Application Note Template

Tyler M. and Helen P.
Rowan University

October 22, 2018

1 Design Overview

In a scene from ”Stranger Things” on Netflix, Will Byers had to communicate with his mother by lighting up Christmas Lights, since he was stuck in the parallel dimension. The following lab, students had replicated the RGB LED communication through implementing two main tasks. The first task required students to program a microprocessor of their choice to PWM (Pulse Width Modulation) an RGB LED. This would enable the groups to choose the displaying color of the LED. The second part of the lab required students to setup their processor to communicate with other teams. Hence, students had to program their processor to both transmit data,instructions for what color to make their LED, and receive data, what color to make their own LED. In addition, students were required to figure out how they would tackle both of these problems, given the constraints on what their microprocessor could physically do. The following report details the methodology and hurdles that were over come in order to allow this project to come to fruition.

1.1 Design Features

These are the design features:

- Ability to PWM RGB LED
  - Each color controlled independently
  - Brightness for each color can be controlled
- Receive LED color instructions from previous node
- Transmit color instructions to next node

1.2 Featured Applications

- RGB LED Devices
  - RGB keyboard or mouse
  - RGB string lights
  - Internet of Things

1.3 Design Resources

A link to the GtHub repository used for this project can be found directly bellow.

`https://github.com/RU09342-F18/milestone-1-nanotech`

1.4 Block Diagram

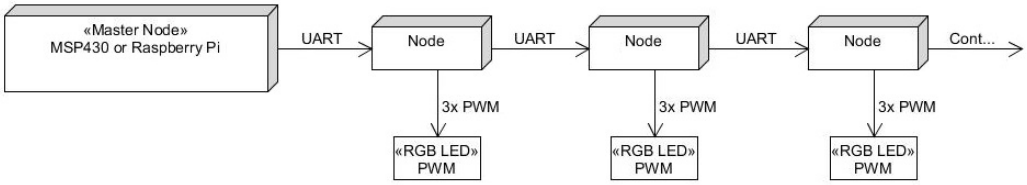

Figure 1: Overall Network of Devices

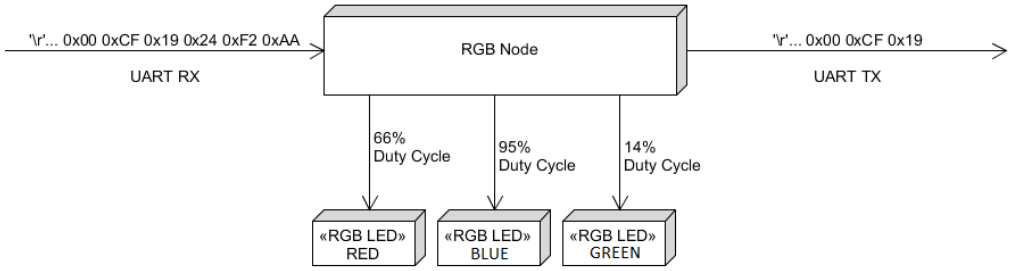

Figure 2: RGB Nodes

1.5 Board Image

The MSP430G2553 board was shown in figure 3.

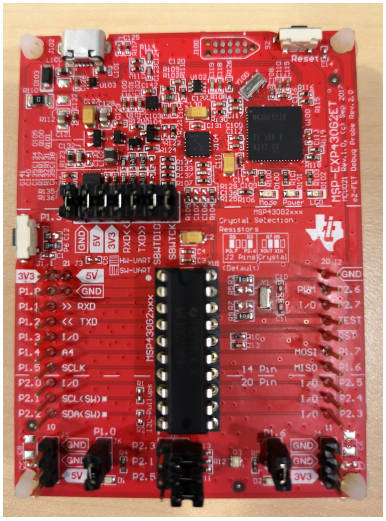

Figure 3: MSP430G2553

In figure 4, the MSP430G2553 was connected to the breadboard using female-male jumper wires. On the breadboard, three 530 ohm resistors were used in series with the RGB LED. Each LED was parallel to one another, in order to reduced the current flowing back into the MSP430G2553 board.

