HACETTEPE UNIVERSITY COMPUTER ENGINEERING DEPARTMENT

Student Information: Merve Müge Deliktaş - 21526896

Onur Cankur – 21526791

Course Information: BBM 434 - Embedded Systems Laboratory

Spring – 2019

Report Information: Lab-03 Experiment Report

Short Brief of Lab-03 and Function Explanations

In this lab, we implemented Approach Lighting System that contains LED lights for airports. It is important to make a safe take off. LEDs flashes from green, through yellow, to red at the beginning. However, when the wind direction changes, switch is pressed by the operator and LEDs are flashed from red, through yellow, to green. Figure 1 shows how we defined variables for Port E, SYSCTL_RCGC2_R and SysTick timer.

```
//*****************************************************************************
// define GPIO registers (PORTE)
// define SYSCTL_RCGC2_R
// define variables for SysTick
//*****************************************************************************
#define GPIO_PORTE_DATA_BITS_R  ((volatile unsigned long *)0x40024000)
#define GPIO_PORTE_DATA_R       (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_IS_R         (*((volatile unsigned long *)0x40024404))
#define GPIO_PORTE_IBE_R        (*((volatile unsigned long *)0x40024408))
#define GPIO_PORTE_IEV_R        (*((volatile unsigned long *)0x4002440C))
#define GPIO_PORTE_IM_R         (*((volatile unsigned long *)0x40024410))
#define GPIO_PORTE_RIS_R        (*((volatile unsigned long *)0x40024414))
#define GPIO_PORTE_MIS_R        (*((volatile unsigned long *)0x40024418))
#define GPIO_PORTE_ICR_R        (*((volatile unsigned long *)0x4002441C))
#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_DR2R_R       (*((volatile unsigned long *)0x40024500))
#define GPIO_PORTE_DR4R_R       (*((volatile unsigned long *)0x40024504))
#define GPIO_PORTE_DR8R_R       (*((volatile unsigned long *)0x40024508))
#define GPIO_PORTE_ODR_R        (*((volatile unsigned long *)0x4002450C))
#define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_PDR_R        (*((volatile unsigned long *)0x40024514))
#define GPIO_PORTE_SLR_R        (*((volatile unsigned long *)0x40024518))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_LOCK_R       (*((volatile unsigned long *)0x40024520))
#define GPIO_PORTE_CR_R         (*((volatile unsigned long *)0x40024524))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
#define GPIO_PORTE_ADCCTL_R     (*((volatile unsigned long *)0x40024530))
#define GPIO_PORTE_DMACTL_R     (*((volatile unsigned long *)0x40024534))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
```

Figure 1-defines

In Figure 2, you can see the implementation of main function. There is variable named “pressed” to check if the switch is pressed or not using in this function. After calling PortE_Init() and SysTick_Init() which are the initialization functions, in a while loop we constantly check if switch is pressed or not. If it is not pressed, the sequence of LEDs is red, green, yellow but if it is pressed, sequence of LEDs is yellow, green, red. Our delay is approximately 1 sec for each LED, and it will be showed in the next chapters.

```c
int main(void){
  PortE_Init();
  SysTick_Init();
  while(1){
    while(!pressed){
      GPIO_PORTE_DATA_R = 0x02; // Red LED is on
      RedGreenYellow_Wait(2);   // Wait and check if pressed
      GPIO_PORTE_DATA_R = 0x04; // Greed LED is on
      RedGreenYellow_Wait(2);   // Wait and check if pressed
      GPIO_PORTE_DATA_R = 0x08; // Yellow LED is on
      RedGreenYellow_Wait(2);   // Wait and check if pressed
    }
    while(pressed){
      GPIO_PORTE_DATA_R = 0x08; // Yellow LED is on
      YellowGreenRed_Wait(2);   // Wait and check if pressed
      GPIO_PORTE_DATA_R = 0x04; // Greed LED is on
      YellowGreenRed_Wait(2);   // Wait and check if pressed
      GPIO_PORTE_DATA_R = 0x02; // Red LED is on
      YellowGreenRed_Wait(2);   // Wait and check if pressed
    }
  }
}
```

