– ECE 09.321: Milestone 1 –

Application Note - Team MD

Bill Dripps and James Merrill

Rowan University

October 21, 2018

1 Design Overview

This AN describes a universal asynchronous receiver/transmitter (UART) enabled red-green-blue (RGB) light emitting diode (LED) addressing node built on an MSP430-based microcontroller (μC) board. The node controls the output brightness of each of the RGB portions of a tricolor LED. As a list of addresses is generated and sent to the node, it extracts information pertinent to its own operation, and transmits a new list of addresses to the next node in the network, if necessary.

1.1 Design Features

The node presents the following features:

- 8 MHz clock speed
- 9600 baud UART interfacing
- PWM controlled RGB LED

1.2 Featured Applications

- Single LED Light Show Generator
- Stranger Things' Will Byers Cosplay Device

1.3 Design Resources

https://github.com/RU09342-F18/milestone-1-team-md/blob/master/mdMain

1.4 Block Diagram

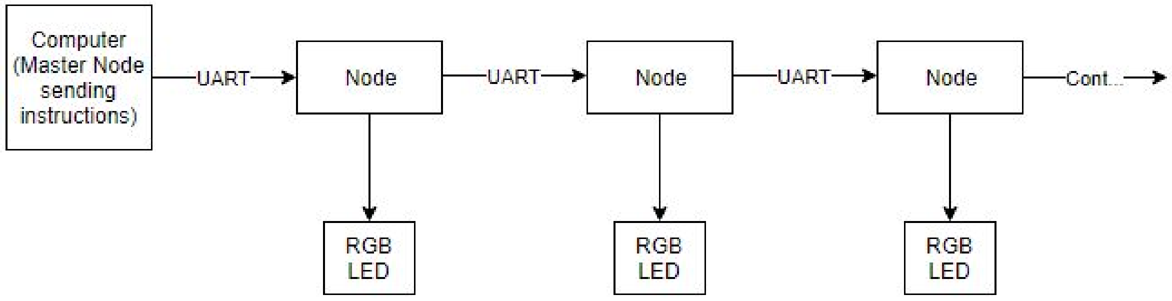


Figure 1.