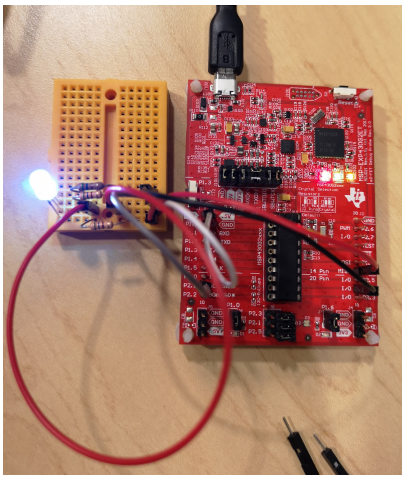

Figure 4: MSP430G2553 connected to breadboard with RGB LEDs

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
LED Voltage	3.3 Volts	The voltage for the LED is 3.3 volts since the LED's use the I/O ports to ground themselves. Higher voltage could theoretically be used if the ports voltage was were also increased
Back feed amperage	0.003 Amps	As per MSP430G2553 design specifications the amperage on the device can not exceed 0.003 amps. This can be solved by using current limiting resistors

3 System Description

The problem at hand requires the microprocessor to both PWM an LED, send instructions to other nodes, and receive instructions from other nodes, with all or some of these happening at the same time. However, since the processor is not multi-threaded, and isn't particularly fast, it can not process multiple things at once or even give the illusion of processing multiple things at once. This means that a very low overhead program must be written to make sure that there are enough free CPU cycles to do all the tasks this problem requires. Another problem quickly encountered during the making of this project is the lack of features on this microprocessor. The RGB has three distinct colors on it (Red, Green, Blue) and each one would need it's own timer in order to be PWM. Hence, the problem is there are only two timers on this microprocessor.

3.1 Detailed Block Diagram

The main difference here between the previous block diagram is the level of detail. This should show specific pins on the device and voltage levels. Don't go so far as putting in pictures of how the peripherals are connected to each other inside of the controller, but enough that you could basically put this thing together again.

3.2 Highlighted Devices

- MSP430G2553
  - The ”Brains” of the project. Where the code is actually run and where all the LED values are stored, processed, sent, and received.
- MSP430 LaunchPad
  - The LaunchPad provides pin outs for numerous useful I/O pins, some of which are used in this project.
- Breadboard
  - The Breadboard allows for the connection of numerous electronic components in a neat and organized way, with out the risk of two components touching.

3.3 MSP430G2553

The MSP430G2553 is the ”Brains” of the overall design. This chip allowed for advanced features that otherwise might not have been possible had the device been designed from the ground up. This chip has multiple features but the ones being used in this project were; CPU, RAM, Timers, UART compatible bus. The CPU allowed code to be compiled and run on the board. The RAM is what stores all the necessary values for the code to work. For example, in order for the program to function correctly, it must know the total number of bytes it is going to receive, and how many bytes it has already received. This would normally prove difficult, because there is nothing encoded into the bytes themselves that distinguish them from one another. However, the program explicitly stores the total number of bytes when it learns this value, and counts the number of bytes it has received every time it receives a new packet. This allows the program to program the correct LED’s as well as send information to the next node when it gets to those bytes. The timers are possibly the most important part. The timers are a hardware element that tick up at a known value and can generate interrupts on a programmable interval. Without the timers present in the device, the CPU itself would have had to serve as the timer, which would have meant that the CPU could have been too busy keeping time to actually do all of it’s other tasks.

3.4 MSP430 LaunchPad

The MSP 430 launchPad helped with the breakout pins for the device. The launch pad allowed for the direct connection of prelabeled pins to the bread board used in the final product. The LaunchPad itself has all of it’s pin outs labeled on the PCB. This made it easier to identify the ones needed and connect them to the breadboard. The LaunchPad also provided a 3.3 volt rail, which was used to provide power to the RGB LED.

