AN-Milestone 1: Stranger Things Light Wall

Nick Scamardi and Nick Setaro

Rowan University

October 22, 2018

1 Design Overview

An addressable RGB LED was constructed which has the ability to produce various colors by alternating the duty cycle of the Red, Green, and Blue components. By changing the duty cycle (brightness) of each node, the 3 main colors could be combined to produce a wide variety of others. Communication with the RGB LED was accomplished using the MSP430F5529 micro-controller along with its UART features. The LED built for this milestone possesses the ability to communicate with others using the RX and TX UART channels given that the Baud Rate (9600) was consistent across all of the components. The overall functionality of the LED includes the ability to receive a package of bytes (any size), take the first three bytes to set the color of the LED, and then pass on the rest of the package so that the next LED can set its color.

1.1 Design Features

There were several required design features implemented in Milestone 1.

These design features include:

- Receive a random package and adjust color of the LED accordingly.
- Transmit an updated version of the package to another LED.
- UART TX and RX Communication using BAUD of 9600.
- Adjust brightness/color by varying the duty cycles.
- PWM with Timer Interface.

1.2 Featured Applications

The featured applications of this device include:

- Retrieving Will Byers from the Upside Down
- Variable Addressable LED Lighting (ie. String Lights)
- Multicolor Indicator LED (ie. Indicates Status of Device)
- Cross-Device Compatible (BAUD 9600)

1.3 Design Resources

The design components (code and README) can be accessed using the following link. The Milestone1.c contains the C code for the milestone.

`https://github.com/RU09342-F18/milestone-1-arduino-al-pacino/tree/master/Milestone_StrangerThings`

1.4 Block Diagram

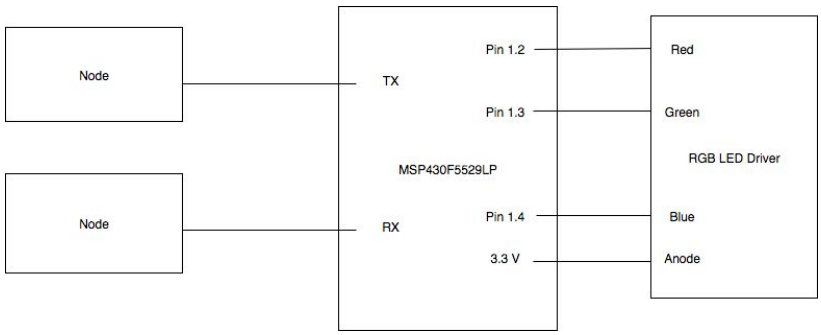

Figure 1: Block Diagram

The block diagram shown in figure 1 shows the configuration of the system. The two main components are the MSP430F5529LP microcontroller and the RGB LED Driver circuit. When chaining the system to another one the TX port of the first system would be connected to the RX of the next.

1.5 Board Image

The RGB LED used was common anode. The 3.3 v output from the MSP430F5529 was connected to the anode of the LED. The output pins P1.2, P1.3, and P1.4 were connected to the Red, Green, and Blue nodes respectively. 300 Ohm resistors were used to connect each color node to the output pins in order to limit the current. The board image is displayed in Figure 2.

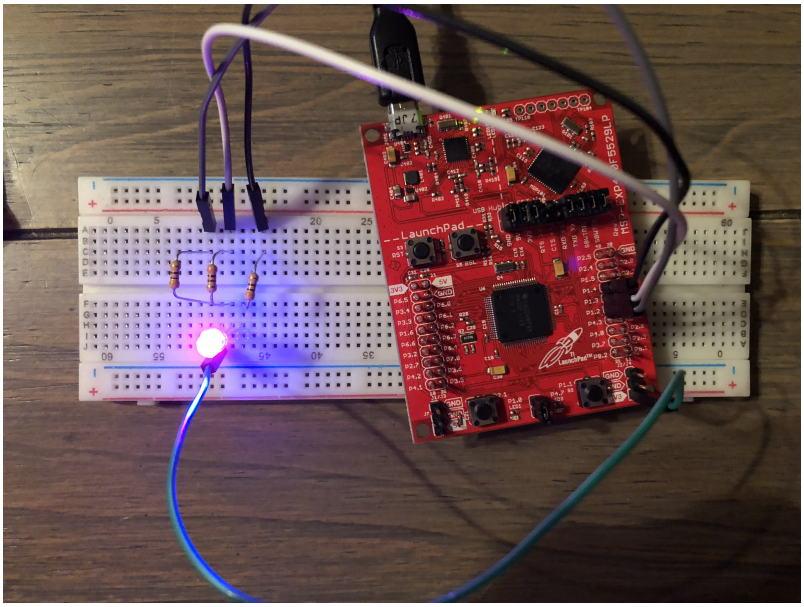

Figure 2: Breadboard Configuration of RGB LED and MSP430F5529

2 Key System Specifications

PARAMETER	SPECIFICATION	DETAILS
BAUD	BAUD 9600	BAUD rate of 9600 (9600 bits/sec). Used for UART communication.
Receiver	RX Communication	Receive package of bytes using UART RX channel.
Transmitter	TX Communication	Transmit package of bytes using UART TX channel.
Package Size	Package Adjustment	Adjust package size to remove bytes that were used previously.