1.5 Board Image

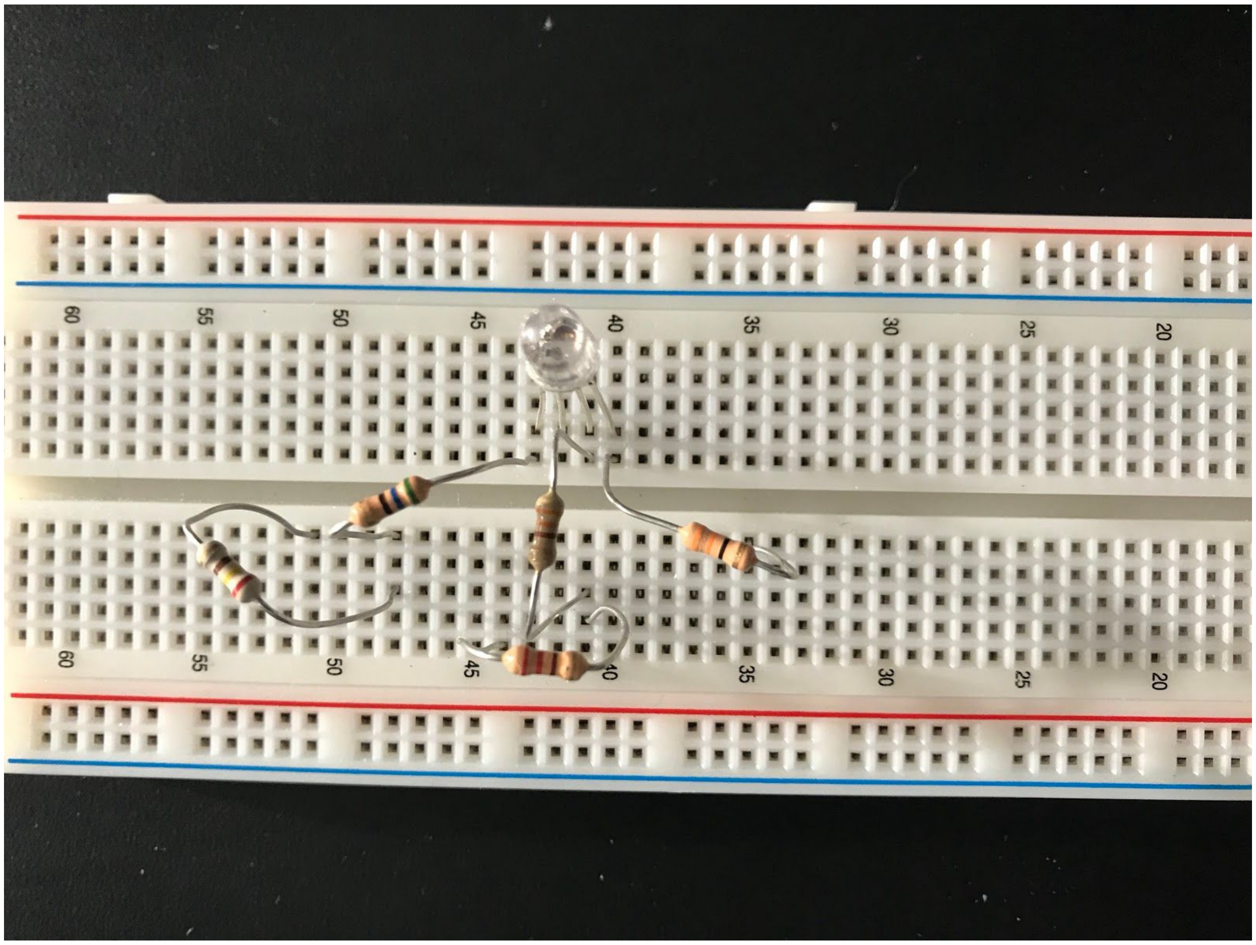


Figure 2.

2 Key System Specifications

Parameter	Specification	Details
Baud Rate	9600	*The rate at which each bit, or pulse, is passed through UART Communication*
PWM Precision	0.4 %	*How precise the brightness of each LED can be controlled (1/255)*

3 System Description

In its entirety, the Milestone 1 system is a single instruction being passed down to a node or series of nodes. The first node takes the first three bytes of actual instructions, and passes this information to the three LEDs on an RGB LED. Once the first node has interpreted the first three bytes, if there's another node in the series, the next three bytes of instructions are used for the second node. The second node interprets the instruction just like the first node and outputs a corresponding color value. These actions repeat themselves for however many nodes are in the series. The number of nodes should correspond to the length of the instruction.

3.1 Detailed Block Diagram

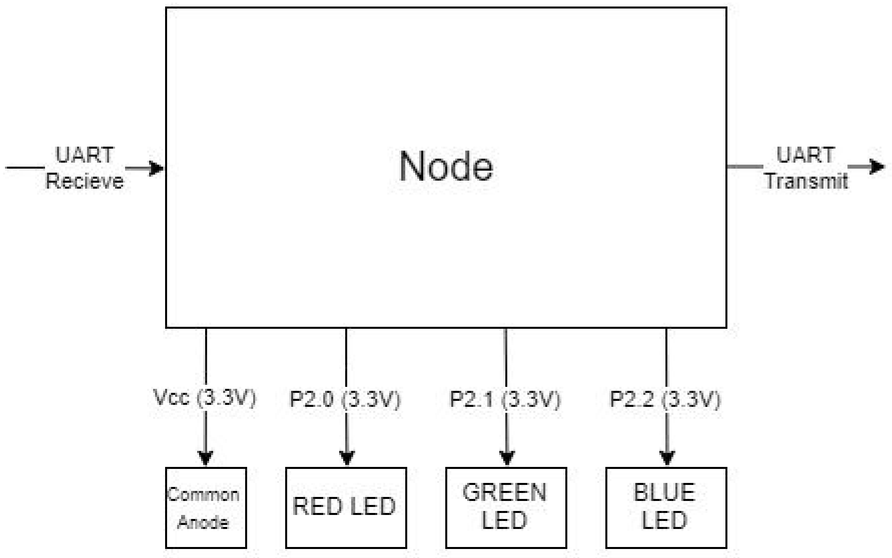


Figure 3.