3.5 Breadboard

The bread board was integral in the functionality of this device. the breadboard allowed for safe and organized construction of the circuitry required to make the device function. The main components on the breadboard are; The LED, mosfets, and resistors. The LED was provided by a lab instructor, so the make and model is unknown. However the the LED had a four prong design. The first, third, and forth prong were Red, Green, and Blue respectively. The second prong for the LED was the power. With this being the case, the power prong was hooked directly up to VCC (3.3 volts) and the other prongs were hooked up to their respective mosfets. The gates of each mosfet went to the reset button. To prevent flashing when the button was pushed. The mosfets then each run to their respective port (For example, the RED LED mosfet runs to port 1.6).

4 SYSTEM DESIGN THEORY

The overall design philosophy for this device was low overhead. The device needed to be fast for a number of reasons. One of the major factors that drove this design philosophy was that if each section had a minimal impact on CPU time, the CPU can be busy doing other things. Another driving factor was power efficiency. The longer the device spends asleep doing nothing, the more power efficient it is. This means that the less instructions the CPU has to process, the longer it can spend asleep.

4.1 Design Requirement 1: Pulse Width Modulation

The first major hurdle from a design prospective was keeping time. The LED's had to PWM such that they shined at the brightness requested by the previous node. The source of how information is obtained and the brightness value of the RBG are irrelevant in this case. The importance was how the brightness values of the RBG LED are being processed and transmitted to the RGB nodes.

Keeping time with a CPU is normally difficult as it is impossible. However, the MSP430G2553 has 2 on-board timers that can be used in the programs. This is good for two reasons. First, the CPU doesn't have to keep time, so it can be busy doing other things. Second, the CPU can actually go to sleep when it isn't busy, and let the timers keep time in the background.

Since the maximum brightness value for any given LED is '256' (Constrained by the maximum value that a byte can store), the maximum value that all the timers will count up to will also be '256'. In this case the timers start at '0', so the number '255' will be used instead. The timer is configured such that when it reaches it's maximum value, it simply resets itself back to '0'. Next, the CPU had

to program each timer to remember the value of the brightness for each color in the LED. For example, if the red value was '128' the CPU would store the value '128' on the timer.

Normally, the timer would be configured to send an interrupt to the CPU every time it reaches the stored value. This way, the CPU could turn on or off the LED with the correct timing. However, this wastes CPU cycles and uses more power. For this project, the LED's were hooked up directly to the square wave generated by the timer itself. From the example before, the timers square wave would initially be high, and then it would go low once it hit the value '128' (Since this is the value the CPU programmed into it) and then would go high again once it reached '0'. This setup allows for an extremely power efficient device that does not compromise on performance.

4.2 Design Requirement 2: UART Communication

The second design for this device was that it had to be UART compliant. The device needed to be able to send and receive instructions over UART. The device also had to be able to do it at a buad rate of 9,600. The internal baud rate timer is by default set to run at 1 Mhz, so it had to be divided by 104 to bring it close to the target value.

$$\frac{1,000,000}{104} = 9615.384... \tag{1}$$

This ends up with the baud rate being off by just 15 cycles every second. This is a completely negligible amount since devices communicating over UART will try to match each others baud rate regardless of the set amount. On top of this, some device's baud rate can be off by as much as one or two hundred, and still communicate fine.

The next problem that would have been run into is actually sending the packets over UART. Sending information over UART requires the CPU's attention. However, this is where this projects fundamental design philosophy came into play. All the modules in this project were written to be clean and slim to reduce CPU overhead, now all of those saved CPU cycles can be used to communicate with other devices. Granted, the CPU is more than fast enough to keep up with a baud rate of 9,600, however, this is yet another thing it has to do.