3 System Description

The main purpose of this system was to configure an RGB LED in an addressable manner which allowed it to receive and transmit packages. Rather than hard-coding the RGB colors, the objective was to utilize UART communication in receiving byte packages as well as interfacing with similar systems. The ultimate end goal was an addressable LED string, each one emitting a specific color based on the bytes received. The package format is described below:

- **Byte 0**: Total Package Size (ie. 0x03: 3 total bytes)
- **Byte 1**: Red Duty Cycle (ie. 0xFF: Full brightness)
- **Byte 2**: Green Duty Cycle (ie. 0x7F: Half Brightness)
- **Byte 3**: Blue Duty Cycle (ie. 0x00: OFF)

The values entered are in hexadecimal. For a larger package, the same pattern is followed with Red, Green, and Blue occurring in the same order.

3.1 Detailed Block Diagram

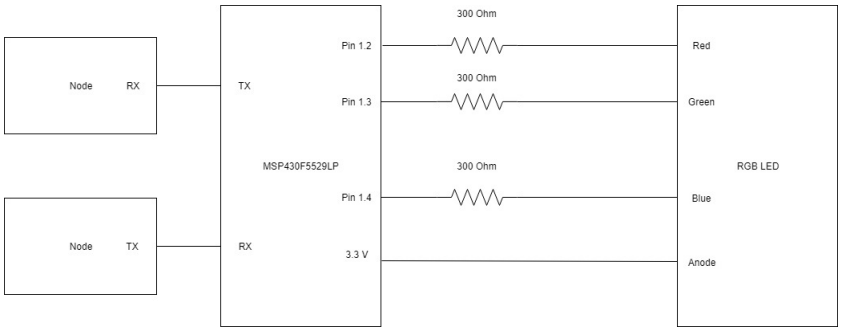

Figure 3: Detailed Block Diagram

Figure 3 shows the detailed block diagram for the system. This diagram shows the circuitry required to interface with the LED as well as the connections to other nodes.

3.2 Highlighted Devices

There are two main active components/devices used for the milestone.

- **MSP430F5529**: Used to power the the LED, Control the duty cycle of each color node, and UART communication.

- **RGB LED**: Used to display specific color based on the received package of bytes.

3.3 Device/IC 1: MSP430F5529

The main component behind this design is the MSP430F5529 micro-controller and its on-board processor. The pins used to interface with the off-board LED were P1.2, P1.3, P1.4, P4.4, P4.5, and the 3.3 V output pin. The three pins, P1.2, P1.3, and P1.4, are general I/O pins and can be used with the capture/compare registers CCR1, CCR2, and CCR3 respectively. They also utilize Timer A0 which is set to use OUTMOD3. OUTMOD3 is the set/reset mode which sets when the value in the capture compare register is reached and resets when the value in CCR0 is reached. P4.4 is used to transmit data (TX) in the USCI_A1 UART mode and P4.5 is used to receive data (RX). SMCLK was used for both UART and Timer A0. This clock runs at approximately 1 MHz and was used without an internal divider. For receiving data, the Receive Buffer register UCA1RXBUF was used to store the incoming package. For transmitting data, the Transmit Buffer register UCA1TXBUF was used.

3.4 Device/IC 2: RGB LED Circuitry

Since the RGB LED is of the common anode form, the 3.3 V pin was connected directly to the anode. This provided a common voltage to power the LED. As for the Red, Green, and Blue leads, the output pins P1.2, P1.3, and P1.4 were used to power them. 300 Ohm current-limiting resistors were used on each lead to connect with the output pins. Given that the output voltage is in the range of 3.3 V, this limited the current to a little over 10 mA. This current value is well within the operating range of the LED while still providing enough brightness it to be visible. This completes the standalone circuitry for the RGB LED.

4 System Design Theory

The overall design operation includes receiving a package of bytes, setting the duty cycle of each color based on the package, removing the used bytes, and transmitting the rest of the package. The minimum package size is 4 bytes, with the first one being used to keep track of the size and the remaining three for setting the duty cycle. The maximum value for each byte is 0xFF or 255. This provides the maximum duty cycle which is equivalent to maximum brightness.

