Milestone 1 Application Note

Richard Dell and Nick Riggins
Rowan University

December 9, 2019

1 Design Overview

For this project the goal is to be able to communicate between two MSP430G2553 boards and enable them to control a different circuit from each of them. To do this, code must be written and a circuit would be created that would be able to change the color and brightness of an RGB LED by utilizing the MSP430G2553. The code controls the PWM and the circuit drives the LED. In addition to controlling the LED of the first MSP430G2553, it must also be able to send a message to another LED connected to the first, and control that LED's color and brightness as well. To control the color and brightness, a message containing hexadecimal values for red, green, and blue would be sent through UART from a laptop using realterm to the first MSP430G2553. This message would contain the desired PWM value that the MSP430G2553 must give to the RGB LED in order to output the correct color and brightness. This PWM signal would change the amount of time that the LED would turn on and off, essentially changing the brightness of the color that it is controlling.

1.1 Design Features

These are the design features:

- Utilizing the UART capabilities of the processor
  - Receiving bits and executing code
  - Altering the received bits and sending the new bits to a different processor
- Communication between nodes

1.2 Featured Applications

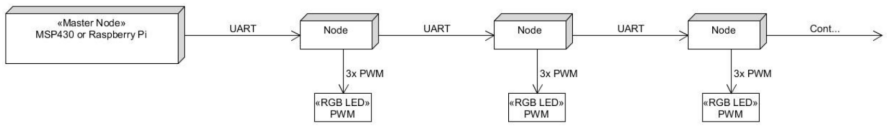
These are the featured applications

- Altering Duty Cycle of RGB LED
- Receiving bit and executing code with received bits
- Altering received bits
- Sending new bits to another board

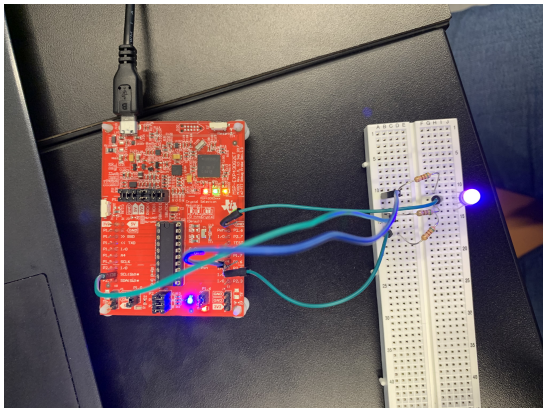
1.3 Design Resources

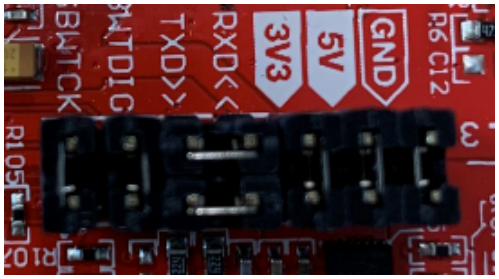
GitHub repository link: https: `//github.com/Intro-To-Embedded-Systems-RU09342/milestone-1-rigginsn7/tree/master/Milestone_StrangerThings`

1.4 Block Diagram



1.5 Board Image





2 System Description

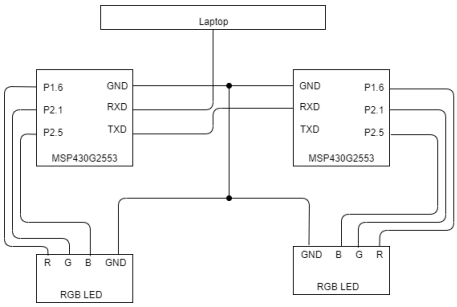
The problem given was to create code for the MSP430G2553 that would be able to send and receive bytes of information between two processors that would be used to alter the color and brightness of two RGB LEDs, one for each MSP430G2553. Once this was done, a circuit must be created that would connect the TXD and RXD pins of

the processors together with a wire, as well as connect the output and ground pins on the processor to their respective positions on the LEDs.

3 Key System Specifications

Parameters	Specifications	Details
RGB LED	Control the color and brightness of the RGB LED	Utilize hardware PWM to change the duty cycle of the red, green, and blue pins of the LED based on the received transmission
UART transmissions	Send and receive UART transmissions across processors	Sending and receiving a number of bytes between two or more processors that can be used to execute specific functions within the code

3.1 Detailed Block Diagram



3.2 Highlighted Devices

These are the highlighted devices

- MSP430G2553
- RGB LED

The MSP430G2553 is the processor used to execute the code and control the RGB LED. The code was written in C language and is executed on the MSP430G2553. The processor took the received bit that was send from a computer and used that the light up the LED to what ever the bits corresponded too. The bits represented the duty cycle of red, green and blue parts of the led.