UART communication is set up very simply. The processor will receive a packet from UART handle handle that packet immediately. Depending on what that packets purpose is it may store the total number of bytes it is about to receive, or program one of the LED's. The last option is that the processor is receiving information that it needs to send to the next device down the line. In this case, the information is immediately piped from the UART input, straight to UART

output. The information is never read or processed. This has a few benefits. First, power and CPU cycles aren't wasted processing the information, only to have that same exact information sent later. Second, it saves space in RAM. If all of the information that the processor was going to receive was stored in RAM, the processor would end up having to store up to two kilobytes of information. This space is essentially wasted, since the cod

5 Getting Started/How to use the device

5.1 Hardware Overview

Figure 3 shows the board used for this device. The MSP430G2553 rests on a LaunchPad that came with the chip. This LaunchPad provides the connection points needed to drive the LED's. The LaunchPad has several exposed pieces on it that allow the user to test and debug if needed. As well as these test points, there is also a provided Ground, 3.3, and 5 volt rail. For the purposes of this project, the 3.3 volt rail is all that will be needed. Other than the MSP430G2553, 3 mosfets and resistors were also used. The mosfets were 2N7000-F20 mosfets. These mosfets are used to help make the LED's not flash when the board is reset. The resistors are used to help reduce the current flowing back into the processor when the LED's turn on.

5.2 Communication

To communicate with the device, a program that can communicate over UART is required. If the host operating system is a Linux based device, the user just has to navigate to the /dev/ directory. The user then finds their device, and pipes all the commands they want to send directly to that device using their terminal of choice.

For windows users, it is recommended that they grab RealTerm. For more information on how to use this software, please reference Section 6.4.

6 Getting Started Software/Firmware

6.1 Software

Two pieces of software are needed to work on this device. The first is Code Composer Studio. This software is available for the three major operating systems and is free to download for all users. This software comes pre-packaged with all the resources need to develop for the board used in this project.

The second piece of software needed for this project is some for of UART compliant program to communicate with the board while it is running the software. For this purpose, there are two major recommendations, RealTerm, and PuTTY. The in house testing used RealTerm, but PuTTY should also work. For instructions on how to connect realTerm to the device, please reference Section 6.4

6.2 Firmware

In order for the device to run properly, an updated copy of the MSP430G2553 firmware must be installed to the board. To update the firmware, simply load up Code Composer Studio and compile the project to the board. Code Composer Studio will automatically detect that the boards firmware is out of date and prompt you to update it. Click ”Yes” and the program will begin to update the boards firmware. The update process should take around 7 minutes, but can take more or less time depending on Computer and internet speed. Never unplug the device while updating firmware, as this could permanently damage the device.

6.3 Hierarchy Chart

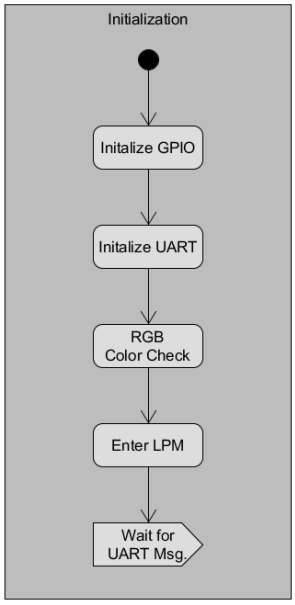

Figure 5: The steps the program goes through to initialize itself for it to function

6.4 Communicating with the Device

Communication with the device while running was almost entirely using UART. In order to send data (Instructions) to the device, a UART connection must be made using a compatible program like PuTTy or RealTerm. The connection to the device is made using the USB port on the launch pad. The other form of communication that the device can be used with is during normal operation. The device makes local UART connections with other compatible devices in order to send and receive instructions on what colors should be programmed to each board.

6.5 Device Specific Information

for more information on the MSP430G2553 please reference the MSP430G2553 Data sheet. Or for more information on the device, please reference the MSP430 Family User Guide.

