Stranger Things LED Light Wall

Jacob Matteo and Alex Jackson
Rowan University

November 25, 2019

1 Design Overview

The goal of this project was to model a light wall from the show ”Stranger Things”. The board used for this was the MSP430G2553, and featured elements such as LED control, timers, interrupts, and pulse width modulation. An RGB LED was positioned on a breadboard, that was then wired to the G2 micro-controller. A code was loaded onto the G2, which allows for the micro-controller to communicate with the RGB LED using UART. Sending hexadecimal codes through the UART port changes the color of the RGB LED. The end goal is to connect multiple micro controllers together, allowing for each LED to receive data from one UART port, and send it to the next LED in the chain.

1.1 Design Features

These are the design features:

- UART communication through MSP430G2553 board
- HEX codes for certain colors of the RGB LED
- Pulse Width Modulation system to control the brightness of the LED, allowing for a variation of different colors to be showcased
- Able to send bytes to LED nodes that store information about the color of the LED
- Takes three HEX messages through UART for each color, and sends any additional messages along to the next node

1.2 Featured Applications

- Transmits data
- Receives data
- Can be used across multiple boards
- Used for programming LED displays

1.3 Design Resources

The repository for this project which contains the project explanation as well as the code can be found at the following link:

https://github.com/Intro-To-Embedded-Systems-RU09342/milestone-1-ajax7778.git

1.4 Block Diagram

The block diagram below shows each of the micro-controllers as ”nodes” connected together in an ideal chain, passing information from one main source to multiple micro-controller boards.

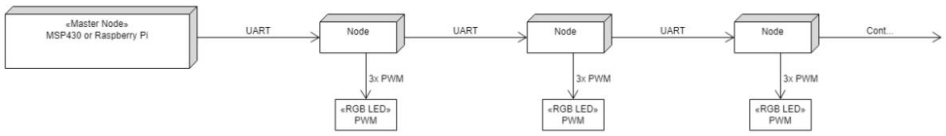

Figure 1: Simple Block Diagram

1.5 Board Image

The following images show the RGB LED positioned on a breadboard which is connected to the micro-controller. In order to visualize the brightness of each color individually, a red, green, and blue LED were positioned behind the RGB LED.

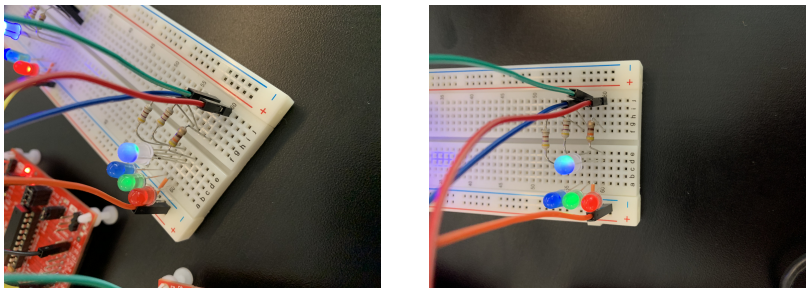

Figure 2: Breadboard Images

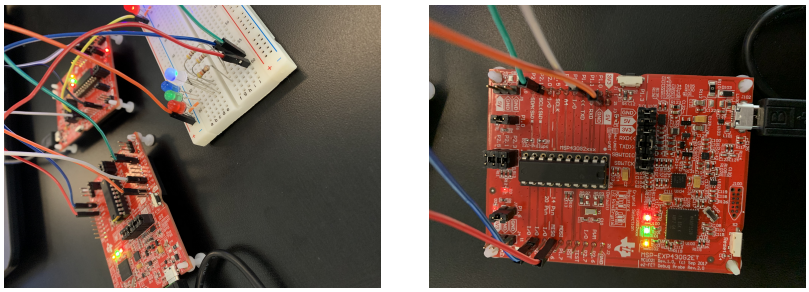

Figure 3: Connection to Micro-controller Images

2 Key System Specifications

The specifications of micro controller timer pins and voltages are shown in the table below. The timers, referred to in code as TA0CCR0, TA1CCR1, and TA1CCR2 are associated with each LED. These are set to specific values that control the pulse width modulation of each of the LED colors. The table below shows which pins and LED colors these timers are associated with. The table also includes the forward voltages of each of the colored LEDs contained within the RGB LED. The forward voltage is the amount of voltage needed in order for the current to flow across the LED.

