گزارش دستورکار سری هفتم ریزپردازنده (بخش عملی)

امیرحسین ادواری – ۹۸۲٤۳۰۰٤

زهرا حیدری – ۹۸۲٤۳۰۲۰

PA Pins :

PIN	PA14	PA13	PA11	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
TYPE	O	O	O	I	I	I	O	O	O	O	I	I	I
FUNC	PWM_EN	PWM_EN	PWM	B3	B2	B1	LED	LED	LED	TX	RX	ADC	ADC

PB Pins :

PB0 برای خط اینتراپت،

PB1 تا PB7 خروجی‌های مربوط به سون‌سگمنت دهگان.

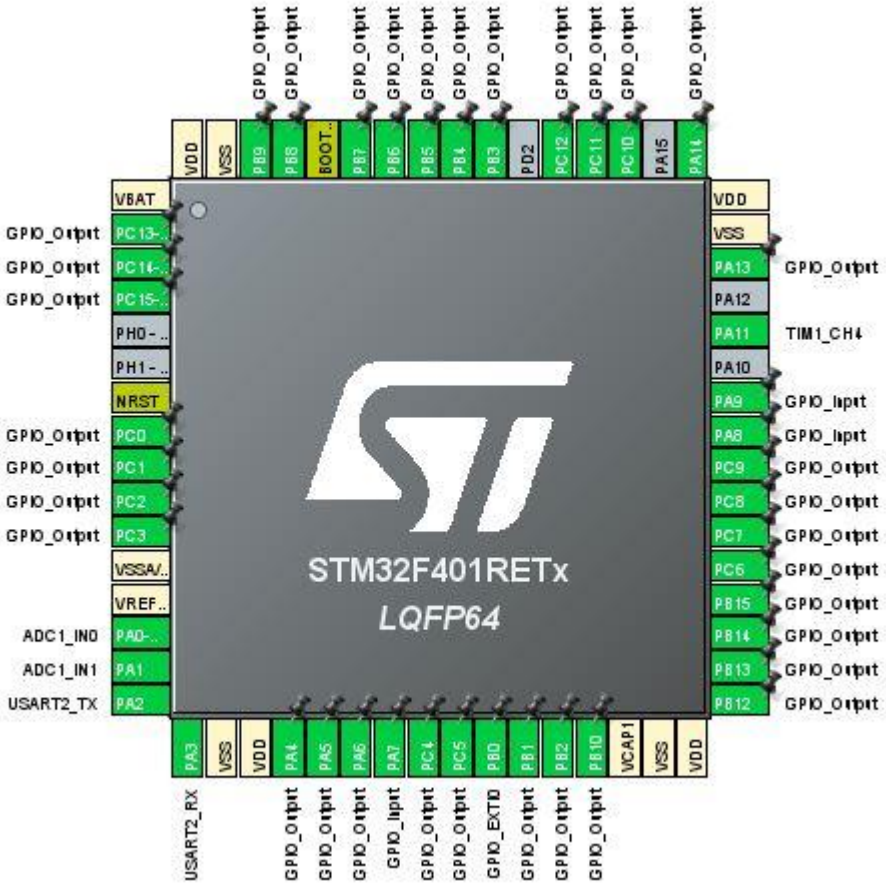
PB8 تا PB15 نیز مربوط به سون‌سگمنت یکان.

PC Pins:

تماما برای DAC (DAC ۱٦ ورودی دارد)

(در این پروژه به منظور دیباگ یک ترمینال اضافه کردیم و توابعی برای آن توسعه داده‌ایم از آنجا که این مورد جز الزامات پروژه نیست از شرح این توابع و نیز توابعی که در دستوکارهای پیشین تشریح شده‌اند، پرهیز می‌کنیم)

Pinout


STM32F401RETx
LQFP64
VDD
VSS
PB9
PB8
BOOT..
PB7
PB6
PB5
PB4
PB3
PD2
PC12
PC11
PC10
PA15
PA14
VBAT
PC13-..
PC14-..
PC15-..
PH0-..
PH1-..
NRST
PC0
PC1
PC2
PC3
VSSA/..
VREF..
PA0-..
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PB10
VCAP1
PA13
PA12
PA11
PA10
PA9
PA8
PC9
PC8
PC7
PC6
PB15
PB14
PB13
PB12
GPIO_Output
TIM1_CH4
GPIO_Input
ADC1_IN0
ADC1_IN1
USART2_TX
USART2_RX
GPIO_EXTI0

```
/* Private typedef -----------------------------------
/* USER CODE BEGIN PTD */
#define GREEN (4)
#define ORANGE (5)
#define RED (6)
#define B1 (7)
#define B2 (8)
#define B3 (9)
#define PWM_ORANGE (13)
#define PWM_RED (14)
#define SA (1)
#define SB (2)
#define SC (3)
#define SD (4)
#define SE (5)
#define SF (6)
#define SG (7)
#define SAA (8)
#define SBB (9)
#define SCC (10)
#define SDD (12)
#define SEE (13)
#define SFF (14)
#define SGG (15)
#define MASK(x)    (1UL << (x))
#define VREF (5)
```

