HACETTEPE UNIVERSITY COMPUTER ENGINEERING DEPARTMENT

Group Name:	Grup
Student Information:	Merve Müge Deliktaş - 21526896
	Onur Cankur – 21526791
Course Information:	BBM 434 - Embedded Systems Laboratory
	Spring – 2019
Report Information:	Lab-05 Experiment Report

Short Brief of Lab-05 and Function Explanations

In this lab, we designed a Fastest Finger reaction game. We used one green LED, one red LED, two 10K resistors, two 470Ω registers and a Nokia 5110 display.

We used positive logic for our LEDs and negative logic for our switches as the previous lab. The circuit is the same for LEDs and switches. In addition to them, we used Nokia 5110 display this time and we connected it as explained in the lecture.

```
void Switch_LED_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;         // activate clock for Port E
  delay = SYSCTL_RCGC2_R;               // allow time for clock to start
  GPIO_PORTE_LOCK_R = 0x4C4F434B;       // unlock GPIO Port E
  GPIO_PORTE_CR_R = 0x0F;               // allow changes to PE0-3
  GPIO_PORTE_AMSEL_R = 0x00;            // disable analog on PE
  GPIO_PORTE_PCTL_R &= ~0x0000FFFF;     // configure PE3-0 as GPIO
  GPIO_PORTE_DIR_R = 0x0C;              // PE0-1 in, PE2-3 out
  GPIO_PORTE_AFSEL_R = 0x00;            // disable alt funct on PE7-0
  GPIO_PORTE_DEN_R = 0x0F;              // enable digital I/O on PE3-0
  GPIO_PORTE_IS_R &= ~0x03;             // PE0 is edge-sensitive
  GPIO_PORTE_IBE_R &= ~0x03;            // PE0 is not both edges
  GPIO_PORTE_IEV_R &= ~0x03;            // PE0 falling edge event
  GPIO_PORTE_ICR_R = 0x03;              // clear flag0
  GPIO_PORTE_IM_R |= 0x03;              // arm interrupt on PE0
  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF00)|0x00000020; // priority 1
  NVIC_EN0_R = 0x00000010;              // enable interrupt 4 in NVIC
}
```

Figure 1 - Initialize switches and LEDs

From Figure 1, you can see initialization of our switches and LEDs. We used PE0 and PE1 for LEDs; PE2 and PE3 for switches. Like in the previous lab, we used falling edge sensitive interrupts for switches.

```
void SysTick_Init(unsigned long period){
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = period-1;// reload value
  NVIC_ST_CURRENT_R = 0;      // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; // priority 2
  NVIC_ST_CTRL_R = 0x07;      // enable SysTick with core clock and interrupts
}
```

Figure 2 - Initialize SysTick

From Figure 2, you can see initialization of SysTick, it also the same function in previous lab.

In addition to these initializations, we also used a function which initializes the Nokia 5110 display. We used library for that initialization which is taken from the lab5.pdf file that is shared with us.

```
/*
* SysTick handler will be operated after every 1 ms which we set when initialization of SysTick.
*/
void SysTick_Handler(void){
  timer++;
  if(timer == start_time){
    Nokia5110_Clear();
    Nokia5110_SetCursor(0,0);
    Nokia5110_DisplayBuffer();
    Nokia5110_OutString("PRESS");
    pressed++;
  }
}
```

Figure 3 - SysTick_Handler()

