HACETTEPE

UNIVERSITY

Computer Science

Departement

EMBEDDED SYSTEMS LAB.

LAB. REPORT V

Team Name:

KENDINI MARUL SANAN MARUL

Members:

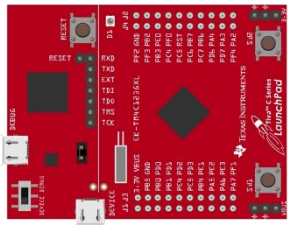
KUTAY BARCIN 21526715

DEFNE TUNCER 21627686

MAY THE FASTEST WIN

In this game, two players are competing to be the first to press their switch with a command from the screen.

STEP1: EQUIPMENT



TM4C123G:

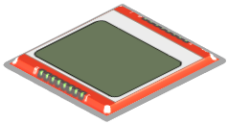
LEDs: Two 1.8V 2mA LEDs

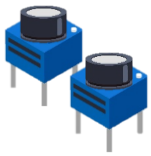
$R = (V_{OH} - V_d)/I_d$

$470\Omega = (3.0 - 1.8)/I_d$

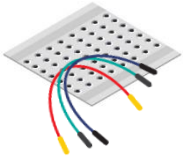
$I_d = 0.0026$ A

NOKIA 5110 LCD:

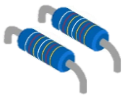




SWITCHES: Two B3F Tactile Switches



BREADBOARD AND CABLES:



RESISTORS:

Two 470Ω resistors

Two 10kΩ resistors

STEP2: I/O

PORTE initialization is done: For the switch interrupts, **falling edge** event is used and **priority** is set to 2.

INPUTS: PE2 **(green)**, PE3 **(red)**

OUTPUTS: PE4 **(green)**, PE5 **(red)**

```
#define PE2 (*((volatile unsigned long *)0x40024010))
#define PE3 (*((volatile unsigned long *)0x40024020))
#define PE4 (*((volatile unsigned long *)0x40024040))
#define PE5 (*((volatile unsigned long *)0x40024080))

void PortE_Init_2345(void){volatile unsigned long delay;
  SYSCTL_RCGCGPIO_R |= 0x00000010;      // activate clock for Port E
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTE_AMSEL_R &= ~0x3C;          // disable analog on PE2-5
  GPIO_PORTE_PCTL_R &= ~0x00FFFF00;     // PCTL GPIO on PE2-5
  GPIO_PORTE_DIR_R = 0x30;              // PE2 PE3 input PE4 PE5 output
  GPIO_PORTE_AFSEL_R &= ~0x3C;          // PE2-5 regular port function
  GPIO_PORTE_DEN_R |= 0x3C;             // enable PE2-5 digital port
  GPIO_PORTE_IS_R &= ~0x0C;             // PE2 PE3 is edge-sensitive
  GPIO_PORTE_IBE_R &= ~0x0C;            // PE2 PE3 is not both edges
  GPIO_PORTE_IEV_R &= ~0x0C;            // PE2 PE3 falling edge event
  GPIO_PORTE_ICR_R = 0x0C;              // clear flag2 and flag3
  GPIO_PORTE_IM_R |= 0x0C;              // arm interrupt on PE2 PE3
  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF00)|0x00000040; // priority 2
  NVIC_EN0_R = 0x00000010;              // enable interrupt 4 in NVIC
}
```

PORTA initialization is done: **Nokia5110_Init** function from **Nokia5110.c** source file is used.