Figure 2-main() function

As you can see from the above figure which is Figure 3, we initialized Port E. PE0 is input bit and PE1, PE2, PE3 are output bits in our implementation. Therefore, we assigned GPIO_PORTE_DIR_R = 0x0E. Details of the design will be explained in further chapters.

```c
void PortE_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;     // 1) activate clock for Port E
  delay = SYSCTL_RCGC2_R;           // allow time for clock to start
  GPIO_PORTE_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port E
  GPIO_PORTE_CR_R = 0x0E;           // allow changes to PE1-3
  GPIO_PORTE_AMSEL_R = 0x00;        // 3) disable analog on PE
  GPIO_PORTE_PCTL_R = 0x00000000;   // 4) PCTL GPIO on PE4-0
  GPIO_PORTE_DIR_R = 0x0E;          // 5) PE0 in, PF3-1 out
  GPIO_PORTE_AFSEL_R = 0x00;        // 6) disable alt funct on PE7-0
  GPIO_PORTE_PUR_R = 0x00;
  GPIO_PORTE_DEN_R = 0x1F;          // 7) enable digital I/O on PE4-0
}

void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;
  NVIC_ST_CTRL_R = 0x00000005;
}
```

Figure 3-init functions

In Figure 4, you can see the implementation of the sequence red, green, yellow. It is the SysTick implementation that we learnt in class. In a while loop, it continuously checks if pressed or not.

```c
void RedGreenYellow_Wait(unsigned int delay){
  int i;
  for(i=0; i<delay*10; i++){
    NVIC_ST_RELOAD_R = 800000-1;  // number of counts to wait
    NVIC_ST_CURRENT_R = 0;
    while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait untill count bit is 1
      SW = GPIO_PORTE_DATA_R & 0x01;
      if(SW == 0x00){   // check if pressed or not
        pressed = 1;
      }
    }
  }
}
```

Figure 4-RedGreenYellow_Wait() function

```c
void YellowGreenRed_Wait(unsigned int delay){
  int i;
  for(i=0; i<delay*10; i++){
    NVIC_ST_RELOAD_R = 800000-1;  // number of counts to wait
    NVIC_ST_CURRENT_R = 0;
    while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait untill count bit is 1
      SW = GPIO_PORTE_DATA_R & 0x01;
      if(SW == 0x00){   // check if pressed or not
        pressed = 0;
      }
    }
  }
}
```

Figure 5-YellowGreenRed_Wait() function

From figure 5, you can see that we implemented the SysTick function for the reverse sequence which is yellow, green, red. Only difference between this function and RedGreenYellow_Wait() function is if the switch is pressed, this function assigns 0 to pressed. It is like making a variable true and false.

PART A

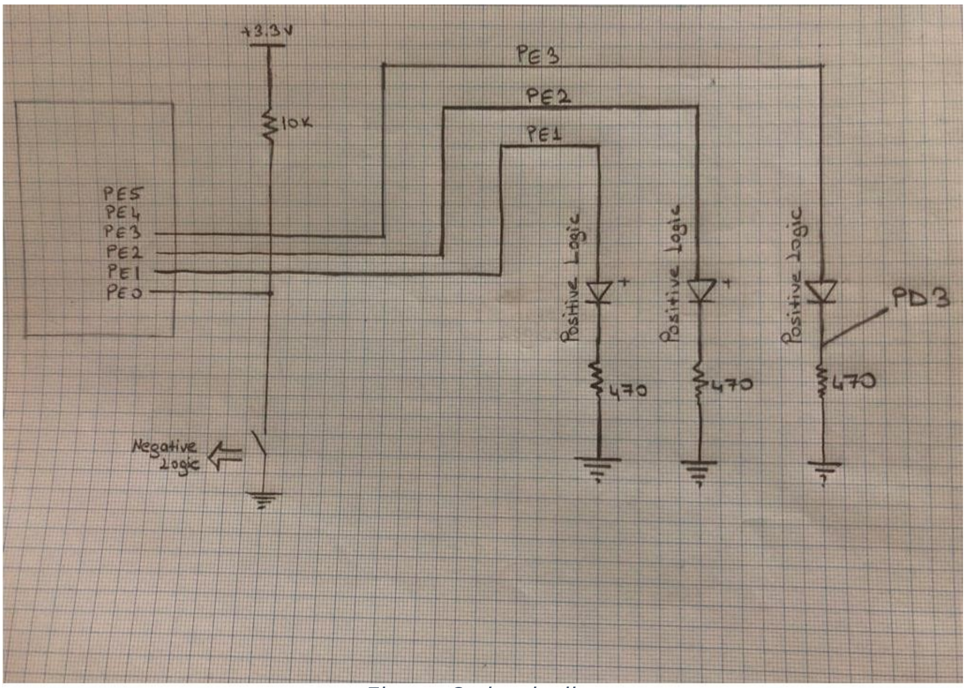

Figure 6-circuit diagram

In this part, we draw the circuit. We used one switch and three LEDs. Negative logic for the switch and positive logic for the LEDs. PE0 input is used for the switch and PE1, PE2, and PE3 outputs are used for LEDs.

