Milestone 1 Report - Boaga Cig

Nolan Foy, Matt Wolf, Joseph DiBenedetto
ECE 09321 — Rowan University

October 22, 2018

1 Abstract

Over the first half of embedded systems, many attributes and concepts have been taught, discussed, and tested. Various qualities and features of the MSP430 microcontrollers (as well as the MSP432) have been experimented with and investigated. For the most part, we have been using the features of these microcontrollers to blink an LED; such as setting the output pins, enabling timer interrupts, and setting PWM duty cycle values. Milestone 1 would combine all of these attributes to control an RGB LED. Instead of simple Code Composer debugging and running, a way to communicate/send signals to the desired microprocessor had to be utilized. Thanks to the built in UART hardware of the MSP430F5529 board (The one we chose), the transition to UART programming was a smooth transition. Through choosing the correct COM ports and selecting the proper length each message byte would be for each node, successful color changes of the LED to red, green, blue, and white were achieved.

1.1 Design Features

Milestone 1 had the following design features:

- Utilizes the MSP430F5529 microprocessor's ability to receive messages over UART
- Connections from the F5529 to the RGB LED are made via breadboard circuitry wiring
- RGB node remains in low power mode until the UART message is received
- Reads and pulls necessary information from UART message
- Converts the first 4 bytes of the UART message into RGB values and information to create a transmission message

- Lights RGB LED to values received from the UART message. These values signify either red, green, blue, or a mixture of the colors (RealTerm)
- Transmits modified message over UART to next node

1.2 Featured Applications

The functions of Milestone 1 are capable of being used in the following applications:

- Change colors of an RGB LED. This includes color mixes such as white and purple that can be done by modifying the UART message to have two colors on at same time
- Modify the brightness of an LED on a range of 256 values. This is done via manipulation of the various PWM duty cycles of each color of the RGB LED
- Use multiple milestone devices in a grid or row to produce any type of color pattern through UART messaging

1.3 Design Resources

This device functions via the main.c code on the Code Composer project file ”Milestone1”. This code can be seen in the link below.

RGB LED F5529.

1.4 Block Diagram

The first of the following two block diagrams show the system of how multiple RGB LED nodes would be connected through the UART hardware. The second diagram shows how one node would feed UART data to it’s ’R’, ’B’, and ’G’ LEDs. As shown in block diagram 2, the duty cycles of each of the LEDs are being manipulated.

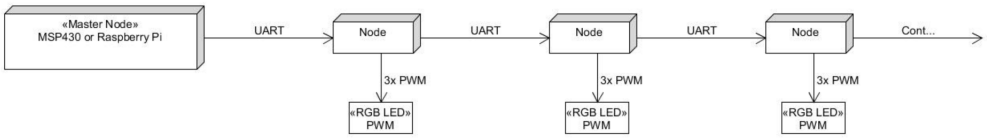

Figure 1: Node Network Block Diagram

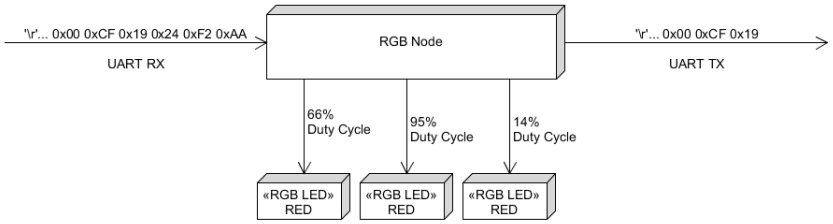

Figure 2: Individual RGB LED Block Diagram

1.5 Board Image

Shown in the following three figures below are the various circuitry wiring and connections made between the F5529 and the breadboard. More information on the pins used and their outputs can be found in the system description section.