PARAMETER	SPECIFICATIONS	DETAILS
TA0CCR0	P1.6	RED LED port of RGB timer
TA1CCR1	P2.1	GREEN LED port of RGB timer
TA1CCR2	P2.4	BLUE LED port of RGB timer
Forward Voltage	2V	RED LED
Forward Voltage	2.2V	GREEN LED
Forward Voltage	3.3V	BLUE LED

3 System Description

The problem trying to be solved is to create a color coded communication system based on an RGB LED and a micro-controller. The data is sent from the micro-controller to a breadboard where an RGB LED receives the data and displays the color. The data is able to be sent using the micro-controller's built in UART communication system. The data transmitted includes the pulse width value for each color of the RGB LED, so that many different colors can be showcased. This G2 board and the code sent to it is designed to work with other micro-processors, so a chain can be made that connects many of these systems together.

3.1 Detailed Block Diagram

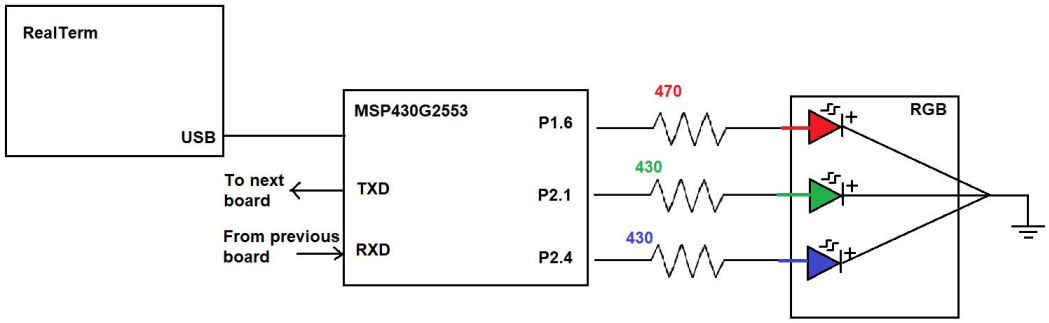

Figure 4: Detailed Block Diagram

3.2 Highlighted Devices

- MSP430G2553: This is the micro-controller used for this project. The code is loaded onto this device and drives the RGB LED.
- RGB LED: Main output of the code, changes color based on specified data bytes
- Breadboard: RGB LED was placed on the breadboard along with three resistors (one for each color), and the micro-controller was routed there

3.3 Device- MSP430G2553

The MSP430 family of micro-controllers includes many different boards with hundreds of features. The MSP430G2 family is a part of the MSP430 series that includes multiple 16-bit timers, 24 accessible pins, ultra-low power, and UART communication capability. This specific board, the MSP430G2553, was chosen due to its two timers, TA0 and TA1, each with three capture and compare registers. Programming both of these

timers allows for each of the LEDs to be interfaced to different capture and compare registers. The only pins needed on this board are the UART transmit and receive, the Timer outputs for each of the LEDs, and the common ground.

3.4 Device- RGB LED

The RGB LED was used as the main source of communication for the light wall. It works by connecting it to a breadboard, and connecting a resistor to each of the RGB nodes in order to properly regulate the voltage. The RGB LED pinout can be seen in the figure below.