SysTick_Handler function can be seen from Figure 3. It increments the timer value every millisecond. Also, it prints “PRESS” command to the screen.

```
void GPIOPortE_Handler(void){
  Nokia5110_Clear();
  Nokia5110_DisplayBuffer();
  // If pressed > 0, that means player pressed the button after "Press" command is displayed on the screen.
  if(pressed > 0){
    if(GPIO_PORTE_RIS_R & 0x01){
      GPIO_PORTE_ICR_R = 0x01;
      Player1 = timer - start_time;
    }
    if(GPIO_PORTE_RIS_R & 0x02){
      GPIO_PORTE_ICR_R = 0x02;
      Player2 = timer - start_time;
    }
    if(Player1 != 0 && Player2 != 0){
      SetWinner();
      turn++;
    }
  }
  // If pressed <= 0, that means player pressed the button before "Press" command is displayed on the screen.
  else{
    if(GPIO_PORTE_RIS_R & 0x01){
      GPIO_PORTE_ICR_R = 0x01;
      Nokia5110_Clear();
      Nokia5110_SetCursor(0,0);
      Nokia5110_OutString("RED: false ");
      Nokia5110_SetCursor(0,1);
      Nokia5110_OutString("start");
      Nokia5110_SetCursor(0,2);
      Nokia5110_OutString("GREEN WINS!");
      GPIO_PORTE_DATA_R |= 0x08;
      GPIO_PORTE_DATA_R &= ~0x04;
    }
    if(GPIO_PORTE_RIS_R & 0x02){
      GPIO_PORTE_ICR_R = 0x02;
      Nokia5110_Clear();
      Nokia5110_SetCursor(0,0);
      Nokia5110_OutString("GREEN: false");
      Nokia5110_SetCursor(0,1);
      Nokia5110_OutString("start");
      Nokia5110_SetCursor(0,2);
      Nokia5110_OutString("RED WINS!");
      GPIO_PORTE_DATA_R |= 0x04;
      GPIO_PORTE_DATA_R &= ~0x08;
    }
    turn++;
  }
}
```

Figure 4 - GPIOPortE_Handler()

Handler for switch interrupts can be seen from Figure 4. First, it checks if the player pressed the button after “PRESS” command or not.

If player pressed after the command, then it calculates time, which will be explained in details later, and calls the SetWinner() function to decide who win the game.

If player pressed before the command, then it prints “false start” and other player wins.

```
void SetWinner(void){
  Nokia5110_Clear();
  Nokia5110_SetCursor(0,0);
  Nokia5110_DisplayBuffer();
  Nokia5110_OutString("RED:");
  Nokia5110_OutUDec(Player1);
  Nokia5110_OutString(" ms");
  Nokia5110_SetCursor(0,1);
  Nokia5110_OutString("GREEN:");
  Nokia5110_OutUDec(Player2);
  Nokia5110_OutString("ms");
  Nokia5110_SetCursor(0,2);
  if(Player1 <= 100){
      Nokia5110_Clear();
      Nokia5110_SetCursor(0,0);
      Nokia5110_OutString("RED: false ");
      Nokia5110_SetCursor(0,1);
      Nokia5110_OutString("start");
      Nokia5110_SetCursor(0,2);
      Nokia5110_OutString("GREEN WINS!");
      GPIO_PORTE_DATA_R |= 0x08;
      GPIO_PORTE_DATA_R &= ~0x04;
  }
  else if(Player2 <= 100){
      Nokia5110_Clear();
      Nokia5110_SetCursor(0,0);
      Nokia5110_OutString("GREEN: false");
      Nokia5110_SetCursor(0,1);
      Nokia5110_OutString("start");
      Nokia5110_SetCursor(0,2);
      Nokia5110_OutString("RED WINS!");
      GPIO_PORTE_DATA_R |= 0x04;
      GPIO_PORTE_DATA_R &= ~0x08;
  }
  else {
    if(Player1 > Player2){
        Nokia5110_OutString("GREEN WINS");
        GPIO_PORTE_DATA_R |= 0x08;
        GPIO_PORTE_DATA_R &= ~0x04;
    }
    else{
      Nokia5110_OutString("RED WINS!");
      GPIO_PORTE_DATA_R |= 0x04;
      GPIO_PORTE_DATA_R &= ~0x08;
    }
  }
}
```

Figure 5 - SetWinner()

In SetWinner() function which can be seen from Figure 5, first, times of the presses are printed to the screen for both users. Then, if players pressing time is less than 100ms, it must be false start because pressing the button like that is impossible for humans. If both players pressing times are higher than 100ms, then checks which player pressed first. If Player 1 pressed first, then green LED is open. Otherwise, red LED is open.

```
void Delay(unsigned int deliminator){
  unsigned long volatile time;
  time = deliminator*160000;
  DisableInterrupts();
  while(time){
    time--;
  }
}
```

Figure 6 - Delay()