PART B

In this part, we wrote the software for our circuit. As we explained briefly before, Port E is chosen and PE0 is used as input port and PE1, PE2, PE3 are used as output ports. In addition to them, we used SysTick timer. You can see its initialization in Figure 3, and implementation in Figure 4 and Figure 5.

Figure 7-Logic Analyzer

You can see from Figure 7, finishing a sequence lasts approximately 3 seconds. It means that each of our LEDs delay for 1 second. Because of using negative logic on switch, the signal that you see at the bottom is 0 when the switch is pressed and 1 when the switch is not pressed. In this example, it is pressed.

PART C

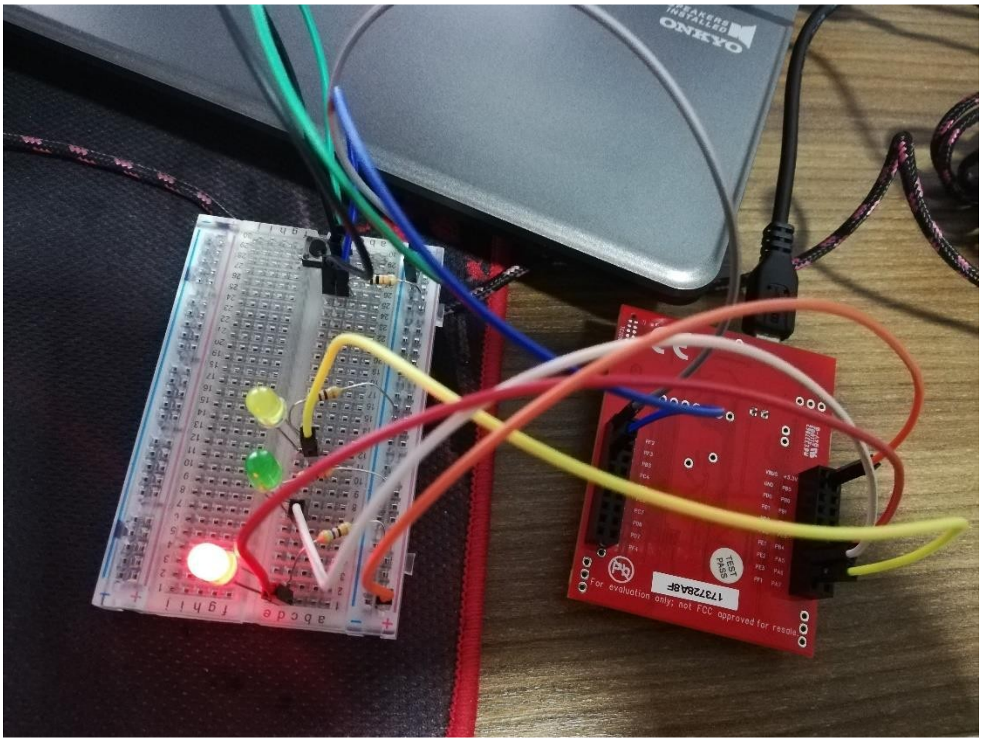

Figure 8-red LED on real board

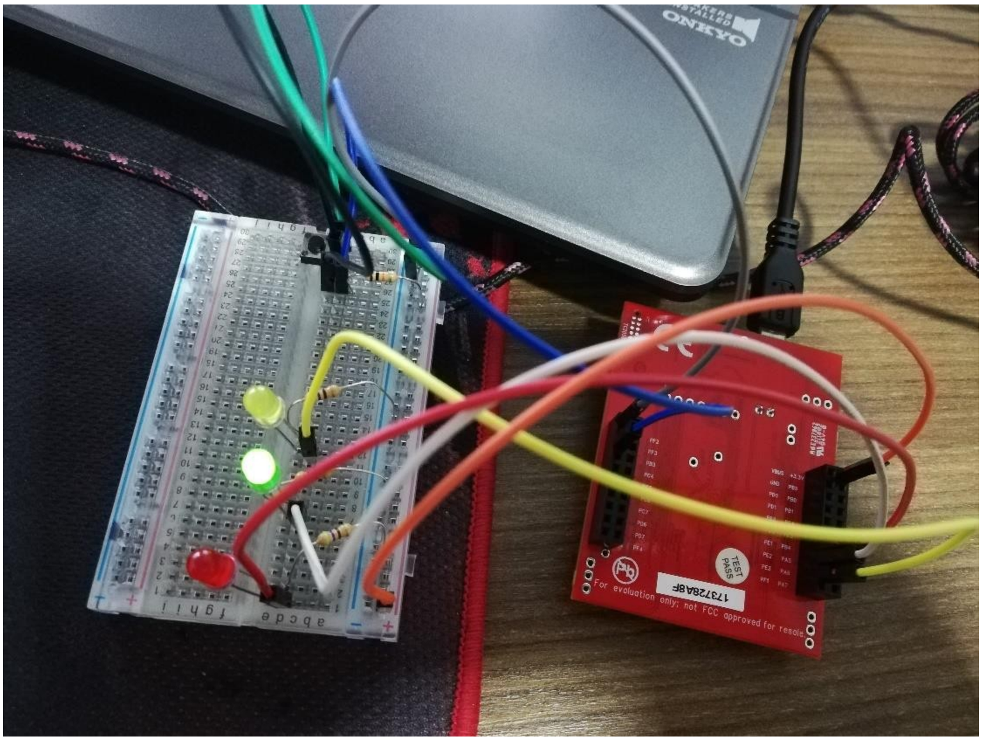

Figure 9-green LED on real board

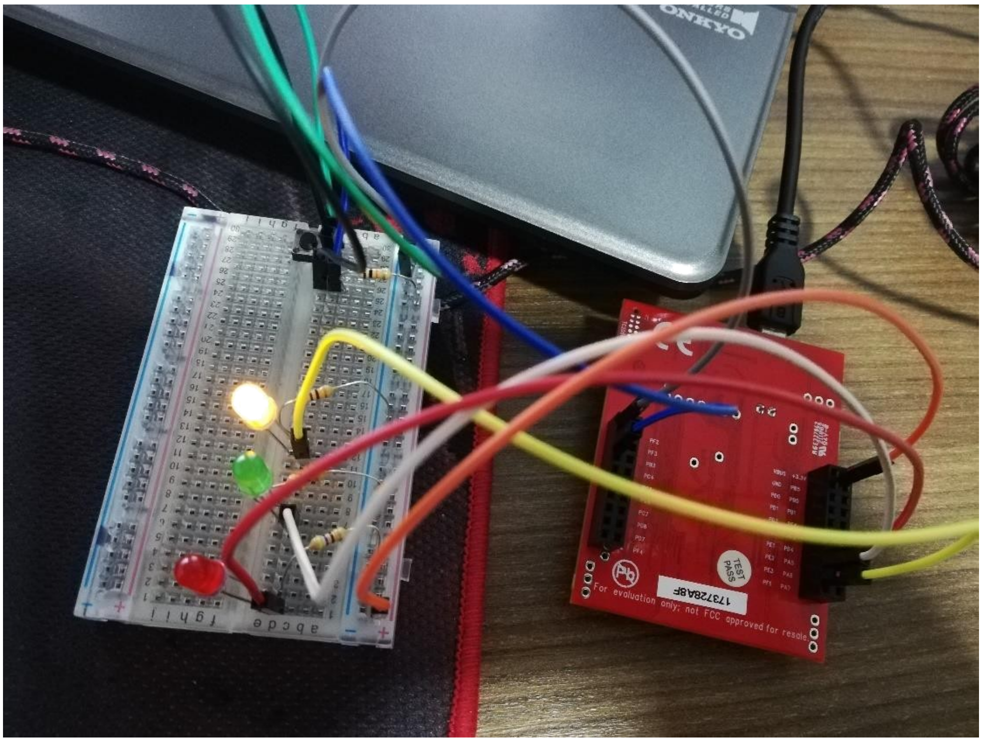

Figure 10-yellow LED on real board

After debugged our software on debugger, we built the hardware on the real board. You can see our hardware in Figure 8, Figure 9, and Figure 10. We built it according to the circuit that we showed in Part A. We used 10kΩ resistor for the switch and 470Ω resistors for each LED. PE0 pin is used for the switch and if it is pressed signal on the port is 0, otherwise 1. PE3 pin used for yellow LED, PE2 is for green LED and PE1 is for red.