3.2 Highlighted Devices

- MSP430G2553 - *The node of the system*
- RGB LED - *output device, displays specific color*

3.3 MSP430G2553

This is the programmable processor that is the centerpiece of Milestone 1. The MSP430G2553 interprets the instructions being stored in the UART RX (Receive) pin and converts the data to display an expected color. This is done by creating three PWMs (software), controlling each of the 3 LEDs, whose duty cycles are controlled by the value of the instruction. Once these values were outputted, the next set of instruction were passed on the UART TX (Transmission) pin onto the next node's UART RX pin.

3.4 RGB LED

The RGB LED used was a common anode RGB LED. This meant that to turn any of the LEDs on, the pins from the μC would need a 0V value to output light, since they were ultimately connected to the separate cathode terminals of each of the LED colors. This made all of the output pins utilize negative logic. This was accounted for by programming each of the pins to output a "1" or 3.3V signal when "on" portion of the PWM signal was over.

4 System Design Theory

The Milestone 1 system is centered around two components, the processor (MSP430G2553) and the output (RGB LED). The processor is the brain of the system controlling what instructions it receives, what is does with those instructions, and what it does with the instructions it does not use. The RGB LED connected to the system shows that the processor is receiving, interpreting and passing these instruction properly. For the processor and LEDs to work properly together, a few design requirements were made.

4.1 Processor Reading Specific Instructions

For the RGB LED to output the proper value, it must receive the correct instructions. This was done by setting the first byte of the package to the size of the package. The number of nodes in the system is equal to the first byte value subtracted by 2 (the initial bit and ending bit don't give any instruction to individual nodes) divided by 3 (for the three LED pins). To extract the proper instruction for each node, the value of the number of bytes is stored into both a non-changing (while one program is running) variable "total", and a decrementing variable "counter". The program checks counter and compares it to the total each time a new byte occurs. When the counter is one less than total, the data inside the current byte is converted into a value the processor can understand and is stored as the red LED duty cycle. When the counter is two less and three less than total, the green and blue values are stored accordingly. Lastly, if the total is greater than or equal to 8 (signifies at least two nodes in the system) the rest of the data is put on the transmission pin which goes to the next nodes in the series.

4.2 Fast Internal Clock Rate

The MSP430G2553 has 3 16-bit capture/control registers (CCRx) per onboard timer. It was decided to utilize only 1 timer, and that CCR0, CCR1, and CCR2 would be used to control the time to switch off since the last timer overflow. Because the overflow of a 16 bit timer was used to synchronize the turning on of each of the colors in the tricolor LED, the period of the output could be described as 65,535 + 1 clock cycles, which in turn would result in an LED blinking rate determined by the equation $f_{clock}/65536 = f_{LED}$. In order to ensure that the LED did not suffer from flicker, an 8-MHz clock speed was chosen to give an LED frequency of 122 Hz.

The 8 MHz clock speed had further design implications beyond the LED. In order to maintain a 9600 baud rate from an 8 MHz clock, the clock must be divided by 833⅓. Since this is not an integer value, the clock generator has to run 2 cycles at an 833 division and 1 at 834 in order to average 833⅓. Successful operation in this mode was made possible by setting the UCA0BR0, UCA0BR1 and UCA0MCTL to values corresponding to this type of operation.

4.3 Each LED Shines as Bright

Inside the RGB LED is essentially 3 LEDs whose anodes are tied together. Using a constant 2 mA source, the resulting voltage drop was measured across each of the three colors. These were empirically determined to be 1.8v for red, 2.3v for green, and 3.1v for blue. Since a 3.3 volt source was to be used, unique resistor values between the individual cathodes and their corresponding pins would have to be used in order to maintain the targeted 2 mA draw in each circuit. The formula $(3.3 - V_{drop})/0.002 = R$ was used to determine the appropriate resistance applied to each cathode-to-pin connection. This resulted in a resistance of 750Ω for red, 500Ω for green, and 100Ω for blue. These values were tried, and while the current draw on each color was nearly perfectly 2 mA, the relative brightness levels of the three colors were not similar. To compensate, different values of resistors were tried until similar duty cycles of PWM applied to the three colors produced similar brightness from each color upon visual inspection. The resistances ultimately used were 296Ω for red, 550Ω for green, and 33Ω for blue.