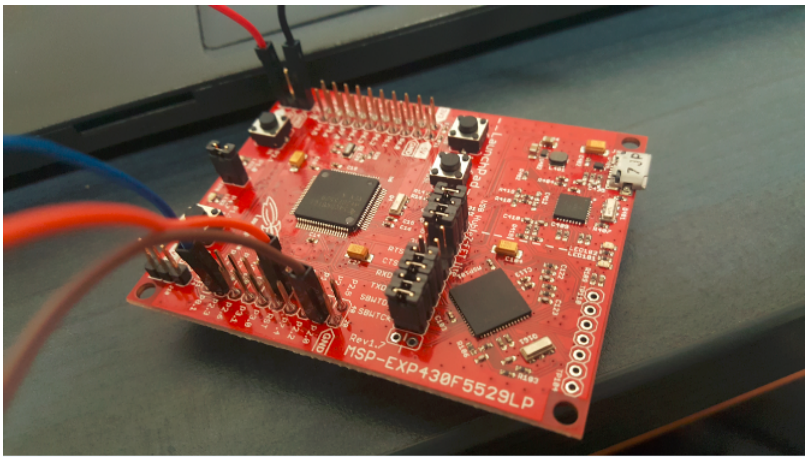

Figure 3: Close up of the MSP430F5529 board and the pins utilized

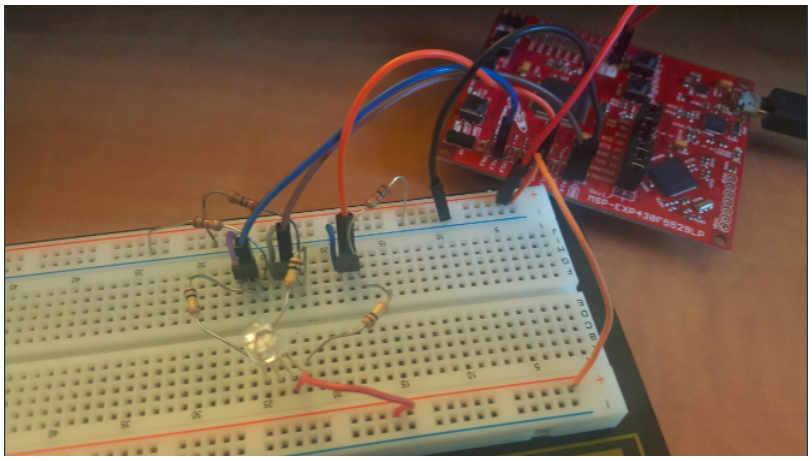

Figure 4: RGB LED Circuit constructed on Breadboard (View 1)

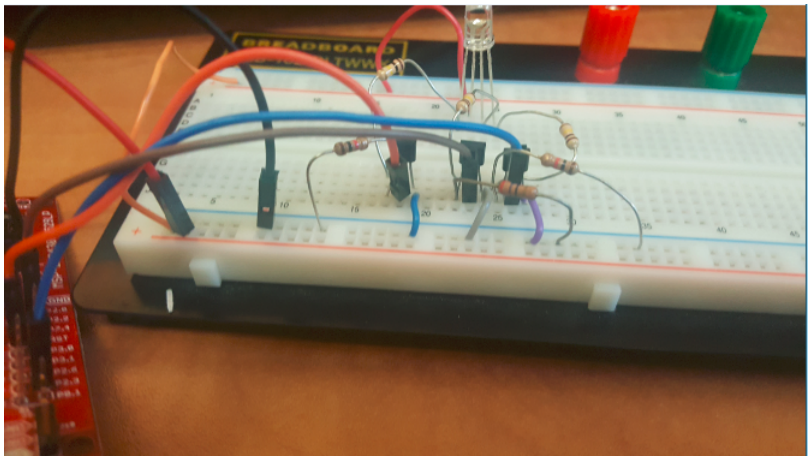

Figure 5: RGB LED Circuit constructed on Breadboard (View 2)

2 Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Microprocessor	MSP430F5529	Microprocessor needed for corresponding code parameter
RGB LED	Common Anode	The given RGB LED Specification
RealTerm	UART	Program for communication with microprocessor
BS250	MOSFET	P-type MOSFETS used in circuit
1 and 10k	Resistors	Resistors used in circuit

3 System Description

The application of illuminating an LED to a specific desired color is a function that all of our eyes see everyday. The system featured in this project constructs a single instance of this, but was also built with the capability of being strung to other LEDs to create a predetermined pattern of colors. While this setup is not exactly how the familiar LED screen illuminates all of its pixels, the ability to successfully produce any string of colors at this level is a majority of the battle. At the base of either scenario is a signal with instructions for the LEDs; we see this in this system with the UART message. From the arrival of this message, the LED is awoken from low power mode. With the assistance of the microprocessor program, and the various pin assignments to the three pins of the LED, the specified color is shown on the diode. The three pins chosen for the F5529 were pin 1.3 for red, pin 2.0 for green, and pin 2.3 for blue. (To see the actual instantiation of the pins, refer to the Design Resources section) Finally, the remainder of the signal is transmitted onto another LED node; where the same happens with a different desired color and a pattern is produced.

