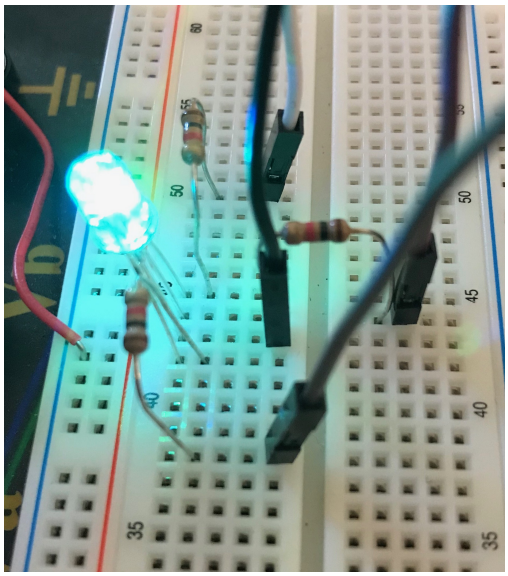

Figure 8: RGB LED Illuminated Teal

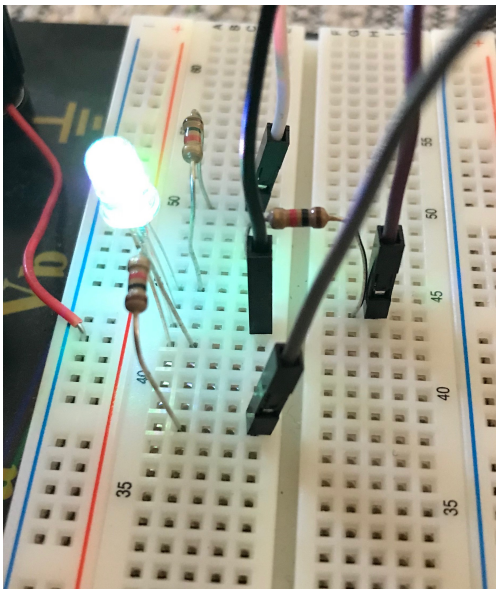

Figure 9: RGB LED Illuminated White