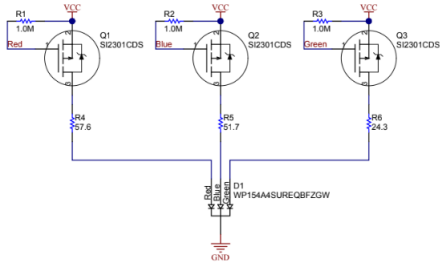
3.3 Device 1: MSP430G2553

The MSP430G2553 is the processor used for this milestone. The code that was put on the processor was written in C. The pins on the processor had to be switched to enable sending and receiving bytes using its UART capabilities (To see how the pins are switched refer to the board image section). Realterm was used to send the initial bits to the first board(Information on this can be seen in the ”communicating with the device” section.). The order the bytes must be sent so that the first byte tells the processor the length of the transmission, the second byte is the duty cycle of the red LED, the third byte is the duty cycle of the green LED, and the fourth byte is the duty cycle of the blue LED. After the processor executes those the first byte's value is lowered by 3 and every thing after the first byte is shifted left 3 times and that new transmission is now the initial transmission length minus 3 and that new transmission is sent. The last byte is just to signal that the transmission is done and it 0X0D every time. A diagram of this can be seen below.

Byte 1	Byte 2	Byte 3	Byte 4	Bytes 5 through N	Final Byte
Initial length of transmission	Red pwm	Green pwm	Blue PWM	RGB pwm for next boards	End transmission

3.4 Device 2: RGB LED

An RGB LED is a combination of 3 LEDS. The three LEDS are red, green, and blue. This makes it possible to produce almost any color by adjusting the duty cycle of each color which would change the intensity of the color. The duty cycle just determines for the amount time the LED is on. The duty cycle of each color was changed using a PWM signal. The PWM signal was determined by the input bits received from the processor.



4 SYSTEM DESIGN THEORY

The design of this project as a whole can be split into two main components, the UART transmission and the PWM signal. In order to perform the task at hand, the micro controller must first receive 8 bytes of information through UART. The microprocessor parses the received bytes and uses a subset of those bytes to set the PWM duty cycle that drives each of the three LEDs, resulting in the display of the desired color. The

remaining bytes, with the most significant bits adjusted with representing the message length, are then transmitted to the next MSP430G2553 board. This process would be continued down a string of micro controllers until the transmission ended, or the final board was reached.

4.1 Design Requirement 1

The first requirement of the project was to be able to send and receive bytes of information through UART that could be used throughout the other parts of the code. The received transmission would be split into 5 parts. The first byte would indicate how many total bytes are in the transmission, the second, third, and fourth bytes would indicate the brightness of the red, green, and blue colors respectively, and the final part would indicate the end of the transmission(a chart what information each byte contains can be seen in the ”System Description” section.). A switch case statement was used to separate which code to execute based on which byte number was being read. If the transmission was more than 5 bytes, the system would alter the transmission signal sent it to the next micro controller. In order for the next micro controller to be able to use the sent information effectively, the transmission would have to be changed by removing the first four bytes corresponding to the length of the transmission, and the red, green, and blue colors that were used by the first micro controller, and set them to the length of the new transmission followed by the bytes that were not removed from the original transmission. To do this, the length of the original transmission was stored upon receiving it, and the new transmission would be the original length reduced by the three bytes that were used (For example, if the original transmission was 8, the new transmission would be 5). After the new length was sent, all that had to be done was to send the rest of the bytes after the fourth byte in the original transmission straight to the transmit pin for the next micro controller.

4.2 Design Requirement 2

The second main part of the project was the PWM signals for each color of the RGB LED. In order to do this, two different timers within the MSP430G2553 were used to set up different duty cycles for each of the colors. Each timer was set to count up to a maximum of 255, the maximum value possible from one byte, and then trigger an interrupt for the rest of the code. In addition to triggering an interrupt, the timers were also set to a new output mode that would set a value upon reaching the first interrupt, and then reset the value upon reaching the next interrupt. Once the interrupts for the maximum PWM were set, new interrupts that would control the duty cycle of each of the colors were to be set up with the red duty cycle being on the first timer, and the green and blue duty cycles being on the second timer. The values for each of the colors' duty cycles would come from their respective transmission bytes. The received values would be set to the new interrupt so that the timers would set and reset the output of each color's pin at the desired duty cycle. Changing the duty cycle would allow the brightness of each color to be changed to whatever value is desired in the received transmission.

