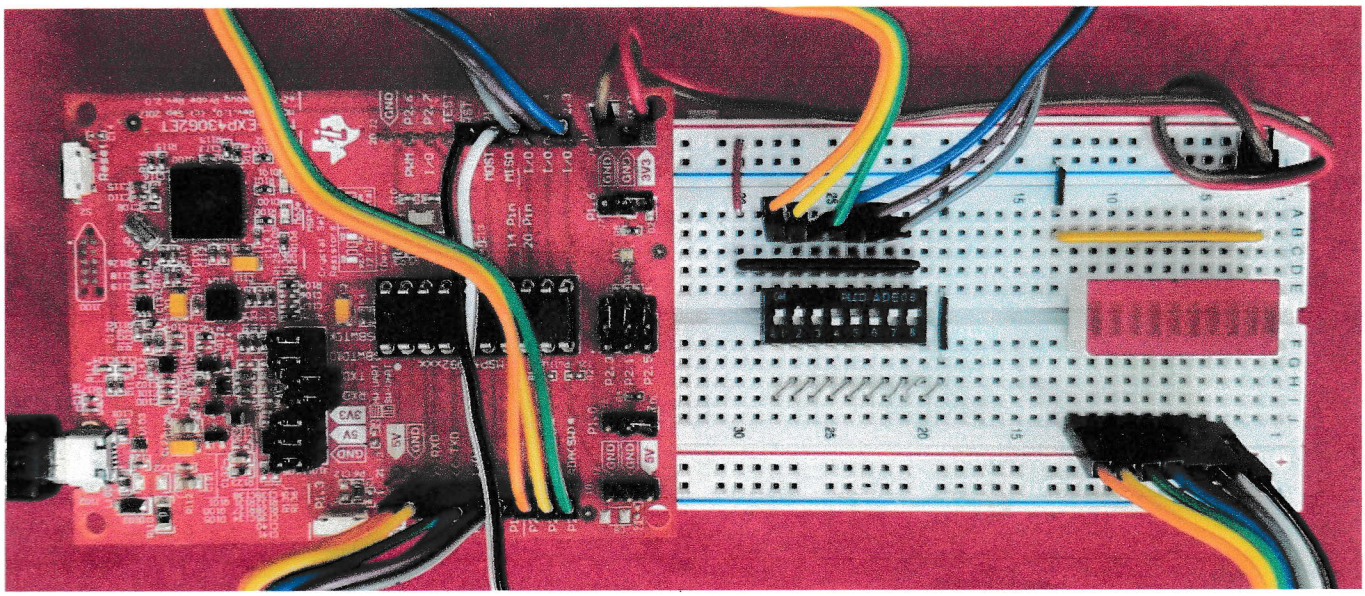
```c
#include <msp430.h>


/*v1*
 * main.c
 */
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer

    int R5_SW=0, R6_LED=0;
//  ^read input, ^ read output
        P1OUT = 0b00000000;      //mov.b    #00000000b,&P1OUT
turn off all leds
        P1DIR = 0b11111111;      //mov.b    #11111111b,&P1DIR
set all port 1.0-1.9 to outputs (LEDs)
        P2DIR = 0b00000000;      //mov.b    #00000000b,&P2DIR
set all port 2.0-2.9 to inputs  (Switches)
        P2REN = 0xFF;            //enable all resistors on port 2
P2OUT now uses it's secondary function (so P2OUT does NOT affect
outputs)
        P2OUT = 0xFF;            //make them all of them pull ups

        while (1)
        {
            //read the states of the switches
            R5_SW = P2IN;        //mov.b    &P2IN, R5            Store
input pattern in R5_SW

            //checking P2.0 for read mode
            if (!(R5_SW & BIT0))   //Check if P2.0 is up
              {
                R6_LED = R5_SW & (BIT3 | BIT4 | BIT5);  // mask the
pattern bits    Read display pattern
                P1OUT = R6_LED;                         // display the
pattern
              }

            //else, it is the rotate and display mode
            else    //It is in rotate mode
              {
                //The following code (3 lines) only toggles the
pattern of the 8 LEDs
                //This is to show you got all your switches and LEDs
working properly
```

```
                //Modify the code so that the last pattern (3 bits)
read during the
                //reading mode is now rotating on the 8 LEDs left to
right base on P2.1

                if (!(R5_SW & BIT1))   //if 2.1 is up
                    R6_LED = (R6_LED >> 1) | (R6_LED << 7); //Rotate
RIGHT, in C
                }
                else
                {
                   R6_LED = (R6_LED << 1)  | (R6_LED) >> 7; //Rotate
LEFT, in C
                }
                    R6_LED &= 0xFF;     //mask any excessive bits
Make sure to keep the value 8-bit
                    P1OUT = R6_LED;     //pattern out - display it

             }

               if (!(R5_SW & BIT2))  //If 2.2 is up, pattern is slow
               {
                    __delay_cycles(80000);    //slow
               }
               else  //Else it is fast
               {
                    __delay_cycles(400000); //fast
               }


        } //end while




    return 0;
}
```