From Figure 6, Delay() function can be seen. Interrupts are disabled in this function and it will be explained in the explanation of main function.

```
int main(void){
  DisableInterrupts();
  Switch_Init();
  SysTick_Init(16000);
  Nokia5110_Init();
  EnableInterrupts();
  while(1){
    Nokia5110_Clear();
    start_time = ((rand() % 10) + 1) * 1000; // random int between 0 and 9 plus 1
    turn = 0;
    timer = 0;
    Player1 = 0;
    Player2 = 0;
    pressed = 0;

    Nokia5110_ClearBuffer();
    Nokia5110_SetCursor(0,0);
    Nokia5110_DisplayBuffer();
    GPIO_PORTE_DATA_R = 0x00;
    Nokia5110_OutString("GET READY!");
    while(turn == 0){
      WaitForInterrupt();
    }
    Delay(50);
    EnableInterrupts();
    GPIO_PORTE_DATA_R = 0x00;   // turns off leds before the new turn
  }
}
```

Figure 7 - main()

Figure 7 shows the main function. Initializations are made in this function. In an infinite while loop, we clear the variables for beginning of turn. “GET READY!” is printed on the screen and interrupts are waited to occur. At the end of the each game, program waits for 5 seconds using Delay() function. In this Delay() function, we disabled interrupts because we only want to see result. The game completely stops and waits for 5 seconds.

