Servo:

```
#include "msp430fr2355.h"
#include <msp430.h>

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

    //PWM period
    P2DIR |= BIT1;
    P2SEL0 |= BIT1;  //selection for timer setting
    P2SEL1 &= ~BIT1;
    PM5CTL0 &= ~LOCKLPM5;
    while(1) {
    TB1CCR0 = 20000;  //PWM period
    TB1CCR2 = 500;  //CCR2 PWM Duty Cycle  !min 350 max 2600 angle 190,
    //350 2350-180 degrees
    TB1CCTL2 = OUTMOD_7;  //CCR2 selection reset-set
    TB1CTL = TBSSEL_2|MC__UP;   //SMCLK submain clock,upmode
    __delay_cycles(1500000);
    TB1CCR2 = 1200;
    TB1CCTL2 = OUTMOD_7;  //CCR2 selection reset-set
    TB1CTL = TBSSEL_2|MC__UP;
    __delay_cycles(1500000);
    }
}
```

RGB LED:

```
//            MSP430FR2355
//         ---------------
//     /|\|               |
//      | |               |
//      --|RST            |
//        |               |
//        |     P6.0/TB3.1|--> CCR1 - RED
//        |     P6.1/TB3.2|--> CCR2 - Green
//              P6.2/TB3.3| --> CCR3 - BLUE
//
//   Darren Lu
//   Texas Instruments Inc.
//   Oct. 2016
//   Built with IAR Embedded Workbench v6.50 & Code Composer Studio v6.2
```

```
//**************************************************************************
*



#include <msp430.h>
#include "RGBLED.h"

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop WDT


    initRGB();
    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;






    while(1) {
       setRGBLED(255, 0, 0);
        _delay_cycles(50000);


        setRGBLED(0, 255, 0);
       _delay_cycles(50000);

       setRGBLED(0, 0,255);
        _delay_cycles(50000);

    }

}
```

```
void initRGB(){
    P6DIR |= BIT0 | BIT1 | BIT2;                          // P6.0 - P6.2 output
    P6SEL0 |= BIT0 | BIT1 | BIT2;
    P6SEL1 &= ~(BIT0 | BIT1 | BIT2);                        // P6.0 - P6.2 select to
    TB3CCR0 = 1000-1;                          // PWM Period
    TB3CCTL1 = OUTMOD_3;                       // CCR1 reset/set
    TB3CCR1 = 750;                             // CCR1 PWM duty cycle
    TB3CCTL2 = OUTMOD_3;                       // CCR2 reset/set
    TB3CCR2 = 250;                             // CCR2 PWM duty cycle
    TB3CCTL3 = OUTMOD_3;                       // CCR3 reset
    TB3CCR3 = 250;                             // CCR3 PWM DUTY

    TB3CTL = TBSSEL__SMCLK | MC__UP | TBCLR;    //TBR




}
```

Single LED : #include "msp430fr2355.h"

```
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                 // Stop watchdog timer

    P5OUT &= ~BIT0;                           // Clear P5.0 output latch for a
defined power-on state
    P5DIR |= BIT0;                            // Set P5.0 to output direction

    PM5CTL0 &= ~LOCKLPM5;                     // Disable the GPIO power-on
default high-impedance mode
                                              // to activate previously
configured port settings


    while(1)
    {
        P5OUT ^= BIT0;                        // Toggle P1.0 using exclusive-OR
        __delay_cycles(100000);               // Delay for 100000*(1/MCLK)=0.1s
    }
}
```

Thermistor:

```
#include "thermistor.h"

Void configureThermistorADC(void)
{
     P1SEL0 |= BIT4;        // P1.4 = A4 = ADC input
     P1SEL1 |= BIT4;

     ADCCTL0 = ADCSHT_2 | ADCON;      //ADC = ON
     ADCCTL1 = ADCSHP | ADCSSEL_2;         //Tiner SMCLK
     ADCCTL2 = ADCRES;                     //12 bit
conversion
     ADCHCTL0 = ADCINCH_4;                 //A4 = P1.4
}

unsigned int readThermistorADC(void)
{
     ADCCTL0 |= ADCENC | ADCSC;
     While (ADCCTL1 & ADCBUSY);
     Return ADCMEMO;
}
```