4.1 Design Requirement 1: Receive a Package

The package is received in one of two ways; through a program such a RealTerm via USB or through the transmit (TX) channel of the previous board in the chain. If the package is being sent directly from the computer using RealTerm, the RX channel does not need to be used. Instead, the bytes are input and sent directly to the processor which controls the LED. If the board is not the first in the chain, the TX pin from the previous board must be connected to the RX of this board. Regardless of where the data is coming from, the bytes received are stored in the the receive-buffer register UCA1RXBUF. The Timer A0 capture/compare registers are then set to hold one byte each from UCA1RXBUF in order to set the duty cycles.

4.2 Design Requirement 2: Update RGB Colors

The RGB LED is driven using three PWM outputs from the microcontroller. When a packet is sent to the microcontroller bytes 1, 2, and 3 set the PWM for the red, green, and blue inputs respectively. This is achieved by using a switch statement to decide which byte had just triggered the UART interrupt. Depending on which byte was identified the corresponding color was then set.

4.3 Design Requirement 3: Transmit Updated Package

In order to transmit the correct package, the bytes used must first be removed. Otherwise, instead of producing the next color in the sequence, the next LED would produce the same color as the previous one. After updating the capture/compare registers and producing a color on the LED, the total byte count is subtracted by three. After removing the used bytes, the UCA1TXBUF is set equal to the UCA1RXBUF, in order to send out the new package.

5 Getting Started/How to use the Device

This section instructs the user on how to properly configure and use the device. The main focus is the circuit configuration, as well as connecting the microcontroller to the RGB LED circuitry. Interfacing this device with others will also be covered. The focus of this section is hardware-based, with the software side being covered in Section 6.

5.1 Configuring the Circuit

Use jumper wires to connect the pins on the microcontroller to the RGB LED circuit. The proper configuration is P1.2 to the red node, P1.3 to the green node, and P1.4

to the blue node. The 3.3 V pin should go to the common anode of the LED. When connecting the output pins to the color nodes, ensure the use of resistors to limit the current flowing through them. This will ensure that the LED does not burn out. 300 Ohm resistors were used, however, higher values can be used to provide a dimmer LED.

5.2 Interfacing with Other Devices

In order to interface this device with others in a ”daisy chain” fashion, the RX and TX channels on the microcontroller must be utilized. P4.4 and P4.5 are used as the TX and RX respectively. The first link in the chain can ignore the RX channel, as it will be receiving the package directly from RealTerm over USB. For the rest of the chain, the TX channel will be connected to the RX channel of the next board. A common ground will also need to be established between the boards which can be done by connecting the ground output pins.

5.3 Device Specific Information

For the MSP430F5529, the RX and TX pins have jumpers connecting the two sides of the board. In order to allow for TX and RX communication, the jumpers must be removed so that the jumper wires can be connected to the pins. The jumper wires should be connected to P4.4 and P4.5, which are the ones on the side where the TXD and RXD labels are.

6 Getting Started Software/Firmware

This section discusses the software side of this project and how to control the addressable LED using it. On the software end, two programs are needed to interface with the RGB LED.

- Code Composer Studio 8.1.0
- RealTerm or PuTTY

Code Composer is used to run the code and send it to the microcontroller. Once the code is flashed to the controller, code composer is no longer needed as the program will be stored on the microcontroller until removed or overwritten. As for RealTerm, this is used to communicate with the LED over the built-in UART channels. RealTerm allows for packages of bytes to be sent to the device, which will in-turn alter the duty cycles.

6.1 Executing the Code

In order to run the code, the first step is obtaining the Milestone1.c from the link provided in Section 1.3. Once the code is inside Code Composer, execute the Debug function. After debugging successfully without errors, press the play/resume button to flash it to the board. This completes the code execution, as it is now running on the microcontroller.

6.2 Communicating with the Device

This portion uses RealTerm to communicate with the device. Setup RealTerm by selecting the correct port for UART, specifying a BAUD rate (9600), and opening the port. After setup, the device is ready to receive packages. Under the 'Send' tab, type in the desired package and send it to the board. Follow the description in Section 3 for the package format.

6.3 Hierarchy Chart

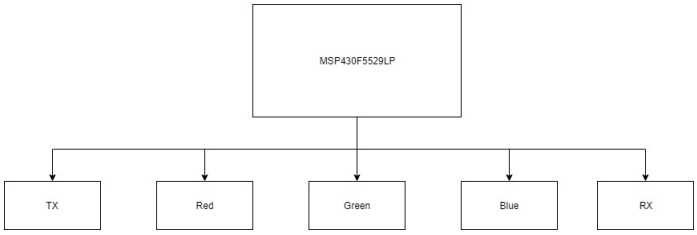

Figure 4: Hierarchy Chart

Figure 4 shows the hierarchy chart for this system. At the top of the hierarchy is the MSP430F5529 microcontroller, with the bottom level being the devices it controls.