3.1 Detailed Block Diagram

Figure 6 shows an in depth look at a simulation circuit, which is a replica of what was used on the breadboard when the RGB LED was tested. 3 BS250 P-type MOSFETs were used in the circuit. As shown in the block diagram, each MOSFET is signified as either red, green, or blue. This is where the jumper cables would go into the breadboard. The jumper cables would jump from the gate of the MOSFETs to the pins on the F5529 board. Since the circuit's power source was from the F5529 itself, the top voltage source is signified as the actual microprocessor. Resistors are used in the circuit to limit current flow and control the brightness of the LED (from a conceptual perspective).

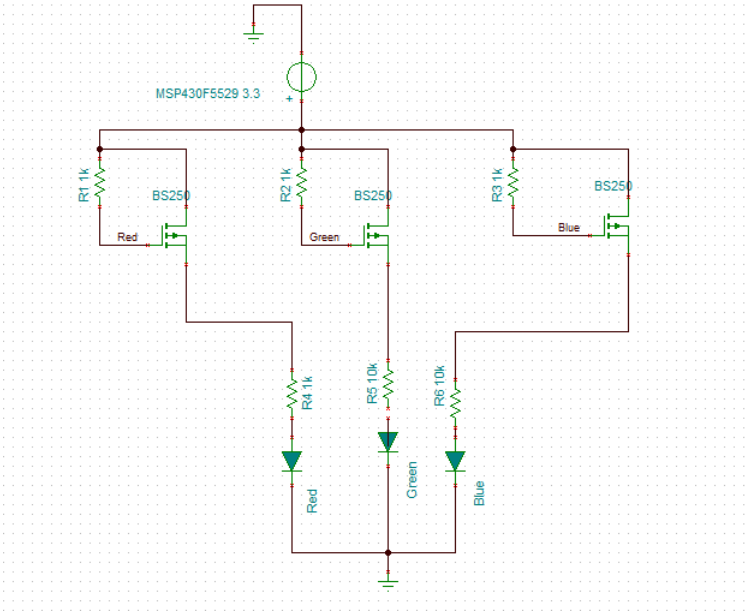

Figure 6: Circuit Model Built on Breadboard

3.2 Highlighted Devices

- MSP430F5529 - The written program is run on this board, enabling the reception and transmission of the UART signal, translation from signal to lit LED, and powering the circuit.
- LED + Circuitry - Powered by the microprocessor on a breadboard, the circuit of the RGB LED is powered, grounded, and receives signals to each of the 'R', 'G', and 'B' output pins.

3.3 Device 1: MSP430F5529

The processor in this project is the brain of the system. The features on the microprocessor used are the timers, interrupts, UART peripheral, and input/output ports. Interrupts are enabled so the program can perform an action only when a timer reaches a value and sends a signal to the processor; or when a UART signal is received. The timer peripheral and its registers provide the ability to set each LED to a specific brightness using PWM. The UART peripheral enables the use of a received signal buffer and a buffer of a signal to be transmitted. The I/O ports are set with the processor and connected to the pins of the LED, enabling each color to be illuminated to the proper brightness as necessary.

3.4 Device 2: LED + Circuitry

The LED and the comprising circuit were constructed on a breadboard. The breadboard was powered and grounded by the microprocessor and its 3.3 V and ground pins. The program set high values to three different I/O pins on the microprocessor whenever either of the three needed to be activated. These three pins were wired to the three 'R', 'G', and 'B' pins of the diode. These three pins–along with power, ground, the BS250 MOSFETs, and various resistors–enabled this whole system to function. More elaboration on these topics can be found in Section 4.

4 System Design Theory

This section goes into further technical detail about both of the major devices in use for this system. The previous section gave an overview of the microprocessor and breadboard circuit's major components functions, whereas here, each of these component's interactions with each other will be explained in detail.

4.1 MSP430F5529

The requirement of the processor in this project was to hold a program that received a signal over UART. This signal is a compilation of 80 bytes. The first byte of this string is the length of this string of bytes, so in this case it would be 80 (0x50). The next three are the values of each of your node's three LED pins. These are to be read and transmitted into values that will set the brightness of each light. Following the activation of each of the three lights, the signal to be transmitted over UART is very similar to that which was received; except the first byte to send out is the same of the packet received, minus 3, accounting for the three values your node used. This new first value will be followed by the remainder of the input bytes, and the entire output package is formed.