در ابتدا پین‌هارا همانطور که پیشتر شرح داده‌شد دیفاین میکنیم (به‌منظور سهولت استفاده در کد)

```
/* USER CODE BEGIN 0 */
enum SystemState { STARTUP, OK, CHECKING, WARNING, COOLING, DANGER, OFF };
volatile enum SystemState systemState = STARTUP;
volatile float avg = 0.0;
volatile float lastAvg = 0.0;
volatile uint8_t count = 0;
```

ابتدا state های برنامه‌را طبق آنچه که در صورت‌پروژه مشروح است، تعریف می‌کنیم و حالت اولیه را روی Startup قرار می‌دهیم.

متغیر avg میانگین دمای فعلی و lastAvg میانگین دمای قبلی را نشان می‌دهد. Count نیز نشان میدهد چندمین سمپل دما از روی adc خوانده شده است.

```
// Transform System  to given State
void ChangeState(enum SystemState state){
    HAL_GPIO_WritePin( GPIOA, MASK(ORANGE) | MASK(GREEN) | MASK(RED) | MASK(PWM_RED) | MASK(PWM_ORANGE), GPIO_PIN_RESET);
    HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_4);
    systemState = state;
    switch( state ){
        case OK:
            HAL_GPIO_WritePin( GPIOA, MASK(GREEN), GPIO_PIN_SET);
            break;
        case WARNING:
            HAL_GPIO_WritePin( GPIOA, MASK(ORANGE), GPIO_PIN_SET);
            break;
        case CHECKING:
                HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
                HAL_GPIO_WritePin( GPIOA, MASK(PWM_ORANGE), GPIO_PIN_SET);
            break;
        case DANGER:
                HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
                HAL_GPIO_WritePin( GPIOA, MASK(PWM_RED), GPIO_PIN_SET);
            break;
        default:
            break;
    }
    logState(state);
}
```

این تابع تغییر حالت سیستم را هندل می‌کند متناسب با اینکه در هر حالت چه ال‌ای‌دی هایی بایستی روشن/خاموش/چشمک زن باشند، تنظیمات مربوطه را اعمال می‌کنیم.

```
}
// Show given Number on segments
void showOnSegments(uint16_t value){
        HAL_GPIO_WritePin(GPIOB, 0xFFFF , GPIO_PIN_RESET);
        uint16_t seg1 = segment1(value/10);
        uint16_t seg2 = segment2(value%10);
        HAL_GPIO_WritePin(GPIOB, (seg1) | (seg2) , GPIO_PIN_SET);
}

// Reverse Given Value
uint16_t reverse(uint16_t input){
    return ((4095 - input)  << 4);
}
```

در تابع اول صرفا مقدار داده‌شده را روی سون‌سگمنت‌های مربوط به دهگان و یکان مینویسیم. (تابع segment نیز در پروژه‌های گذشته شرح داده شده، صرفا مسک مربوط به نمایش عدد مربوطه روی پین‌های متناسب آن را بازمی‌گرداند.

```
/* Configurations for selecting/reading adc channels */
void Select_ADC_OSC(){
    ADC_ChannelConfTypeDef sConfig = {0};
    sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
}
void Select_ADC_TEMP(){
     ADC_ChannelConfTypeDef sConfig = {0};
    sConfig.Channel = ADC_CHANNEL_1;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
}
uint32_t Read_ADC_OSC(){
    HAL_TIM_Base_Stop_IT(&htim3);
    uint32_t raw;
    Select_ADC_OSC();
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    raw = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    HAL_TIM_Base_Start_IT(&htim3);
    return raw;
}

uint32_t Read_ADC_TEMP(){
        Select_ADC_TEMP();
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
        uint32_t raw = HAL_ADC_GetValue(&hadc1);
        HAL_ADC_Stop(&hadc1);
        return raw;
}
/**********************************************************************************/
```

