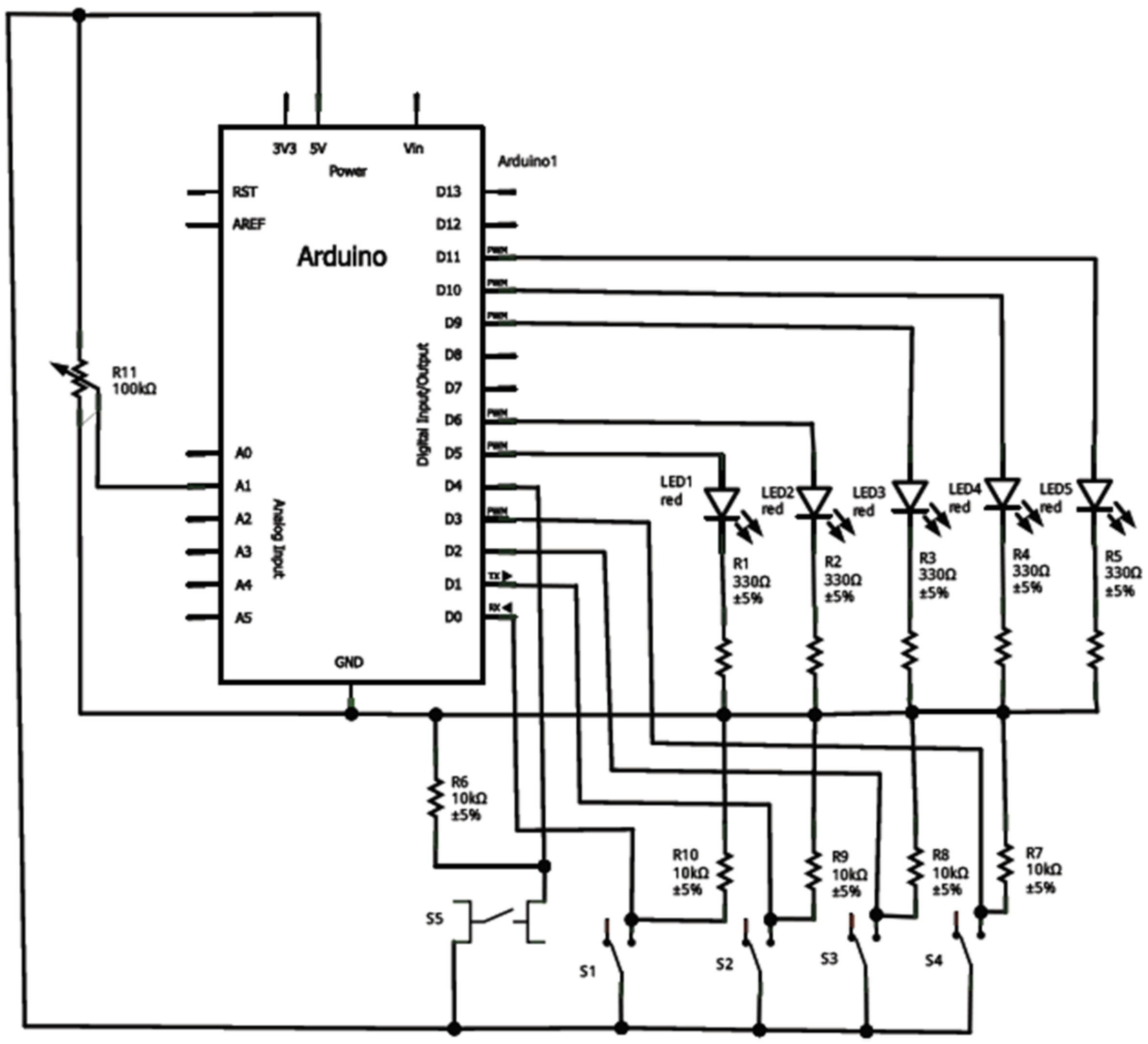
CG2271 Real Time Operating Systems

Tutorial 4

Question 1

This question carries on from last tutorial, where you built a binary to BCD converter. A possible solution to Questions 1a and 2a is shown below:



In summary, in last week's tutorial you were asked to:

i. Implement a binary to BCD converter.
ii. Add in code to read the potentiometer and write the value to a global variable called adc_in.
iii. Add in a timer so that the switches are read and LEDs updated every 5 ms instead of whenever the push-button S5 is pushed.

For this week, you will modify the code so that you control the LED brightness using a 200 Hz PWM frequency (or something as close as possible).

a. Complete the following table showing the mappings between PWM the Uno's digital/PWM pins, the associated timer output pin, and the timer connected to that pin. The second entry has been filled for you.

Arduino Pin Number	Timer Output Pin	Timer
5		
6	OC0A	TIMER0
9		
10		
11		

b. What is a suitable prescaler value for this program? Can it be achieved? If not what is the closest PWM frequency that you can get?

c. Modify your program from Tutorial 3 so that the LEDs are fully off when adc_in is 0, fully on when adc_in is at its maximum value, and intermediate levels of brightness in between.
Note: You can use any timer to implement part c, regardless of whether you used them previously in Tutorial 3. This doesn't work in the real world but our objective here is to see whether you understand PWM programming.

Question 2

What are the relative advantages and disadvantages of round robin, round robin with interrupts, function queue scheduling and RTOS-based software architectures? What sorts of applications are best suited to each of these architectures?

Question 3

Through Google or any other means, research on the INT0 and INT1 external interrupts available on the Atmega microcontrollers.

a. Which Arduino pins do INT0 and INT1 connect to? (Hint: There is an Atmega data sheet in the Lectures folder).

b. Design a circuit so that the pin on the Arduino corresponding to INT0 is pulled LOW when a push button is pressed, and is held high when the button is not pressed. See the "electronics.pdf" file in the Lab folder in IVLE for details of how to use a pull-up resistor. In addition up a photoresistor to analog channel 2 and an LED to digital pin 13. Your circuit must be properly designed with the proper protection resistors, voltage dividers, etc.

c. Write a program that toggles the LED on and off with each button press. I.e. turn on the LED the first time the button is pressed, turn it off the second time, on the third time, etc. You must use INTERRUPTS to do this, not simple GPIO programming. Research on how to set up and handle INT0 properly.

d. In Tutorial 2 you looked briefly at how to do debouncing. Set up Timer 0 and use it to debounce the push button.

We will extend on this question in Tutorial 5.