7 Test Setup

In order to test the device. First ensure that the code is currently flashed to the MSP430. Once the device has the code running on it, the simplest way to test the device is by running the device using just resistors and an LED, no mosfets. First connect the 3.3 volt pin out on the launch pad to the power pin on the LED. Next connect the Red, Green, and Blue lead on the LED to a 500 ohm resistor (In the original testing environment, 530 ohm resistors were used). Connect the Red LED resistor to port 1.6 on the launch pad, the Green LED resistor to port 2.1, and the Blue LED resistor to port 2.4 on the launch pad. The hardware for testing is now fully setup.

The next step requires software to communicate over UART. The original testing environment used RealTerm version 3, however any program that is compatible with the UART protocol should work fine as well. Connect the launch pad to the machine that has RealTerm on it using USB. The Baud rate must be set to 9600, and the connection port must be the same COM port that launch kit is connected to.

Once software is setup, go to the send tab on RealTerm and input the sequence of bytes that will be sent to the processor. The first byte must hold the value that is the total number of bytes in the sequence. The second byte is Red, the third Green, and the forth Blue. The Red, Green, Blue, pattern continues throughout the sequence until you have reached the last byte you want to send. The total number of bytes that is being sent should be such that one minus the total number ”n” is divisible by three.

7.1 Test Data

Internal testing revealed many factors about the device.
One of the first things testing helped prove was on the software side. There is a hard limit on the total number of bytes that can be sent. Only 256 bytes (Including the byte that stores the number of bytes) can be sent. This is because the device uses an 8 bit number to count the current byte it is on. Any further information sent to the device will cause it to overflow and reprogram it’s own LED’s. This problem can be circumvented however. If more than 256 bytes need to be sent, simply send the remaining bytes to the board, but ensure that the first 4 bytes re-encode the number of bytes and the values for Red, Green, and Blue.

8 Design Files

8.1 Schematics

Figure 6 shows a hand drawn schematic used during the development phase to help break down the basics of how the device would function.

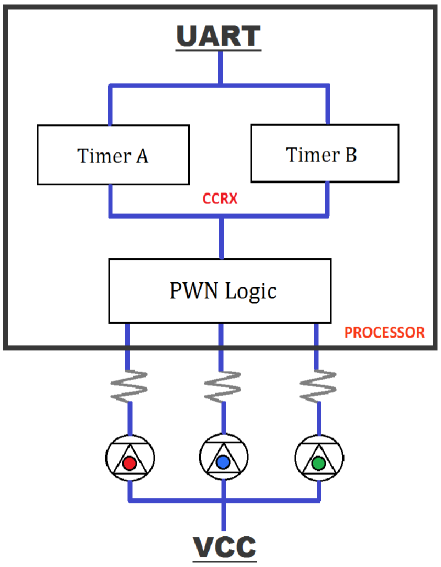

Figure 6: Drawing of the abstract idea behind the device

8.2 Bill of Materials

ITEM #	QUANTITY	NAME	MANUFACTURER	DESCRIPTION
1	1	MSP43G2553	Texas Instruments (TI)	Microprocessor, ”Brains” of the project. All code was run off this device
2	1	MSP430G2553 Launch kit	Texas Instruments	Used as a break out board for the micro processor. Made using breadboards easier
3	1	Bread Board	N/A	Allows for easier connection of electronic devices (Resistors, mosfets, cables)
4	5	Female-Male Cables	DuPoint	Used to connect the Launch kit to the bread board
5	6	Male-Male Cables	DuPoint	Used to connect different nodes on the bread board to each other
6	3	530 resistor	N/A	Resistor to limit current back flow into the Launch kit. Specifically calculated to allow exact amount of current according to TI specifications, with 30 extra ohms of buffer
7	1	RGB LED	N/A	LED with three separate LED’s inside it of colors Red, Green, and Blue
8	3	2N7000-F20	ON Semiconductor	Mosfets used to create a low side switch to prevent blinking when the processor is reset