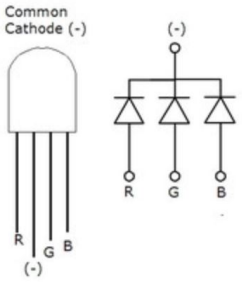

Figure 5: RGB LED Pinout

4 SYSTEM DESIGN THEORY

The system design for this project contains two major components, the MSP430G2553 code, and the breadboard circuit. The MSP430 code is the driving part of this system. It sets up the pins for each LED, the UART communication, and the PWM system to control the brightness of each LED. The breadboard supporting circuit takes the inputs from the UART programmed in the G2 micro-controller, and displays them as a unique LED color. The following sections explain in detail the functionality of each system component.

4.1 Pin Setup

The pins on the micro-controller are set up to correspond with the colors of the RGB LED. Pin P1.6 is connected to the Red LED, P2.1 to the Green LED, and P2.4 connected to the blue LED. The on board LED is also initialized to indicate when data has been received from UART. This pin is P1.0. The other on board pins used are the transmit and receive pins, TXD (P1.2) and RXD (P1.1). The receive pin takes in data from the UART communication and processes it, lighting the LED according to

the received HEX codes. Setting up a jumper wire from the transmit pin to the receive pin of another board will connect them together, allowing for the UART to send data down a line of boards. The ground of the breadboard and the G2 are also utilized.

The G2 board comes with preset functionalities for each pin. Since Timer A0 was chosen for the Red LED, it must correspond with the proper pin on the G2. The same idea applies to the green and blue LEDs, which both utilized Timer A1. Figure 6 shows the pinouts of the G2 and their functionalities, which the pins used in this project highlighted.

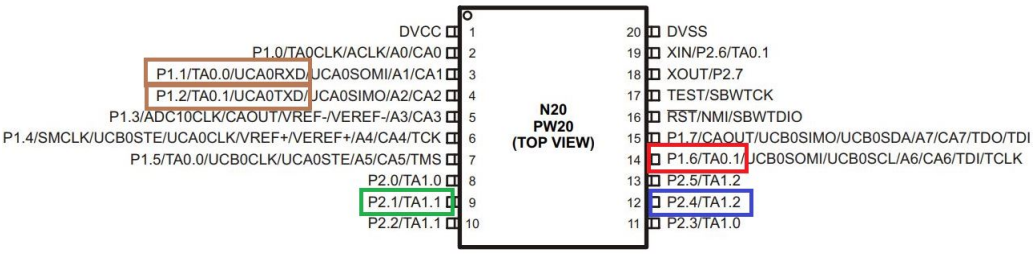

Figure 6: MSP430 Board Pinouts Utilized for this System

4.2 UART Communication

The G2 board allows for universal asynchronous receiver-transmitter, or UART, communication. The code sets the transmit and receive pins, as well as the transmission speed to a baudrate of 9600. The UART communication allows for data bytes to be sent, received, and analyzed to output the proper color on the RGB LED. A block diagram of the UART communication is shown in Figure 7.

The UART works by reading byte strings in hexadecimal, which translate to a specific color. Each board can read up to three bytes at a time, one for each LED color. The first byte in a string must contain the total number of bytes. The device then analyzes the next three bytes as the colors in order of red, green, then blue. If there are more color bytes in the string, they are repackaged and sent through to the next board if one is connected. The final byte in a string is 0x0D, which indicates that the string has ended.

Figure 8 below shows an example byte string. In the example, the first board connected should display a purple color. This is clear because the red LED is at full brightness, indicated by the HEX value 0xFF, and the blue LED is at over half brightness, indicated by the HEX value 0x88.