5 Getting Started/How to use the device

Initially, the device needs to be powered. This can be done via USB or connecting a 5V power supply with the positive lead on a 5V terminal and the negative lead on GND. Once powered, the device needs to be properly connected to the RGB LED. This is done by connecting the processor's designated red, green and blue pins (P2.0, P2.1, P2.2, respectively) to the actual pins on the RGB LED and the 3.3V power pin to the LEDs common anode pin. The next step is to test the UART communication capabilities. The MSP430G2553 has two pins controlling its UART capabilities, RX (Recieve) and TX (Transmission). To connect processors together, connect the first processor's transmission pin to the following nodes receive pin. Continue this connection procedure for the entire system.

6 Getting Started Software/Firmware

Similar to the starting the device, the system needs to be functional with one node. This is done by creating a program that controls three separate PWMs manually. Creating a program where the data input isn't passed in via UART creates a framework that UART can be easily implemented into.

6.1 Communicating with the Device

Initially, with no UART connected, the device will communicate via USB. From there the initial PWM framework can be created. Once UART is implemented into the program, if you are not the first node of the system, communication will be done via the UART RX and TX pins. The device will need to be programmed properly via USB before connecting the UART pins.

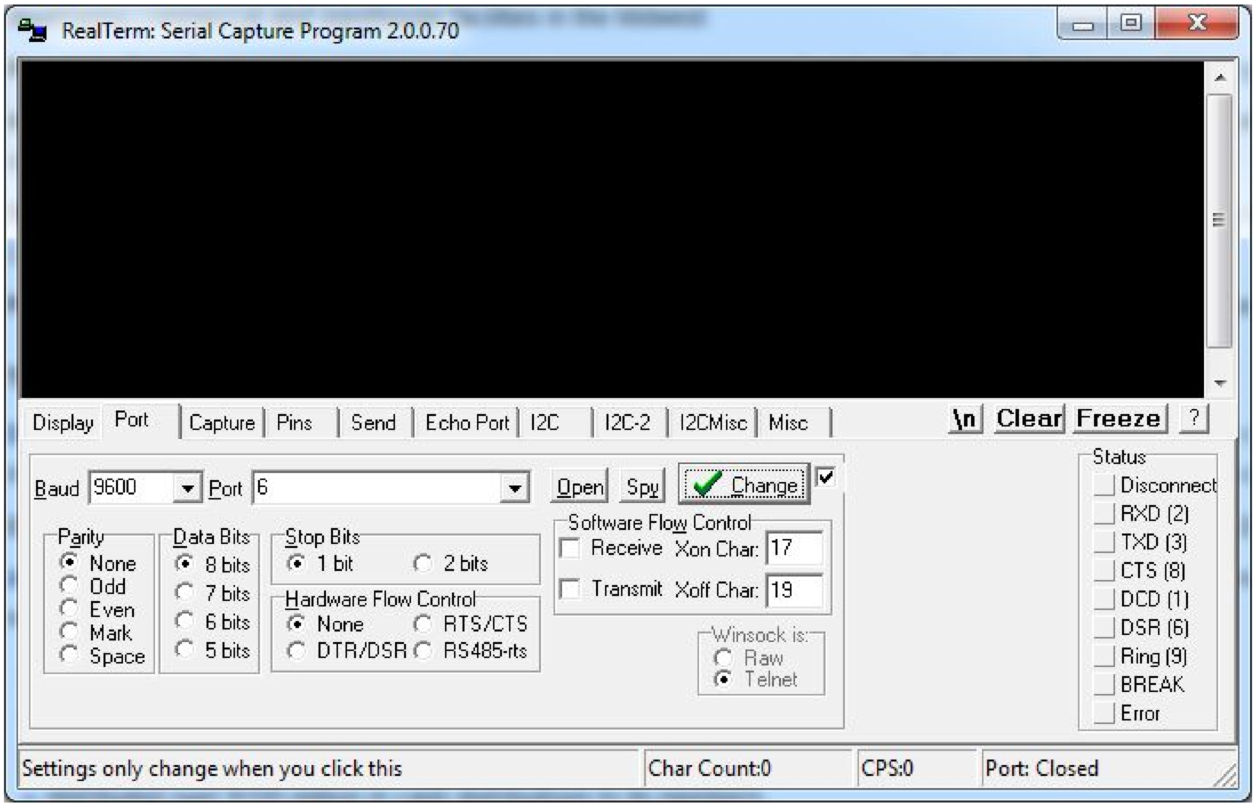
6.2 Device Specific Information

As previously stated the MSP430G2553 only includes 3 capture/compare registers per internal timer. For most PWMs you need a capture/compare register for the duty cycle and the period. Since there are 3 PWMs, there's not capture/compare register for period. To solve this issue, either two timers must be used or the clock overflow set to be the total period and the 3 capture/compare registers as each duty cycle.