INPUTS: PA2, PA3, PA5 **(SSI0)**

OUTPUTS: PA6, PA7 **(GPIO)**

```
// Blue Nokia 5110
// ---------------
// Signal         (Nokia 5110) LaunchPad pin
// Reset          (RST, pin 1) connected to PA7
// SSI0Fss        (CE,  pin 2) connected to PA3
// Data/Command   (DC,  pin 3) connected to PA6
// SSI0Tx         (Din, pin 4) connected to PA5
// SSI0Clk        (Clk, pin 5) connected to PA2
// 3.3V           (Vcc, pin 6) power
// back light     (BL,  pin 7) ground
// Ground         (Gnd, pin 8) ground
```

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PA0		*Port*	*U0Rx*							CAN1Rx		
PA1		*Port*	*U0Tx*							CAN1Tx		
PA2		Port		SSI0Clk								
PA3		Port		SSI0Fss								
PA4		Port		SSI0Rx								
PA5		Port		SSI0Tx								

STEP4: SYSTICK TIMER

SYSTICK TIMER initialization is done: `SysTick_Init(1600000); // set to 100ms`

```
unsigned short timer;

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)|0xA0000000; // priority 5
  NVIC_ST_CTRL_R = 0x07;       // enable SysTick with core clock and interrupts
}

void SysTick_Handler(void){timer+=1;}

unsigned short SysTick_TimePassed(void){return (((NVIC_ST_RELOAD_R-NVIC_ST_CURRENT_R)*62.5)/1000000)+(timer*100);}
```

SysTick_Handler: unsigned short **timer** variable is increased with every SysTick interrupt.

SysTick_TimePassed: Passed time is measured.

STEP5: MAIN

```
int main(void){
  PortE_Init_2345();
  Nokia5110_Init();
  EnableInterrupts();
  while(1){newGame();}
}
```

STEP6: SWITCHES

After deciding which switch is pressed, corresponding flag is cleared. During a game, both switches can be pressed only once, so switch is disabled after it is pressed. That way switch bounces are prevented too.

- If the game is over, **replay** is set, so that after pressing one of switches a new game starts.
- Falsestart happens when switch is pressed before GO! appears or after 100ms passes.

```
void GPIOPortE_Handler(void){
  if(GPIO_PORTE_RIS_R&0x04){                    // GREEN LED:PE4 SWITCH:PE2
    GPIO_PORTE_ICR_R = 0x04;                    // clear flag4
    if(green_enable){
      green_enable=0;
      if(replay) replay=0;                      // replay
      else if (red==-1) green=1;
      else if(!timer | falsestart) green=-1;    // falsestart
      else green=SysTick_TimePassed();
    }
  }
  else if(GPIO_PORTE_RIS_R&0x08){               // RED LED:PE5 SWITCH:PE3
    GPIO_PORTE_ICR_R = 0x08;                    // clear flag5
    if(red_enable){
      red_enable=0;
      if(replay) replay=0;                      // replay
      else if (green==-1) red=1;
      else if(!timer || falsestart) red=-1;     // falsestart
      else red=SysTick_TimePassed();
    }
  }
}
```