The UART message loads into the processor through the UCA1RXBUF register. This register acts like a buffer and allows for the program to process each incoming byte one by one. The first value read is the length of the packet in number of bytes, and is used as a stopping point in a software loop. The second, third, and fourth values are all set to the TA0CCR1, TA0CCR2, and TA0CCR3 registers. This sets the three output pins on if the value of these bytes is anything but zero, and creates a PWM for a specific brightness on each of the pins otherwise. The pins used here were 1.3, 2.0 and 2.3. Finally, the UCA1TXBUF register is loaded with the value first taken from UCA1RXBUF minus three, and then comes the remainder of the bytes that were loaded into the UCA1RXBUF register.

4.2 LED + Circuitry

The breadboard was required in our system in particular to hold, power, and wire the LED. The 3.3 V pin of the microprocessor was wired to the positive rails of the

breadboard, and the ground pin to the negative rails. Pins 1.3, 2.0, and 2.3 were wired to the red, green, and blue pins of the RGB LED, respectively. They were not connected directly but each through a series of two resistors and a MOSFET to avoid sending too much current through the LED. Each MOSFET was connected to ground as well as the fourth pin of the diode since the LED configuration was common anode.

5 Getting Started

Once the microprocessor is properly loaded with functioning code, and the circuit is constructed as previously pictured and explained, the whole system can function. Some specifics on how to get each device to fully function come in the following subsections.

5.1 How to Use the Device

The microprocessor must be plugged into a laptop where Code Composer Studio is open with the functioning code. The code must be debugged in order for the code to finalize on the processor. This can occur after all of the connections are made from the microprocessor to the breadboard where the LED circuit is constructed. To reiterate, the 3.3 V pin must be connected to the breadboard ground rails, ground pin to the negative rails, and pins 1.3, 2.0, and 2.3 have to be wired to the resistor to MOSFET setups before reaching the pins of the RGB LED. Once all of this is wired properly, the program of your choice can send UART messages as described below.

5.2 Software

The software needed to use this milestone device is Code Composer Studio (CCS, mentioned above) and either the program PUTTY or RealTerm. To reiterate CCS is used to generate the code/create the functions to be programmed into the microprocessor. Once the code is completed, it can be debugged and ran in the microprocessor.

5.3 Communicating with the Device

While there are two programs that can be used to communicate with the F5529 microprocessor (PUTTY and RealTerm), RealTerm is the recommended program, since it is able to send concise bytes of data to the processor. In order to properly use RealTerm, the settings must be completely accurate and must reflect certain settings of the processor that is used. The baud rate and COM port are two important settings to mention. The baud rate for this device should be set to 9600, and the COM port must match up with the port of the processor. By going to the ”Device Manager” on your computer, you will be able to find the COM port by looking under ”Ports” of the USB hooked up to your computer. Listed next to the processor USB port will be the COM port number.

Once all of this information is found, the settings, as mentioned above, must be changed. For starters, the messages sent to the processor should be set to hexadecimal. Under the Display tab, in the selections ”Display as”, select Hex[space], and check the box Half Duplex. Once done, go to the Ports tab and change the baud rate to 9600 and the COM port to the number that was found in device manager (as explained in the above steps). Once done, hit ”Change”, and the settings are now properly configured to begin communication with the microprocessor.

6 Test Setup

Before setting up a network of multiple nodes, the individual node must be tested to verify it can handle each color correctly, as well as different brightness's of a color. The input to enter in RealTerm to test these different functionalities are as follows:

- '0x03 0xFF 0x00 0x00' - Tests red
- '0x03 0x00 0xFF 0x00' - Tests green
- '0x03 0x00 0x00 0xFF' - Tests blue
- '0x03 0xFF 0xFF 0xFF' - Tests white
- '0x03 0x7F 0x7F 0x7F' - Tests white at half brightness
- '0x05 0xxx 0xxx 0xxx 0xxx 0xxx' - Tests ability to receive and process color as well as transmit remaining

These separate lines of code are the bytes. The first Byte, 0x03, specifies the amount of bytes in the package. Because there are three colors in the RGB LED, the first byte would be red, second green, and third blue. 0xFF is the byte code to set the LED on at a 100 percent duty cycle (from 0 to 255). This achieves max brightness of the LED in the respective color. By triggering all of the colors at once, the white color would be activated. 0X7F will cause the LED to be at a 50 percent duty cycle, or half brightness.