7 Test Setup

In order to test the node, a terminal program capable of sending data into the node should be used. In this case, the program RealTerm was used, which can be found free of charge at:

- https://sourceforge.net/projects/realterm/



Baud 9600 Port 6

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

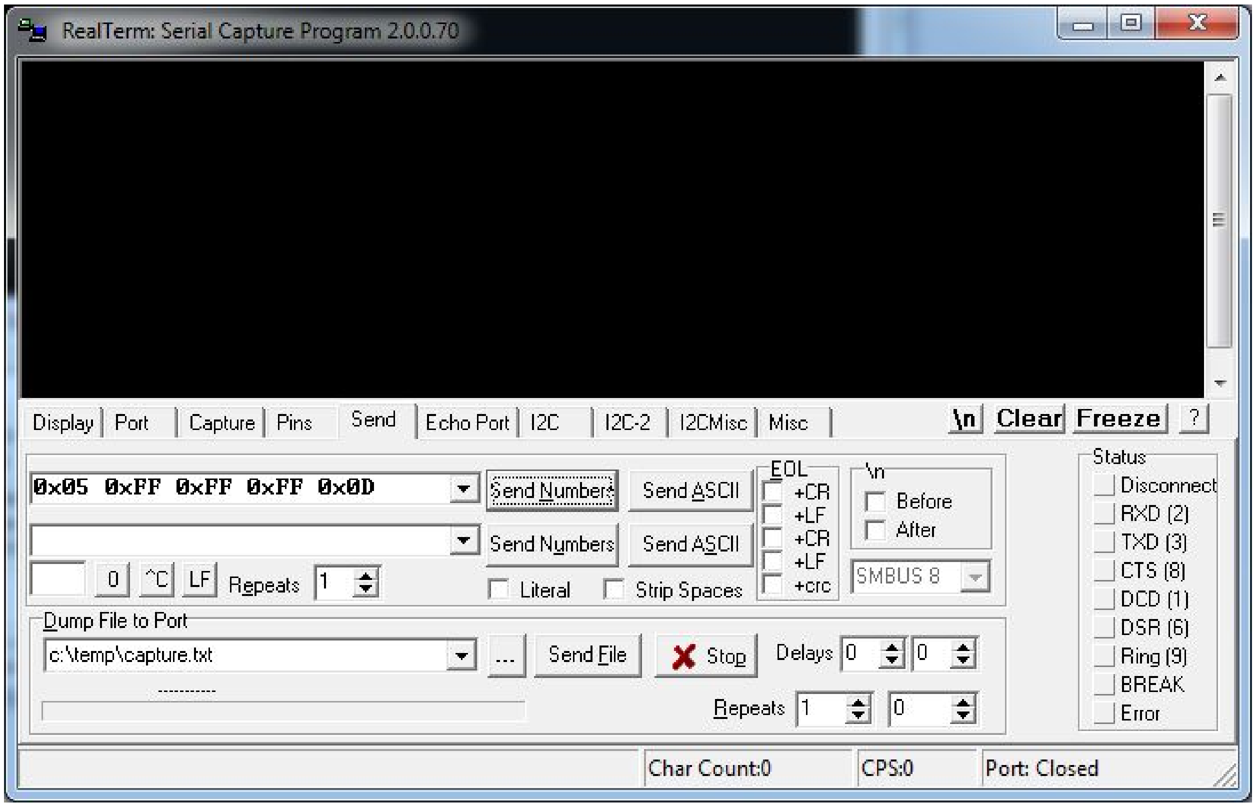
Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Settings only change when you click this | Char Count:0 | CPS:0 | Port: Closed

Figure 4.



```
0x05 0xFF 0xFF 0xFF 0x0D
```

Repeats 1

Dump File to Port: c:\temp\capture.txt

Delays 0 0

Repeats 1 0

Char Count:0 | CPS:0 | Port: Closed

Figure 5.