STEP7: GAME

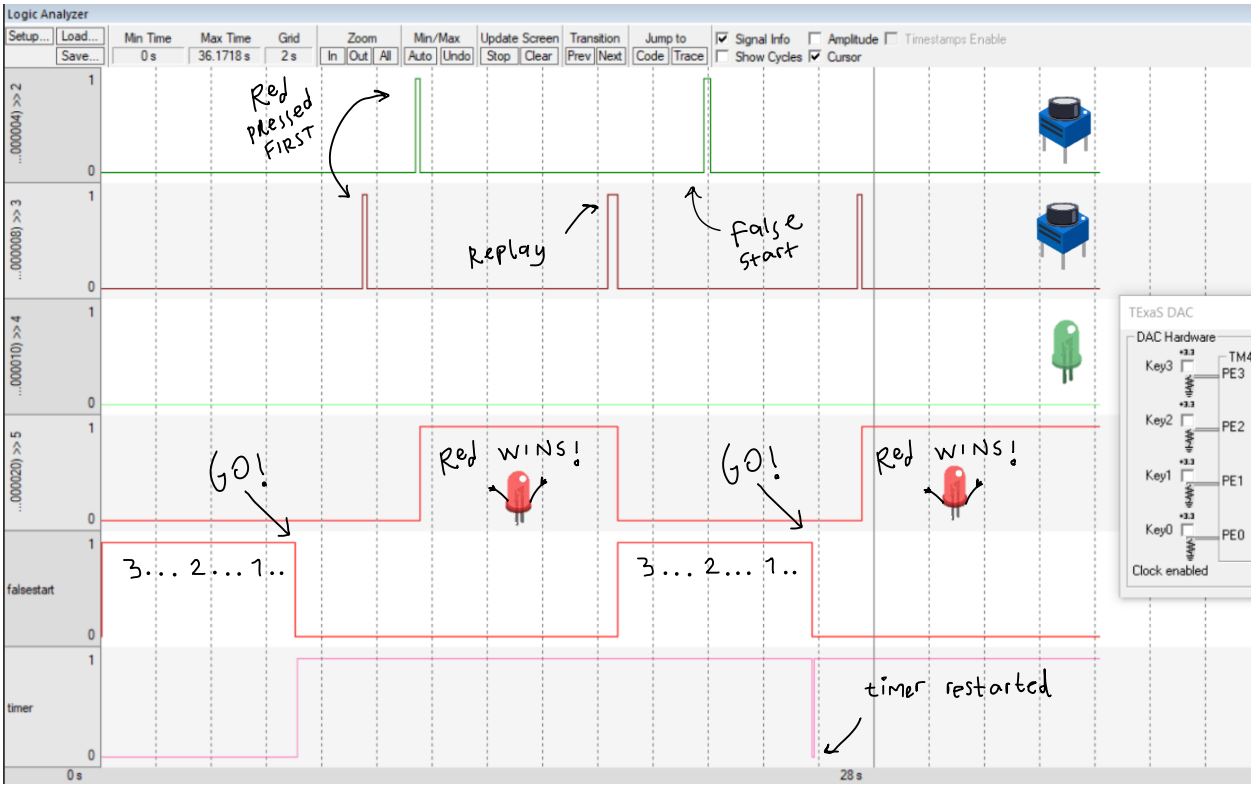
During tests, bounces are observed for both when switch is pressed and when switch is released:

- Red pressed → green pressed and bounced → a new game starts before announcing results
- Green pressed for replay and bounced → falsestart for green
- Switch is released too late and bounce happens

Prior to announcing results, it is made sure that both are pressed and released, and all bounces are over.