Board Pictures

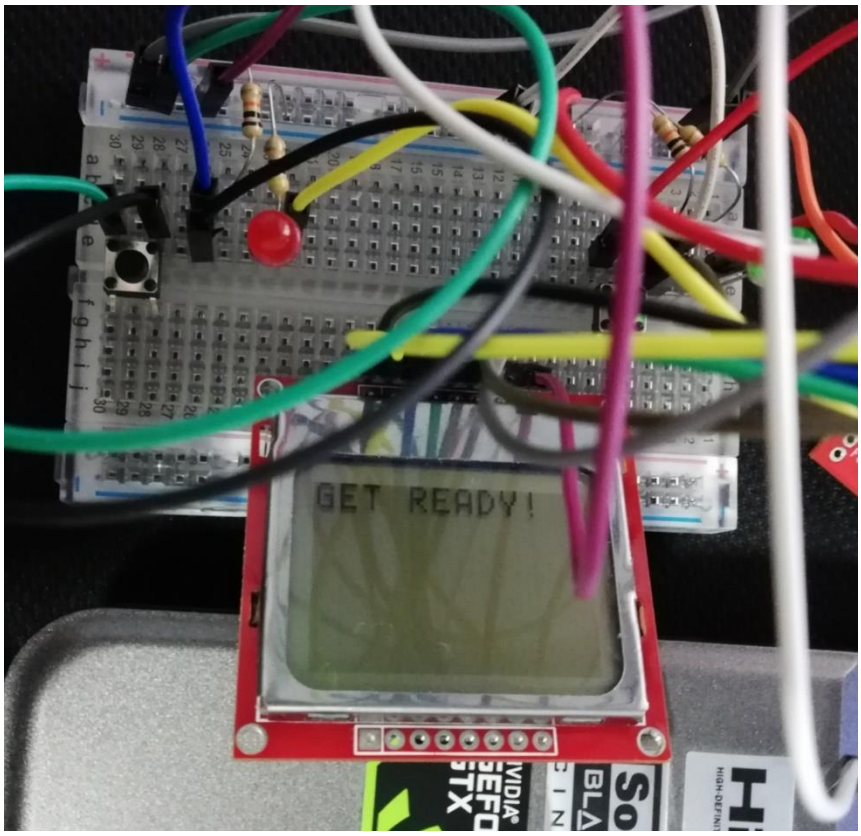

Figure 8 - GET READY! command on board

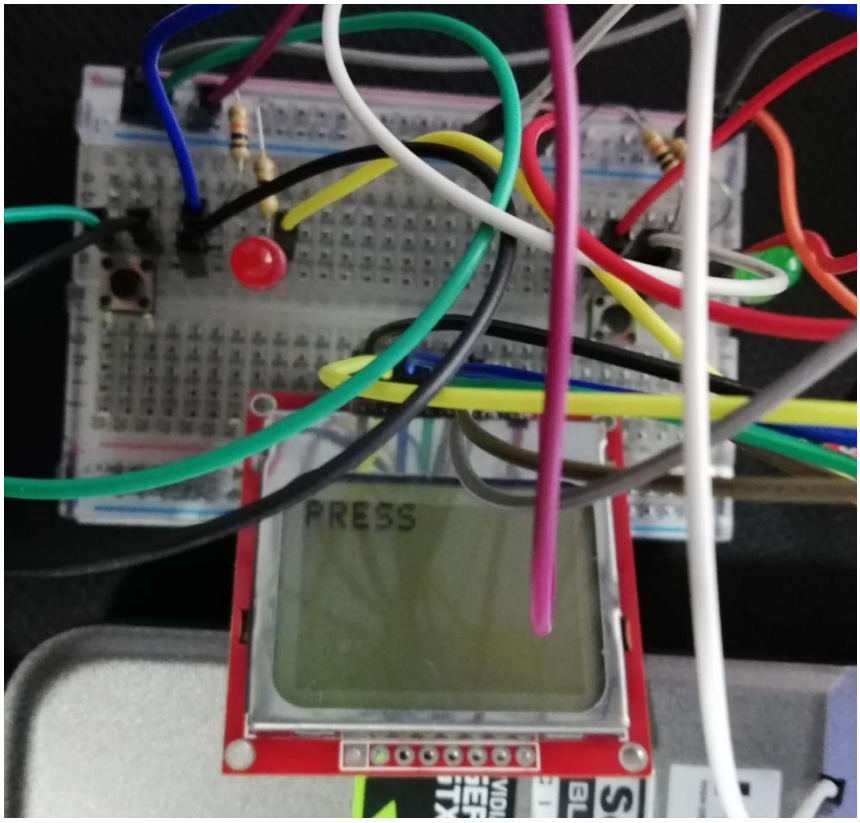

Figure 9 - PRESS command on board

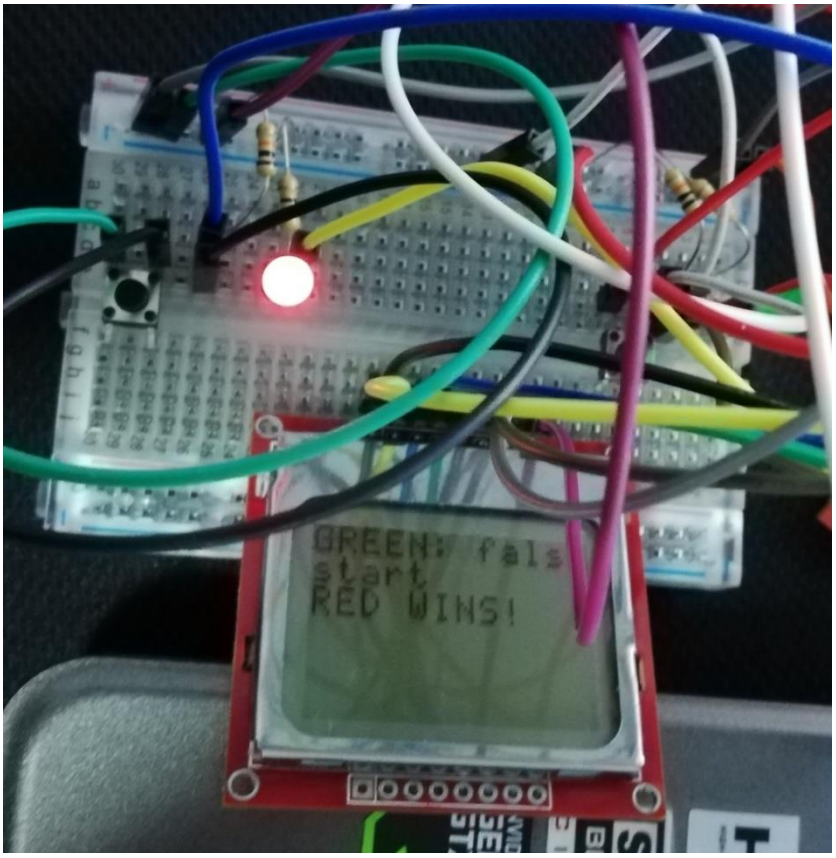

Figure 10 - Green false start, red wins

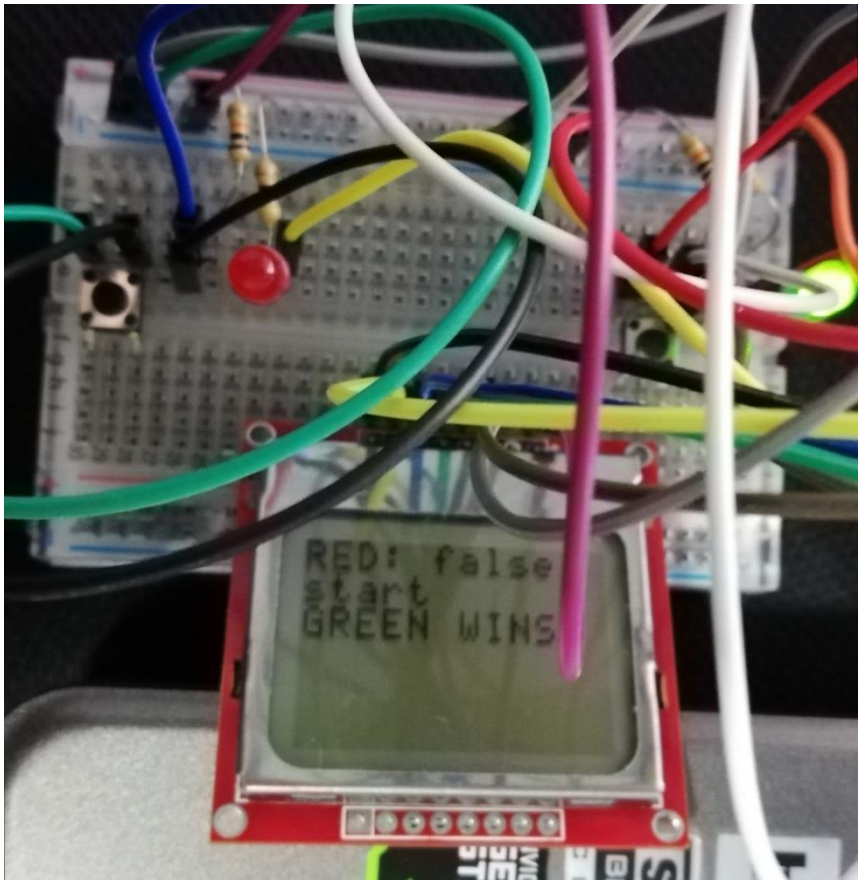

Figure 11 - red false start, green wins

