Demonstration:

1. When S1 (J4.33) of the MKII board is pressed, demonstrate blinking of blue LED (J4.37) based on the given series (Requirements 1, 5, 6, and 7)

2. When S2 (J4.32) of the MKII board is pressed, demonstrate blinking of green LED (J4.38) based on the number of presses (Requirements 2, 5, 6, and 7)

3. Demonstrate the generation of beep when switch S1 (J4.33) or S2 (J4.32) is pressed and demonstrate the resetting of the given series count and the press count to zero when the push button S1 (P1.1) is pressed. (Requirements 3, 4, 6, and 7)

Requirements:

1. When the push button S1 (J4.33) of the MKII board is pressed, blink (on and off) the blue LED (J4.37) based on the series (n^2+1)/ 2 where n is the number of presses. (1 st press → 1 blink, 2nd press → 2 blinks, 3rd press → 5 blinks and so on)

2. When the push button S2 (J4.32) of the MKII board is pressed, blink (on and off) the green LED (J4.38) based on the number of presses. (1st press → 1 blink, 2nd press → 2 blinks, 3rd press → 3 blinks, 4th press → 4 blinks, and so on)
3. Each time when the switch S1 or S2 is pressed, generate a beep using the buzzer

4. When push button S1 (P1.1) is pressed on the MSP432 board, the series count for push button S1 (J4.33) and the press count for push button S2 (J4.32) should be reset to zero

5. A new push event on push button S1 (J4.33) or S2 (J4.32) will be done only after the blinking event is completed

6. The blinking should be accomplished using delay functions

7. Write required software routine to handle switch debouncing and do not use any delay loops or delay functions for handling switch bouncing.

Learning Objectives:
The general learning objectives of this lab were to introduce the MSP432 board and booster pack from Ti and the energia software tool. This lab also gives you hands-on experience to use the push buttons and the RGB LED.

General Steps:
The general steps needed to complete this lab were . . .

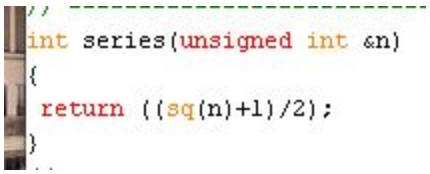
1. Configure S1 to accept input, blink based on the series function, (n^2+1)/2 while generating a single confirmation beep

2. Configure S2 to accept input, blink based on the counter while generating a single confirmation beep
3. Configure the Reset P1 button to accept input, generate confirmation tone, and reset Both button counters to zero.

Detailed Steps:

Some detailed steps to complete this lab were . . . .

1. I created a function for the Buttons that uses a for loop for the number of blinks, tone generation, and pass parameters into the function so the button responds uniquely based on the inputs.
2. Create a Series function for the (n^2+1)/2 which accepts an unsigned int from the counter to transform the output based on the series, that gets inserted into the button for loop function
3. reset button just resets the counters for both buttons and generates one confirmation tone



```
// ---------------------------
int series(unsigned int &n)
{
 return ((sq(n)+1)/2);
}
```

```
if (buttonOneState == LOW)
{

  cnt1++;
  Serial.println(cnt1, DEC);
  button(buttonOneState, ledBlue, ledState1, cnt1, button1);
}

if (buttonTwoState == LOW)
{
  cnt2++;
  button(buttonTwoState, ledGreen, ledState2, cnt2, button2);
}

if( resState == LOW)
 {
   Serial.println(res,DEC);
 tone(buzzer, freq1 , 100);
// tone(buzzer, freq2, 100);
   cnt1 = 0;
   cnt2 = 0;
```

```
void button(int &buttonState,const int &ledPin, int &ledState, unsigned int &counter, unsigned int &which)
{

  if(which == 1)
  {
   led = series(counter); // where led is the number of blinks after (n^2+1)/2 function
  }
  else
  {
      led = counter;
         Serial.println(led, DEC);
  }
   Serial.println(led, DEC);

   tone(buzzer, freq1 , 100);
   tone(buzzer, freq2, 100);
    for( int i = 0; i < led; i++) // LED blinks "led" times
    {
        // Blinks Led once
        if (ledState == LOW)
        {
          ledState = HIGH;
          digitalWrite(ledPin, ledState);
          delay(1000);
          ledState = LOW;
          digitalWrite(ledPin, ledState);
          delay(1000);
        }
     }
}
```

Observations:

Some important observations while completing/testing this lab were

1. Debouncing is necessary to reject unintended inputs
2. Confirmation beeps are necessary only to improve user experience
3. setting up the code modularly helps reusability of code if similar functionality is used for each button

Summary:

In this lab we learned how to map hardware to software and be able to interact with the hardware constant variables by using variable variables. We were able to perform several functionalities, such as receiving external input, implementing a debounce, generating a tone, and configuring a reset feature.