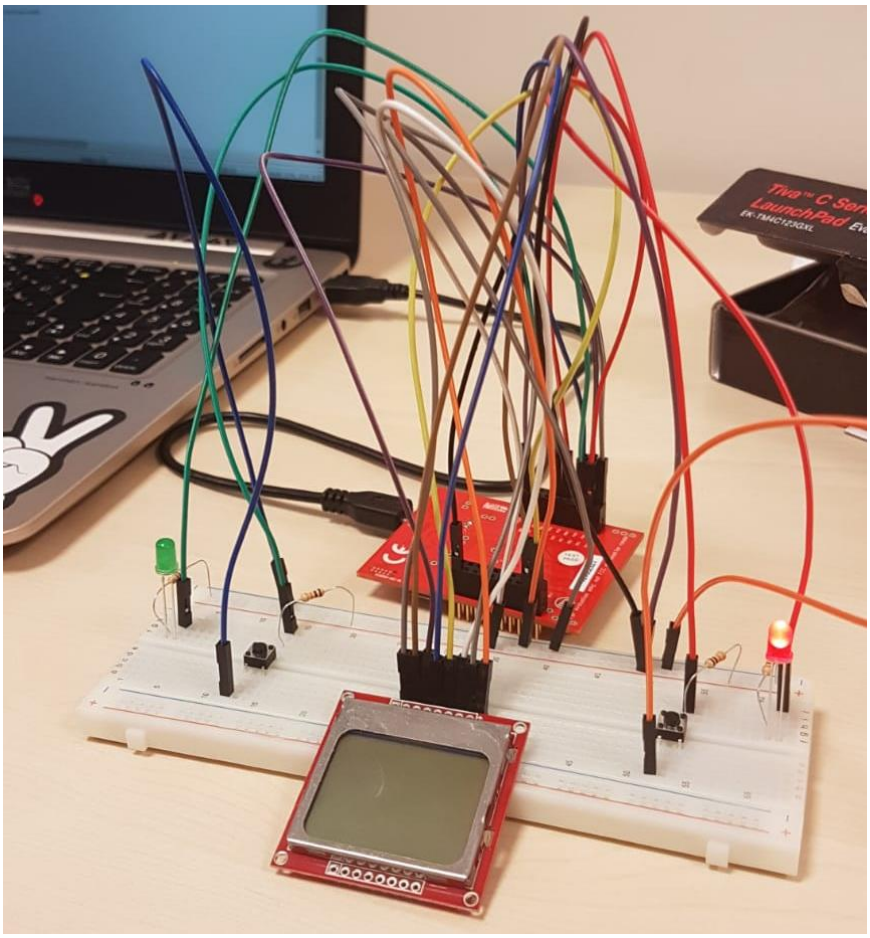
```
void newGame(void){
  falsestart=1; green_enable=1; red_enable=1;
  PE4=0x00; PE5=0x00;
  red=0; green=0; replay=0;
  Nokia5110_ClearBuffer(); Nokia5110_DisplayBuffer(); Nokia5110_Clear();
  Nokia5110_SetCursor(5, 1); Nokia5110_OutString("MAY");
  Nokia5110_SetCursor(0, 2); Nokia5110_OutString("THE FASTEST");
  Nokia5110_SetCursor(4, 3); Nokia5110_OutString("WINS!");
  Delay(4); Nokia5110_Clear(); Nokia5110_SetCursor(6, 2); Nokia5110_OutString("3");
  Delay(1); Nokia5110_Clear(); Nokia5110_SetCursor(6, 2); Nokia5110_OutString("2");
  Delay(1); Nokia5110_Clear(); Nokia5110_SetCursor(6, 2); Nokia5110_OutString("1");
  Delay(1); Nokia5110_Clear(); Nokia5110_SetCursor(5, 2); Nokia5110_OutString("GO!");
  falsestart=0; timer=0;
  SysTick_Init(1600000);           // initialize SysTick timer, every 100ms
  while(!(red && green)){WaitForInterrupt();} Delay(0.1);
  while(!(PE2&0x04 && PE3&0x08)){WaitForInterrupt();} Delay(0.1); // wait both to release (incase of release bounce)
  Nokia5110_Clear();
  if(red==-1){
    Nokia5110_OutString("RED:");
    Nokia5110_SetCursor(0,1); Nokia5110_OutString("false start");
    Nokia5110_SetCursor(0,2); Nokia5110_OutString("GREEN WINS!"); PE4=0x10;
  }
  else if(green==-1){
    Nokia5110_OutString("GREEN:");
    Nokia5110_SetCursor(0,1); Nokia5110_OutString("false start");
    Nokia5110_SetCursor(0,2); Nokia5110_OutString("RED WINS!"); PE5=0x20;
  }
  else {
    Nokia5110_OutString("RED:");
    Nokia5110_SetCursor(0,1); Nokia5110_OutUDec(red);
    Nokia5110_SetCursor(9,1); Nokia5110_OutString(" ms");
    Nokia5110_SetCursor(0,2); Nokia5110_OutString("GREEN:");
    Nokia5110_SetCursor(0,3); Nokia5110_OutUDec(green);
    Nokia5110_SetCursor(9,3); Nokia5110_OutString(" ms");
    if(red<green){Nokia5110_SetCursor(0,4); Nokia5110_OutString("RED WINS!"); PE5=0x20;}
    else {Nokia5110_SetCursor(0,4); Nokia5110_OutString("GREEN WINS!"); PE4=0x10;}
  }
  Nokia5110_SetCursor(0,5); Nokia5110_OutString("PRESStoPLAY");
  replay=1; red_enable=1; green_enable=1;
  while(replay) {WaitForInterrupt();} Delay(0.1);
  while(!(PE2&0x04 && PE3&0x08)){WaitForInterrupt();} Delay(0.1); // wait both to release (incase of release bounce)
}
```