The program should be set to send and receive data at the same rate as the node, namely 9600 baud. The port through which the program tries to communicate should be set to whichever port the node is interfaced through. If another program is using the communication port, this could cause interference in trying to send data to the node. A gaggle of key commands and expected outcomes are described below:

- 0x05 0xFF 0xFF 0xFF 0x0D - Full Brightness
- 0x05 0x7F 0x7F 0x7F 0x0D - Half Brightness
- 0x05 0x00 0x00 0x00 0x0D - Zero Brightness
- 0x05 0xFF 0x00 0x00 0x0D - Full Red
- 0x05 0x00 0xFF 0x00 0x0D - Full Green
- 0x05 0x00 0x00 0xFF 0x0D - Full Blue
- 0x08 0xFF 0xFF 0xFF 0x55 0x55 0x55 0x0D - Receive full brightness and send 0x05 0x55 0x55 0x55 0x0D to the next node.

In the last case shown above, the reasoning for choosing 0x55 for each of the RGB duty cycles for a second unit is that without a second unit to test, the TX pin can instead be connected to an oscilloscope, and the output checked for an easily recognizable pattern. 0x55 should produce alternating low and high states. The image shown below did not utilize these values, but demonstrates the concept effectively.

7.1 Test Data

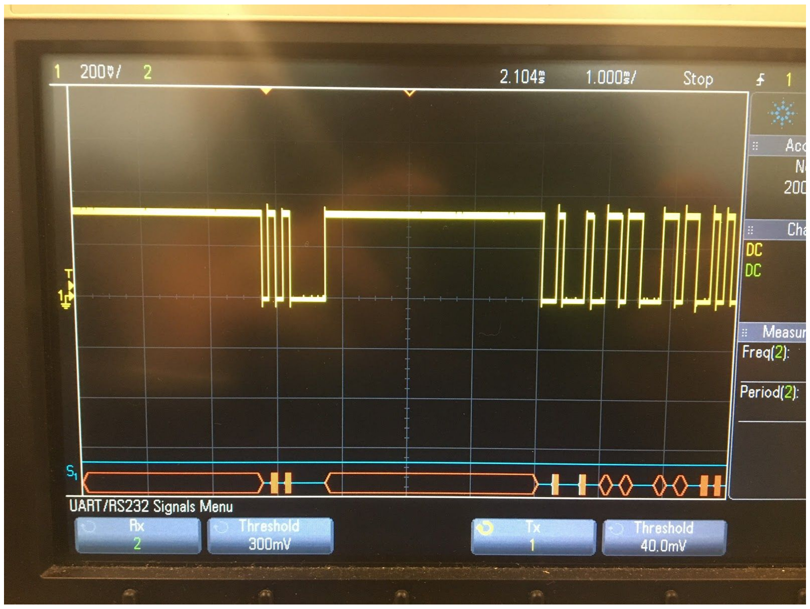


Figure 6.

8 Design Files

8.1 Schematics

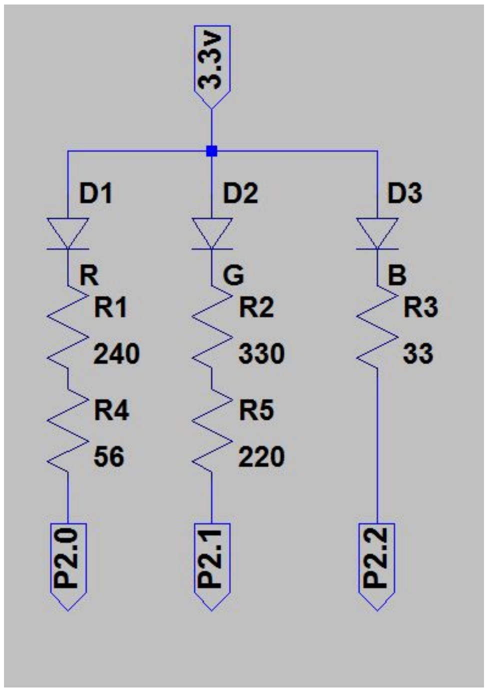


Figure 7.

8.2 Bill of Materials

Component	Description	Tolerance
MSP430G2553 Development Board	n/a	n/a
Resistor	240Ω	5%
Resistor	330Ω	5%
Resistor	33Ω	5%
Resistor	56Ω	5%
Resistor	220Ω	5%
LED	Tricolor RGB	n/a