Pilot (Solenoid):

```
#include <solenoid.h>

Void configureSolenoid(void)
{
        P2DIR |= SOLENOID_PIN;          //Set 2.5 as OUT
        P2OUT &= ~SOLENOID_PIN;                 //Must be OFF initially
}

Void solenoid_on(void)
{
        P@OUT |= SOLENOID_PIN;                  //Set 2.5 to HIGH
}
```

```
Void solenoid_off(void)
{
        P2OUT &= ~SOLENOID_PIN;             //Set 2.5 to LOW
}
```

Potentiometer:

```
#include <msp430.h>
#include <stdint.h>
#include "RGB_LED.h"

void pot_Init(void);
uint16_t pot_Read(void);

int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

  pot_Init();              // Initialize potentiometer (P1.5 ADC)
  initRGBLED();              // Initialize RGB LED PWM

  PM5CTL0 &= ~LOCKLPM5;       // Unlock GPIOs

  uint16_t potValue;

  while (1)
  {
    potValue = pot_Read();   // Read potentiometer value (0-1023)

    // Map potentiometer reading to RGB values
    char red = (potValue >> 2) & 0xFF;   // Scale ADC 0-1023 to 0-255
    char green = (~red) & 0xFF;          // Inverse of red
    char blue = (red >> 1) & 0xFF;       // Half-intensity blue

    setRGBLED(red, green, blue);         // Update RGB LED color

    __delay_cycles(50000);               // Small delay for stability
  }
}
```

```
// Initialize P1.5 as ADC input
void pot_Init(void)
{
  P1SEL0 |= BIT5;   // Set P1.5 to ADC input mode
  P1SEL1 |= BIT5;

  ADCCTL0 &= ~ADCENC;              // Disable ADC during configuration
  ADCCTL0 = ADCSHT_2 | ADCON;       // Sample hold time, ADC ON
  ADCCTL1 = ADCSHP | ADCCONSEQ_0;    // Sampling timer, single-channel
  ADCCTL2 = ADCRES;                 // 10-bit resolution
  ADCMCTL0 = ADCINCH_5;             // Select channel A5 (P1.5)
}

// Read from ADC channel A5 (P1.5)
uint16_t pot_Read(void)
{
  ADCCTL0 |= ADCENC | ADCSC;        // Enable and start conversion
  while (!(ADCIFG & ADCIFG0));      // Wait until conversion complete
  return ADCMEM0;                   // Return ADC value (0-1023)
}



#include <msp430.h>
#include <stdint.h>
#include "RGB_LED.h"

void pot_Init(void);
uint16_t pot_Read(void);

int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;     // Stop watchdog timer

  pot_Init();                 // Initialize potentiometer (P1.5 ADC)
  initRGBLED();               // Initialize RGB LED PWM

  PM5CTL0 &= ~LOCKLPM5;        // Unlock GPIOs
```

```
    uint16_t potValue;

    while (1)
    {
        potValue = pot_Read();    // Read potentiometer value (0-1023)

        // Map potentiometer reading to RGB values
        char red = (potValue >> 2) & 0xFF;    // Scale ADC 0-1023 to 0-255
        char green = (~red) & 0xFF;           // Inverse of red
        char blue = (red >> 1) & 0xFF;        // Half-intensity blue

        setRGBLED(red, green, blue);          // Update RGB LED color

        __delay_cycles(50000);                // Small delay for stability
    }
}

// Initialize P1.5 as ADC input
void pot_Init(void)
{
    P1SEL0 |= BIT5;   // Set P1.5 to ADC input mode
    P1SEL1 |= BIT5;

    ADCCTL0 &= ~ADCENC;                 // Disable ADC during configuration
    ADCCTL0 = ADCSHT_2 | ADCON;         // Sample hold time, ADC ON
    ADCCTL1 = ADCSHP | ADCCONSEQ_0;     // Sampling timer, single-channel
    ADCCTL2 = ADCRES;                   // 10-bit resolution
    ADCMCTL0 = ADCINCH_5;               // Select channel A5 (P1.5)
}

// Read from ADC channel A5 (P1.5)
uint16_t pot_Read(void)
{
    ADCCTL0 |= ADCENC | ADCSC;          // Enable and start conversion
    while (!(ADCIFG & ADCIFG0));        // Wait until conversion complete
    return ADCMEM0;                     // Return ADC value (0-1023)
}
```