5 Getting Started/How to use the device

These are the steps you should take to implement

- Connect device to PC

  Insert the USB cable that is included with the MSP430G2553 into both the micro controller and PC.

- Make sure MSP430G2253 send and receive pins are turned on

  Two jumpers that are connecting RXD and TXD pins on the board must be switched from connecting the pins vertically, to connecting the pins horizontally.

- Connect the red pin on the MSP430G2553 to the red portion of the RGB LED

  P1.6 on the msp must be connected to the leftmost pin on the RGB LED

- Connect the green pin on the MSP430G2553 to the green portion of the RGB LED

  2.1 on the msp must be connected to the left most pin on the RGB LED

- Connect the green pin on the MSP430G2553 to the blue portion of the RGB LED

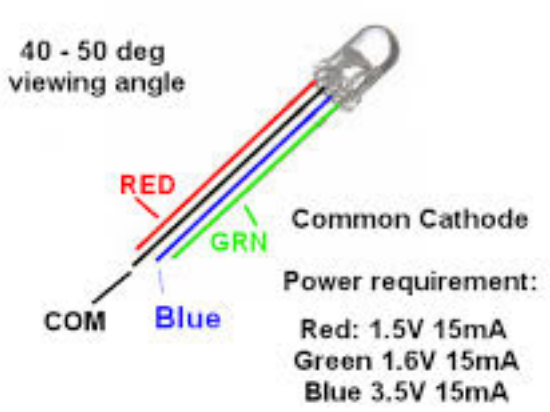
  2.6 on the msp must be connected to the second right most pin on the RGB LED

- Make sure the processors and RGB LED are both grounded

  The ground pins on all of the MSP 430G2553 devices and RGB LEDs must be connected to each other

- Connect the TXD pin of the first processor to the RXD pin of the next processor

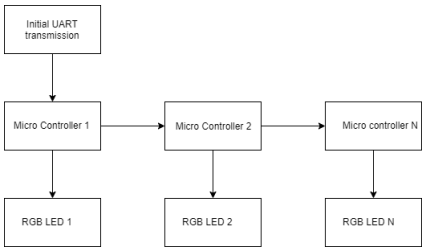
  The TXD pin of the first MSP430G2553 must be connected to the RXD pin of the second pin. This process would then be continued for the second TXD and third RXD pins, and so on for as many boards as there will be connected.



6 Getting Started Software/Firmware

The code for the MSP430G2553 was written in C code. After the code was loaded on to the processor chip then RealTerm is used to send the initial bits to the board and then the boards start to interact with each other.