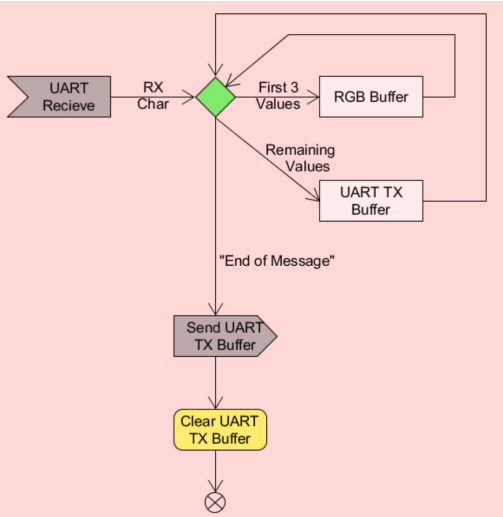

Figure 7: UART Communication Diagram

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 8: Example Byte Configuration

4.3 Pulse Width Modulation

Another main part of this system is the PWM. PWM stands for Pulse Width Modulation, and allows for the LED's brightness to be controlled and display many different colors. The PWM works by modifying the duty cycle of the LED to a certain percentage. The lower the percentage, the dimmer the LED will be. With the RGB LED, the duty cycles of the red, blue, and green LEDs can be modified separately. Figure 9 shows an example of the duty cycle and how it relates to the LED brightness.

In the code, this is implemented using the timers capture and compare values. Each of the LED pins are connected to different timers within Timer A in the G2, and the capture and compare values control the duty cycle. As discussed in section 4.1, the Red LED is connected to Timer A0's capture and compare register 0, which means the CCR0 value controls its duty cycle. A CCR0 value of 255 translates to 0xFF in Hex, meaning it would be at full brightness with this value (a duty cycle of 100%). The green LED is connected to Timer A1's capture and compare register 1, and the blue

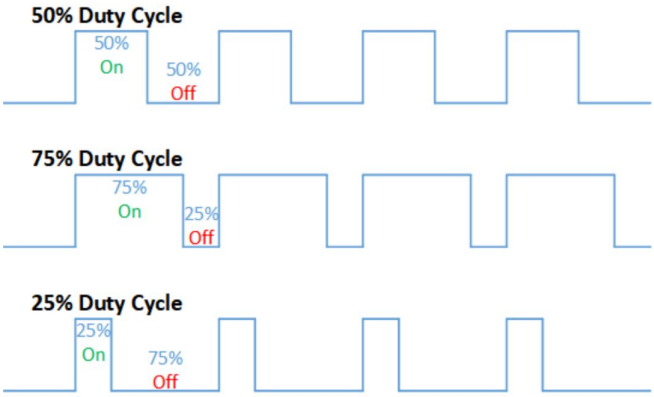

Figure 9: Duty Cycle Examples

LED is connected to Timer A1's capture and compare register 2. This works the same way as the Red LED, except the CCR1 value controls the green LED and the CCR2 value controls the blue LED.

Each of the timers, however, has the same clock setup. The SMCLK in Timer A is set to trigger in up mode, or the high edge, meaning it will count up to the capture and compare value repeatedly. An if statement within the code compares the CCR value to the value sent through the UART communication and modifies it accordingly.

5 Getting Started- Hardware Setup

For the beginning of the setup process, the off board circuit must be constructed. The RGB LED can be positioned on a breadboard, connecting a 470Ω resistor to the red pin of the RGB LED, and 430Ω resistors to the blue and green pins of the RGB LED. The RGB LED circuit can be seen in the figure below. The resistors are then taken to the MSP430 board by using male to female jumper wires. The male end is connected to the resistor, and the female end is connected to its respective MSP430 pin.

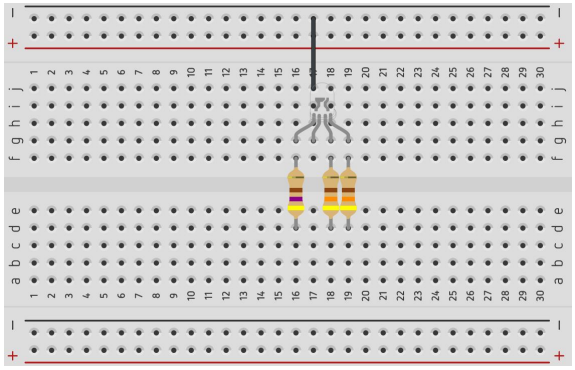

Figure 10: RGB Circuit

6 Getting Started- Software Setup

The code is generated using Code Composer studio. Once the code is retrieved from the Github repository linked in section 1.3, it can be flashed onto the MSP430G2553 board. Flashing the code onto the device allows it to operate without being directly connected to a computer. This makes it possible to connect multiple boards together that are only receiving UART data from one main source.