Name/Semester: Aaron Mills **Grade:** /20

The MSP430 microcontroller is designed to allow port pins to be individually configured as digital input or digital output and in the case of Port1, all pins P1.0-P1.7 could also be configured as analog inputs. Some pins can be used as analog outputs (analog write). Finally some pins have specific functions, such as serial communication SPI. See the TI MSP430G2553 user's manual for complete description of all pins functionalities.

Assemble the circuit below making sure to follow the directions given during the class lectures. For digital inputs we are using six (6) dip switches connected to port Port2 pins 0-5. If your dip switches have more than 6 positions, ignore the unused ones. Please note the use of external pull-up resisters (1KΩ) with the switches as they are all configured as active-low. When a switch is turned on, it asserts a logic zero (GND) on the port pin. When the switch is off, the pull-up resistor will make the micro pin floats high. For resisters, you can use discrete ones or a resistor network as shown in the photos below.



For digital outputs we are using eight (8) LEDs connected to Port 1 pins 0-7. Please note in the photo, we used 10-segment LED bar, we connected 8 of the LEDs and ignored the two on the right. In your kit, you may have discrete LEDs, so you can line up over the breadboard valley. Also note in the photo, we used a resister network for the current limiting resistors (330Ω) for the LEDs on the cathode (GND) side of the LEDs. The resister network has its common bus connected to the GND. Eight (8) wires are used to connect the anode side of the LEDs to the micro pins Port1, so the LEDs are configured as active high. A logic high signal (VCC) from the port pin will turn the LED on. For all the connections form the breadboard to the launchpad, we used Female-Male jumper cables.

If you choose to take MSP430G2553 chip off the Launchpad board and insert it on your breadboard directly, make sure you use a 3.3VDC supply on VCC pin. Also a Reset resistor of 47KΩ must be used to connect Reset pin to VCC.

Name/Semester: Aaron Mills **Grade:** /20

[20] 4) For this lab, you need to write a program in C that reads a 3-bit desired light pattern from the 3 input switches connected to pins P2.3-P2.5 and displays the pattern on the 8 LEDs (pins P1.0-P1.7) with some "light playing" options based on the input from other 3 switches (pins P2.0-P2.2), as described below. You should start with the skeleton code provided in your course Canvas site. Before you make any changes to the skeleton code, make sure all your hardware pieces are properly functioning. The following is how the program should be working, some of these requirements are already in the skeleton code.

[8] 4.a) Switch on Port2.0 (Read/Rotate mode)

- On (logic 0) – Read mode: Your program continuously reads the switches and takes only the 3-bits (Port 2.3-2.5) as a desired display pattern and displays them on the LEDs (Port 1.3-1.5). All other LEDs on Port 1 are masked to zero (turned off). Already in the skeleton.
- Off (logic 1) – Rotate mode: Once this switch is turned off, your program will rotate the pattern on the 8 LEDs connected to Port1 pins 0-7. The pattern is the 3-bit value you constantly updated and saved during the Read mode.
  - The skeleton code implements the Read mode, but for the Rotate mode, it would simply toggle the state of the 8-bit LEDs, part of which is the 3-bit pattern. You need to change the code so the pattern rotates on the 8 LEDs.

[6] 4.b) Switch on Port2.1 (Left/Right direction mode)

- On (logic 0) – The LED pattern rotates to the Left.
- Off (logic 1) – The LED pattern rotates to the Right.
  - The skeleton does not have code for this switch, but you can follow the way the skeleton reads the state of first switch. The C language does not have a rotate function, by you can implement it using shift and OR functions.

[6] 4.c) Switch on Port2.2 (Fast/Slow speed mode)

- On (logic 0) – The pattern rotation is Fast.
- Off (logic 1) – The pattern rotation is Slow.
  - Hint: use software delays.

[2] 4.d) **[BONUS]** Modify the program so that the two Switches on Port2.0-1 provide four different fun patterns which continuously display/move on the 8 LEDs. Keep Switch on Port 2.2 to change speed. ***Please be creative.***