7 Test Setup

This RGB LED controller uses UART to receive and transmit information. RealTerm is used to communicate with the microprocessor over UART. To configure RealTerm the correct port must be selected and set to open with a 9600 baud rate under the port tab. The package can then be sent to the microcontroller using the send tab. The package must be sent as a series of hexidecimal numbers where the first is the size of the package and the next three set the brightness of the red, green and blue LEDs. The rest is to be transmitted out over UART for the next LED. RealTerm shows what

has been transmitted back in the main terminal. The system is functioning correctly if it transmits back the correct package and displays the correct color based on the bytes received.

7.1 Test Data

For testing the design, various packages of bytes were sent using Realterm and the output behavior was observed. One sample of test results is displayed in both Figure 3 and Figure 4. The package sent was 0x06 0xFF 0x00 0x00 0x66 0xAC 0xFF. This is a 6-byte package.

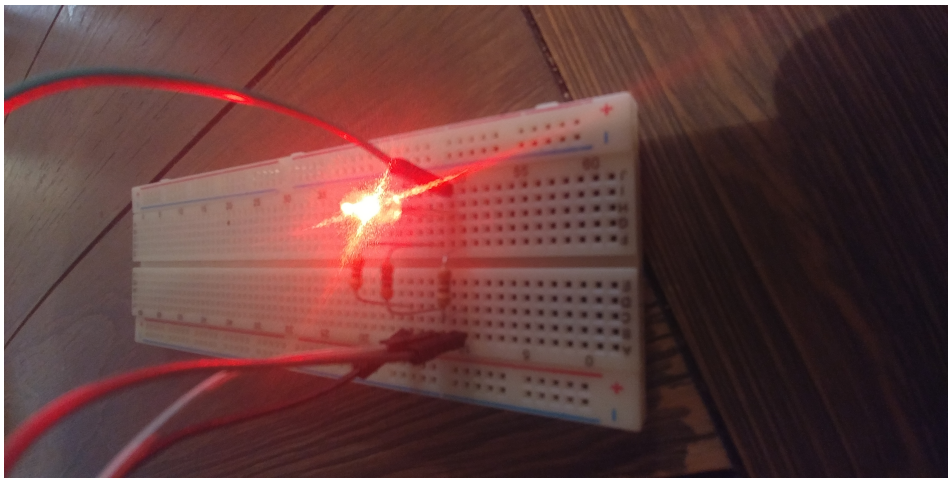

Figure 5: LED Results based on test package

As displayed in Figure 3, the LED is red, which is correct for the given package. The first three bytes are 0xFF 0x00 0x00, which should set the LED to red at full brightness.

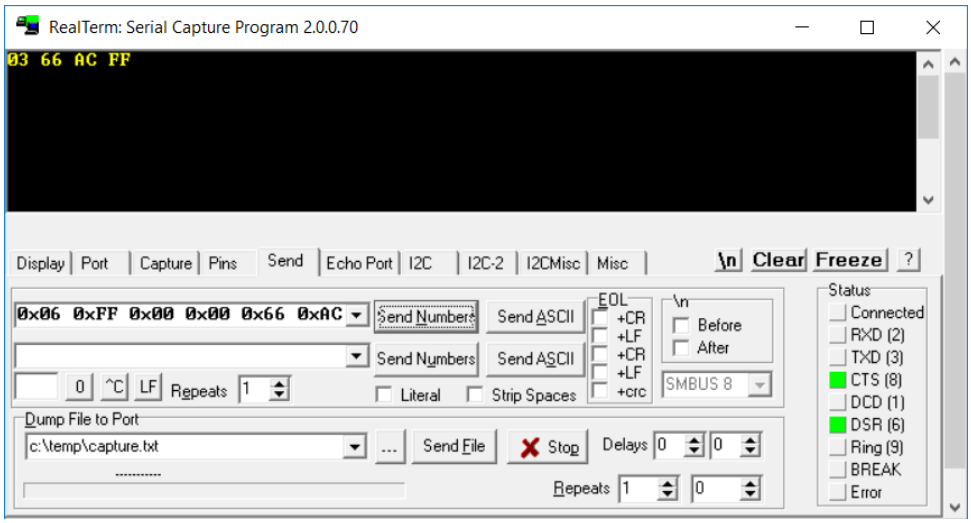

Figure 6: Expected Results

Figure 4 shows the package being transmitted from the TX channel. The package is correct as the first 3 bytes are removed and the second set of three is being passed on. This confirms proper functionality of both RX and TX as the correct package was received and the update package was transmitted.

8 Design Files

8.1 Schematics

See Figure 3 for the system schematic as it is identical to the block diagram.

8.2 Bill of Materials

Component	Description	Value
MSP430F5529	Texas Instruments Microcontroller	NA
RGB LED	Red, Green, Blue LED; Common Anode	NA
R1, R2, R3	Resistors: 0.25 W 5% Tolerance	300 Ohm
Breadboard	Solderless Breadboard	NA