PART D

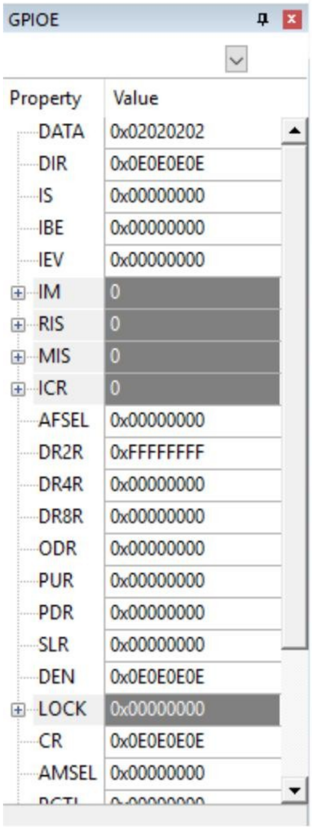

Figure 11-red on debugger

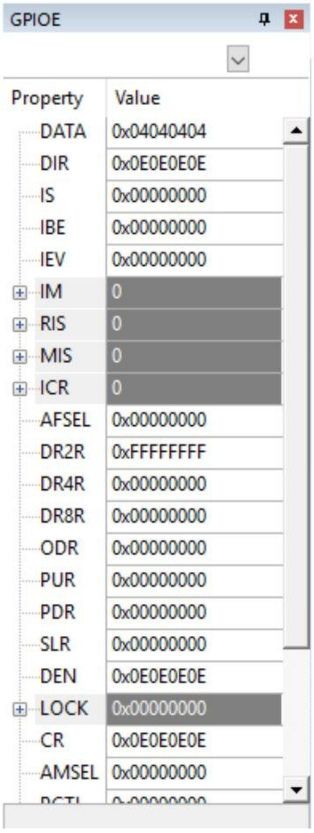

Figure 12-green on debugger

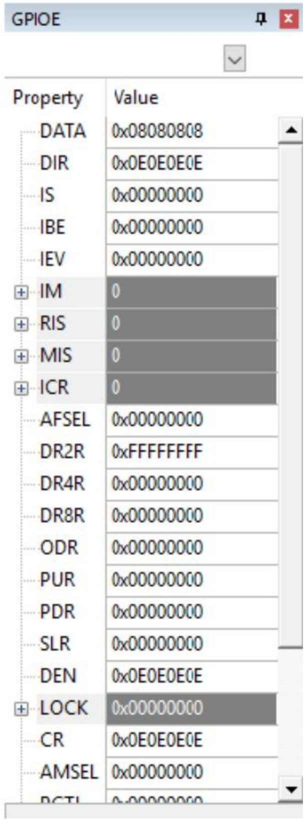

Figure 13-yellow on debugger

In this part, we debugged our combined hardware/software system on actual board. From Peripherals->SystemViewer->GPIO->GPIOE, we opened the debugger. As you can see from the Figure 11, Figure 12 and Figure 13, values of DIR, DEN and CR is 0x0E because we assigned them like that in our PortE_Inıt() function which can be seen in Figure 3. In addition, you can see that in Figure 11, DATA is 0x02 which represents red; in Figure 12, DATA is 0x04 which represents green and in Figure 13, DATA is 0x08 which represents yellow.

CALCULATIONS

As it is written in experiment pdf, we assume that our chip produces 3.0V and our LEDs operate at 1.8V and 2mA. If we directly send 3.0V to our LEDs, they will not work. Therefore, we should add resistors to our circuit to reduce volt on our LEDs. In order to get 1.8V on our LEDs, result is approximate, but we should do this calculation:

$$R = \frac{V_{OH} - V_{LED} - V_{Ground}}{I_{LED}} = \frac{3.0 - 1.8 - 0.4}{0.002} = 400\Omega$$