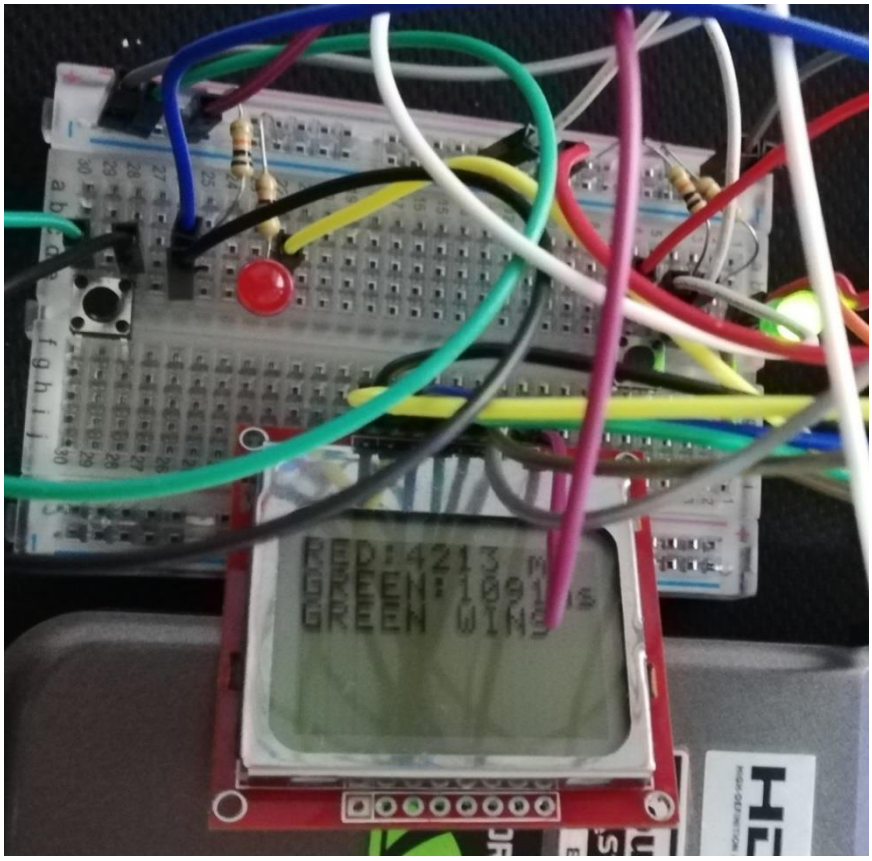

Figure 12 - green presses early and wins

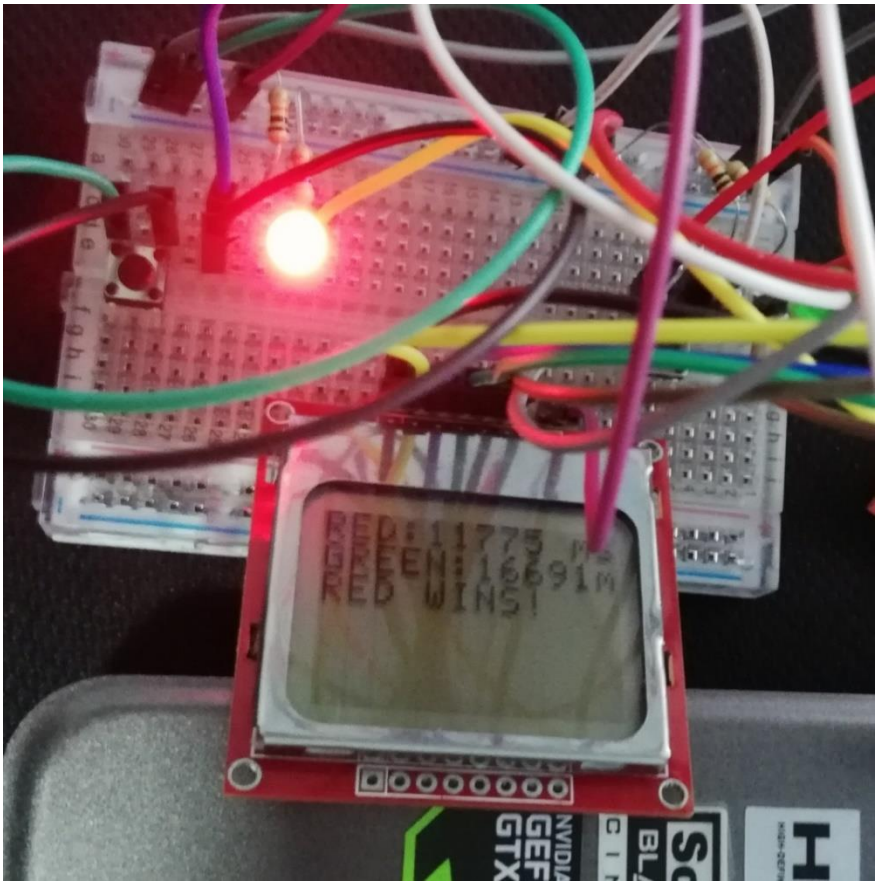

Figure 13 - red presses early and wins

Time Measurement

To be able to measure time, for every millisecond, we increment a variable named “timer” by one. In addition to it, we use a variable named “start_time” and it stores a random number between 1-10 and multiplies it by 1000. Game starts after waiting this start_time.

For the players, when switch interrupt is occurred, calculation for pressing time is done by subtracting start_time from the timer.

Because of incrementing timer for every millisecond, results of our calculations are also in milliseconds.