The other main software needed is RealTerm, which is used to send the data through UART. Once RealTerm is downloaded and opened and the board is flashed, the device is ready to begin setup.

6.1 Communication Between Multiple Boards

Every board is connected to one another through the RXD and TXD links, forming a long chain. The first board in the chain link is the board connected to the host terminal, typically through the USB interface with no connections to the RXD pin. The next board, and all boards thereafter, is connected from the previous board's TXD pin to its own RXD pin. As more boards are interconnected to one another, each addition is farther down the chain. The last board will have no connections to its Txd pin.

7 Test Setup

With the code properly flashed to the MSP430 and the breadboard circuit built, the last steps in setting up for testing is connecting the micro controller to the breadboard and setting up RealTerm. The diagram below shows how to connect each piece of the system using jumper wires.

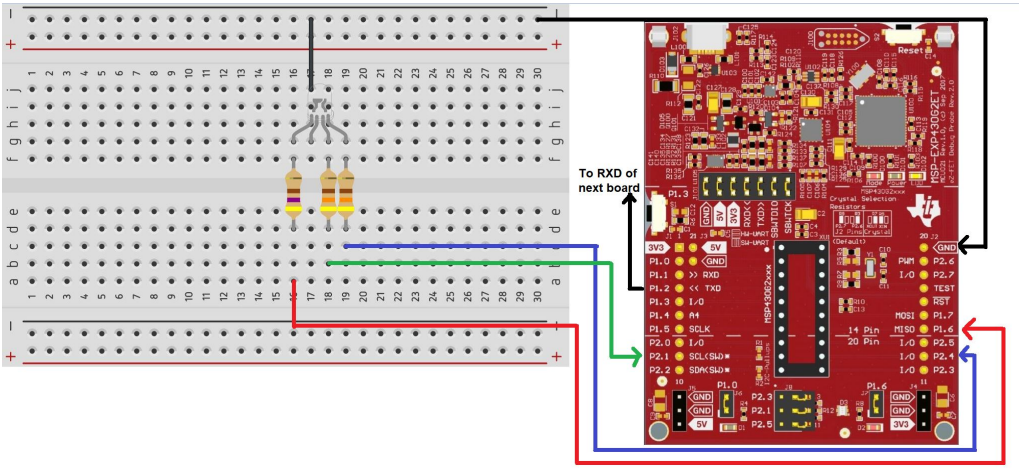

Figure 11: Jumper Connections from MSP430 to BreadBoard

Using RealTerm, bytes can then be sent to the board and the LED. In RealTerm, the baudrate should be set to 9600, the display set to HEX[space], and the Half Duplex box checked. In the laptop's Device Manager, the COM port associated with UART can be viewed. This is the port that it should be set to in RealTerm. Pushing the "Change" button in the Display and Port tabs saves these settings. The figures below show what RealTerm should look loke before bytes are sent.