این چهارتابع فرایند کانفیگ و نیز خواندن از پین آنالوگ را انجام میدهند، توابع SELECT چنل مربوطه را کانفیگ میکنند (چنل صفر که متناظر با PA0 است برای خواندن شکل موج ورودی و چنل یک که متناظر با PA1 است برای خواندن دما استفاده می‌شود) توابع READ ابتدا چنل مربوطه را SELECT می‌کنند سپس فرایند تبدیل را آغازکرده، و پس از خواندن مقدار آنرا متوقف می‌کنند.

```
void EXTI0_IRQHandler(){
        if( HAL_GPIO_ReadPin(GPIOA, MASK(B1)) == GPIO_PIN_SET ){
            if( systemState == STARTUP){
                ChangeState(CHECKING);
            }
        }
        else if( HAL_GPIO_ReadPin(GPIOA, MASK(B2)) == GPIO_PIN_SET ){
            if( systemState == OFF)
                ChangeState(CHECKING);
        }

}
```

دکمه‌های START و RESET را OR کرده و وارد خط وقفه می‌کنیم. اگر در STATE مربوط به STARTUP دکمه START زده شود به STATE مربوط به CHECKING می‌رویم. (دکمه Cooling با POLLING بررسی می‌شود)

```
// This Function handles Danger State. (Cooling Check , Turn Off,  .... )
void handleDanger(){
    uint32_t start = HAL_GetTick();
    bool coolingPressed = false;
    while( HAL_GetTick() - start < 1000 ){
        if( HAL_GPIO_ReadPin(GPIOA, MASK(B3)) == GPIO_PIN_SET ){
            coolingPressed = true;
            break;
        }
    }
    HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_4);
    HAL_GPIO_WritePin( GPIOA, MASK(PWM_RED) , GPIO_PIN_RESET);

    if( !coolingPressed )
        ChangeState(OFF);
    else {
            HAL_GPIO_WritePin( GPIOA, MASK(ORANGE) | MASK(GREEN) | MASK(RED) | MASK(PWM_RED) | MASK(PWM_ORANGE), GPIO_PIN_RESET);
            HAL_GPIO_WritePin( GPIOA, MASK(RED) , GPIO_PIN_SET);
            uint32_t miliSeconds = 0;
            while( miliSeconds < 2000 ){
                if( lastAvg <= 35){
                    ChangeState(OK);
                    return;
                }
                HAL_Delay(1);
                miliSeconds++;
            }
            HAL_GPIO_WritePin( GPIOA, MASK(RED) , GPIO_PIN_RESET);
            HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
            HAL_GPIO_WritePin( GPIOA, MASK(PWM_RED) , GPIO_PIN_SET);
            HAL_Delay(500);
            HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_4);
            ChangeState(OFF);
    }
}
```

این تابع حالت Danger را با استفاده از Systick هندل می‌کند. ابتدا 1 ثانیه (درصورت پروژه نیم‌ثانیه ذکر شده بود، برای اینکه تست کردن راحتتر شود آنرا 1 ثانیه کردیم) منتظر میمانیم (ال‌ای‌دی قرمز توسط pwm چشمک زن میشود) سپس چشمک زن خاموش می‌شود. اگر دکمه cooling فشرده نشده بود، سیستم وارد حالت OFF می‌شود در غیراینصورت، ال‌ای‌دی قرمز را

ثابت روشن می‌کنیم. سپس 2 ثانیه منتظر میمانیم، اگر دما به زیر 35 درجه رسید، وارد حالت OK می‌شویم در غیراینصورت پس از نیم‌ثانیه چشمک زدن ال‌ای‌دی قرمز وارد حالت OFF می‌شویم.

```
// Change AVG temprature periodically
void TIM3_IRQHandler(void){
        if( systemState != STARTUP && systemState != OFF  ){
            if( count == 10){

                count = 0;
                showOnSegments( (uint16_t) avg);
                if( avg <= 35){
                    if(systemState != OK)
                        ChangeState(OK);
                }
                if( avg > 35 && avg < 46){
                    if( systemState != WARNING)
                        ChangeState(WARNING);
                }
                else if( avg >= 46 ){
                    if( systemState != DANGER)
                        ChangeState(DANGER);
                }
                lastAvg = avg;
                avg = 0;
            }
            else{
                avg += readTemp()/10;
                count++;
            }
        }
}
```

در هندلر مربوط به TIM3 که هر 20 میلی‌ثانیه اجرا می‌شود، دما را میخوانیم و هر 10 بار آنرا روی سگمنت‌ها چاپ می‌کنیم. اگر میانگین دما کمتر از 35 باشد سیستم را وارد حالت OK کرده و برای حالات دیگر نیز طبق صورت‌پروژه عمل می‌کنیم.