STEP8: SIMULATION

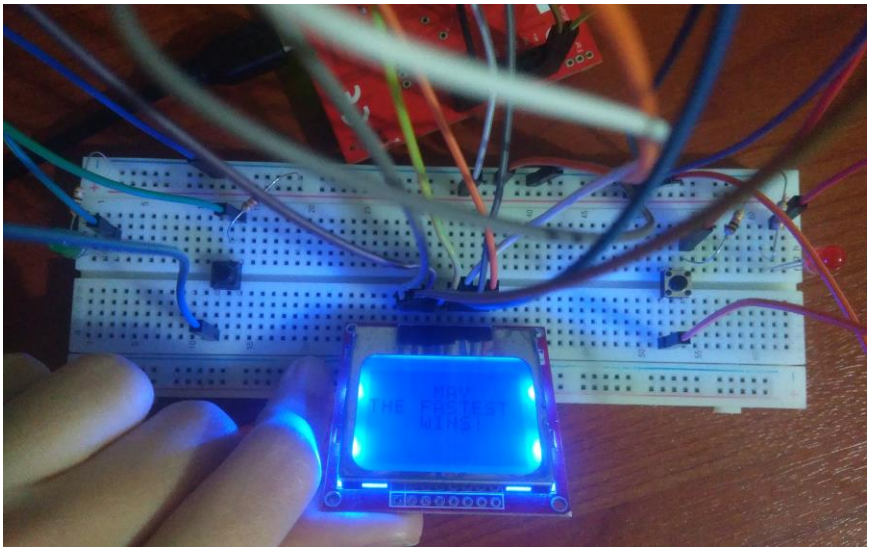


STEP8: CIRCUIT DESIGN

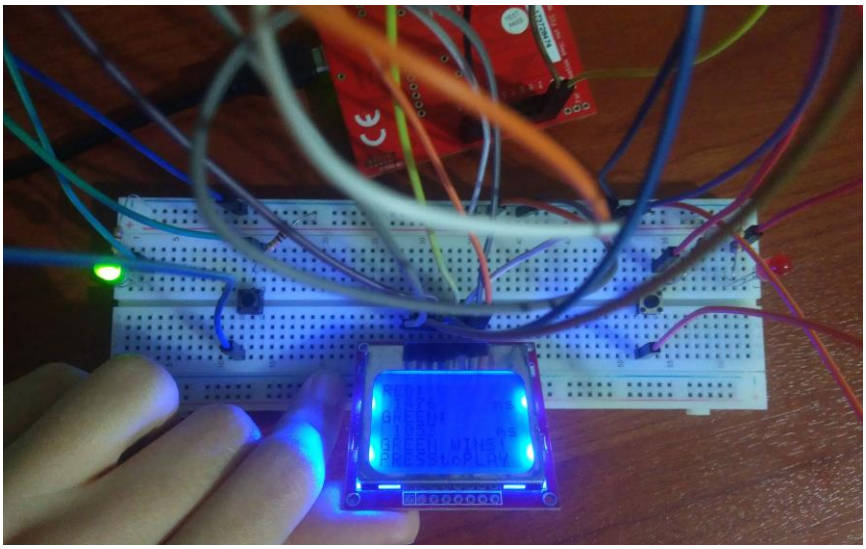
1. Circuit



2. Game Start



3. Win



4. False start