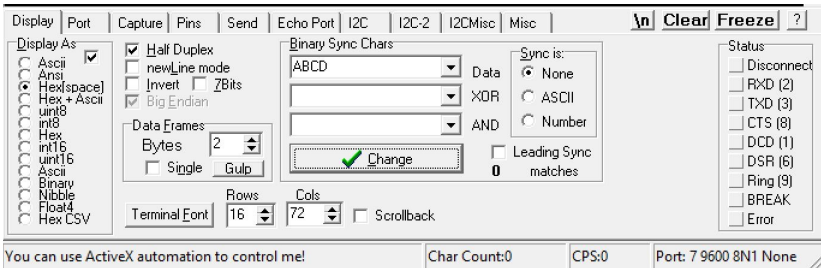

Figure 12: RealTerm Display Tab Setup

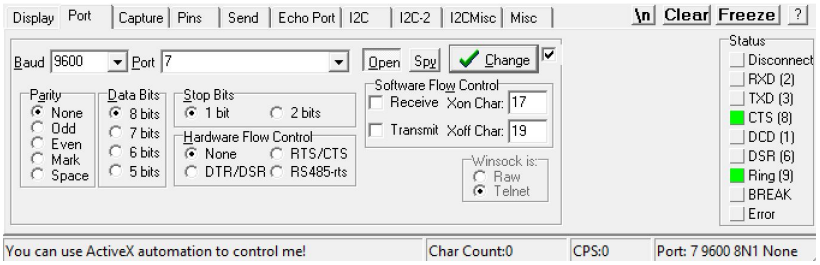

Figure 13: RealTerm Port Tab Setup

7.1 Test Data

Using Realterm to send data, the string ”0x05 0xFF 0xFF 0x00 0x0D” which will turn on the red and green LEDs inside the package, producing a yellow color. Then, the value ”0x05 0x00 0x00 0xFF 0x0D” was sent, turning the LED from yellow to blue. Since this worked in testing, two boards were connected with a wire from the first’s Txd to the second’s Rxd. Then, the string ”0x08 0x00 0xFF 0xFF 0xFF 0x00 0xFF 0x0D” was sent to turn the LED of board 1 to cyan and board 2 to purple. Then the string ”0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x0D” was sent, turning the LEDs off.

8 Schematics

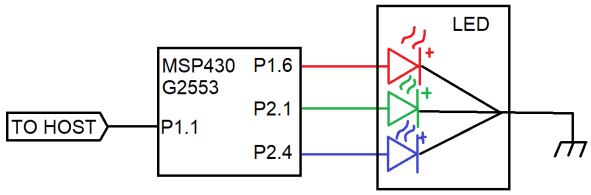

Figure 14: Schematic for a single board setup

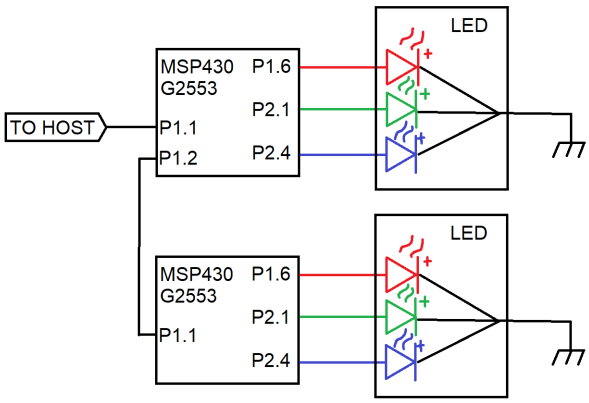

Figure 15: Schematic for a dual board setup

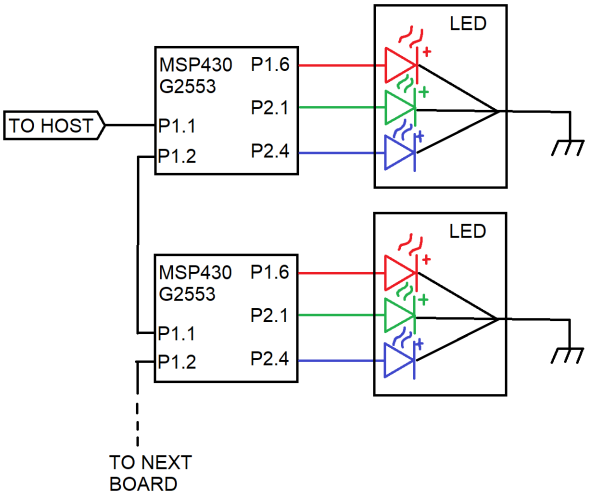

Figure 16: Schematic for a multiple board setup