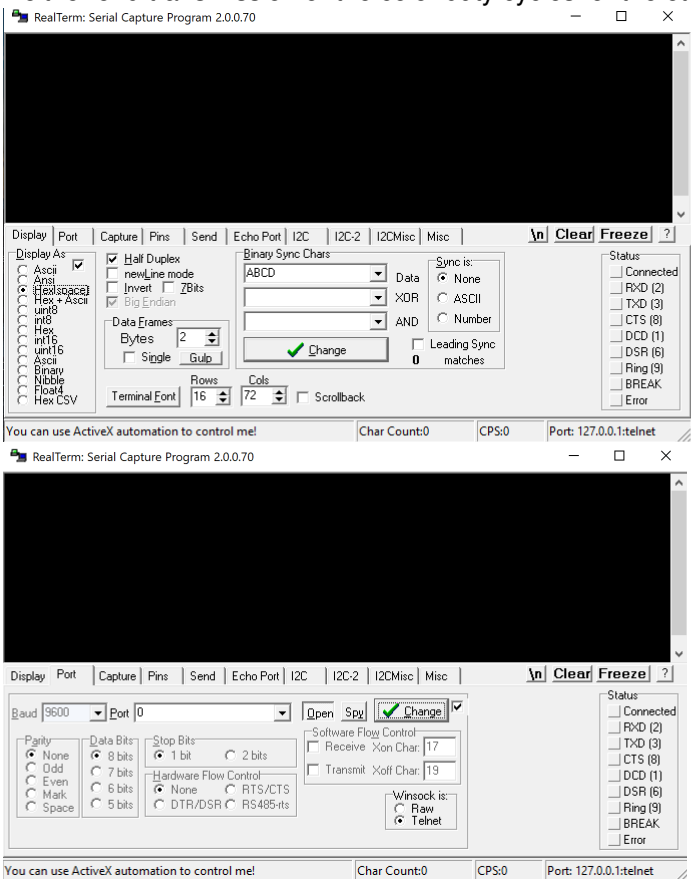
6.1 Hierarchy Chart



6.2 Communicating with the Device

In order to communicate successfully with the device and send it a transmission the software Realterm is used. Several things must be changed in Realterm for the software to communicate with the micro controller. First in the display tab, the display as setting must be changed to Hex[space] and half duplex must be checked off. In addition to this, under the port tab the baud must be changed to 9600, the open button

must be pressed down, and the port must be changed to the port that the micro controller is connected to. Once these are done, a hexadecimal signal can successfully be sent to the micro controller. The order of the bytes that are sent must be the total number of bytes first, followed by the red duty cycle, green duty cycle, blue duty cycle, then either end transmission or the color duty cycles for the other micro controllers.

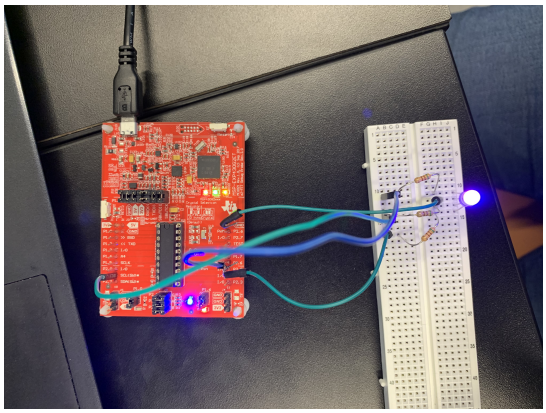


7 Test Setup

To test the project, the circuit must be set up first. The red, green, and blue pins of the LED must be connected to a resistor which is then connected to the correct pin for each color. The corresponding pins for red, green, and blue are P1.6, P2.1, and P2.5 respectively. After the pins are connected, the grounds of all micro controllers must be connected to the ground pins on all the LEDs so that all of the ground pins are connected to each other. Next, the TXD pin of the micro controller that is receiving the initial transmission must be connected to the second micro controller's RXD pin. Once all of the wiring is done, the code can be flashed onto each board and Realterm can be set up to send the initial transmission (for information on Realterm set up, please

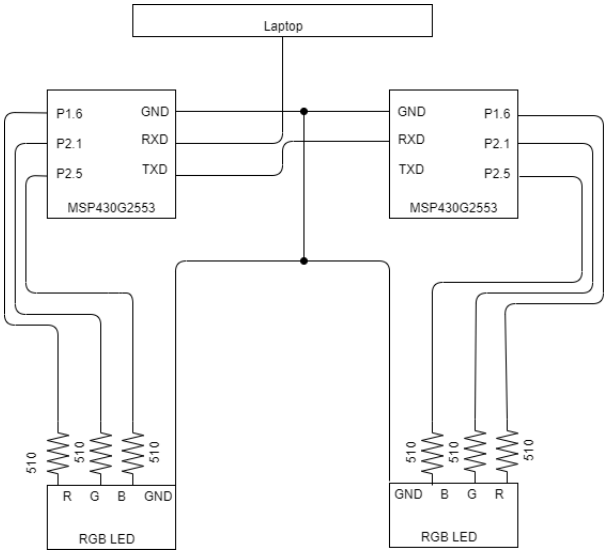
refer to the ”Communicating with the Device” section.). Specific values that are easy to see should be used at first to be able to see if each of the three LED colors are working probably.

7.1 Test Data



8 Design Files

8.1 Schematics



8.2 Bill of Materials

Item #	Description	Quantity
1	MSP430G2553	1
2	RGB LED	1
3	510 ohm, 1/4 watt, 5% tolerance resistor	3

8.3 Notes

Some devices can have a faulty baud rate, and because of this they can send incorrect transmissions even if they receive the correct transmission initially. To fix this, baud rate trimming can be added to the code to correct the faulty baud rate.

- More information:

  https://www.allaboutcircuits.com/technical-articles/the-uart-baud-rate-clock-how-accurate-does-it-need-to-be/

- Example Code:

  https://timmurphy.org/2009/08/04/baud-rate-and-other-serial-comm-settings-in-c/