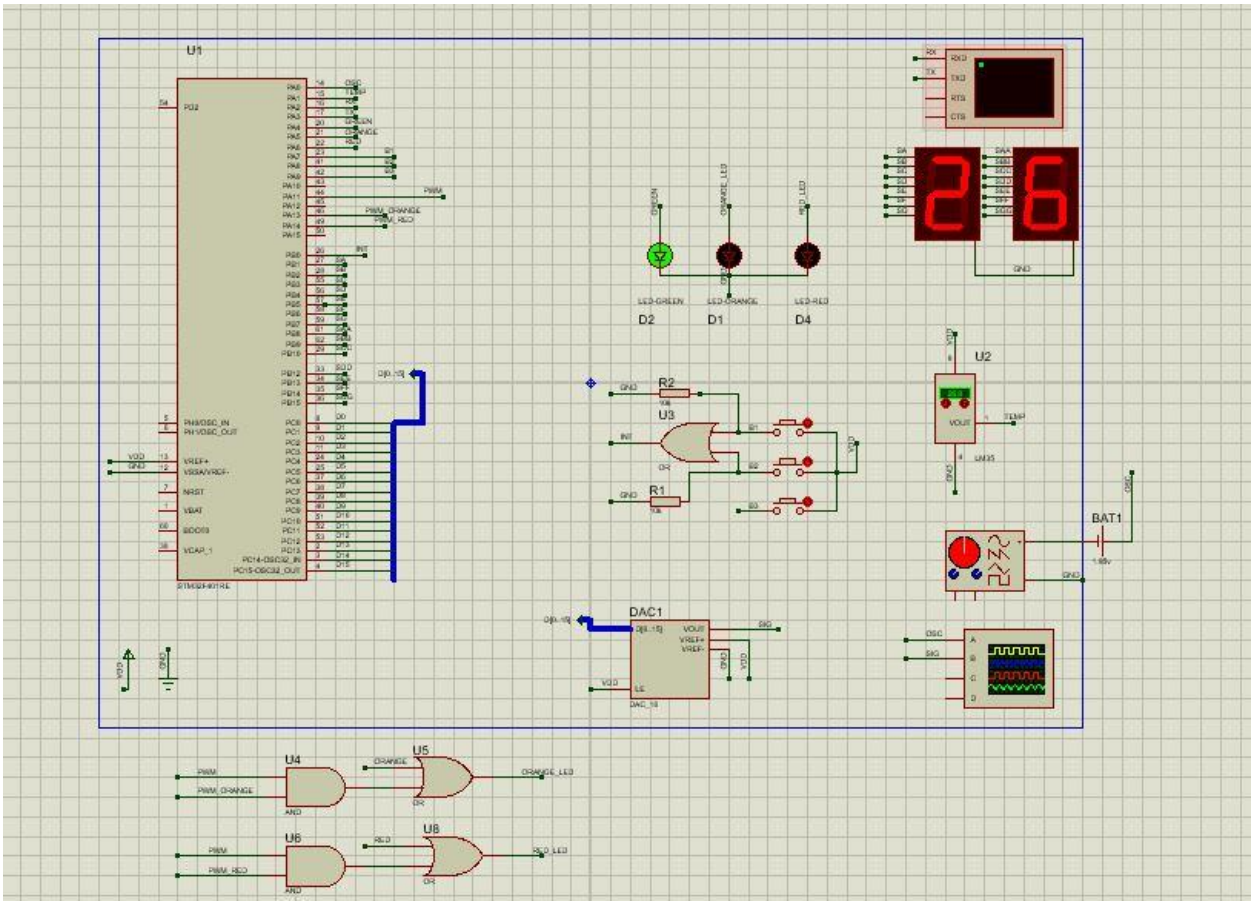
```
  while (1)
  {
        // If systemState has been changed to DANGER (via TIMER) handle it.
        if(systemState == DANGER)
            handleDanger();
        // if systemState is OK, read ADC, reverse and put on PC Pin.
        else if(systemState == OK ) {
            in = Read_ADC_OSC();
            GPIOC->ODR = reverse( in );
        }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

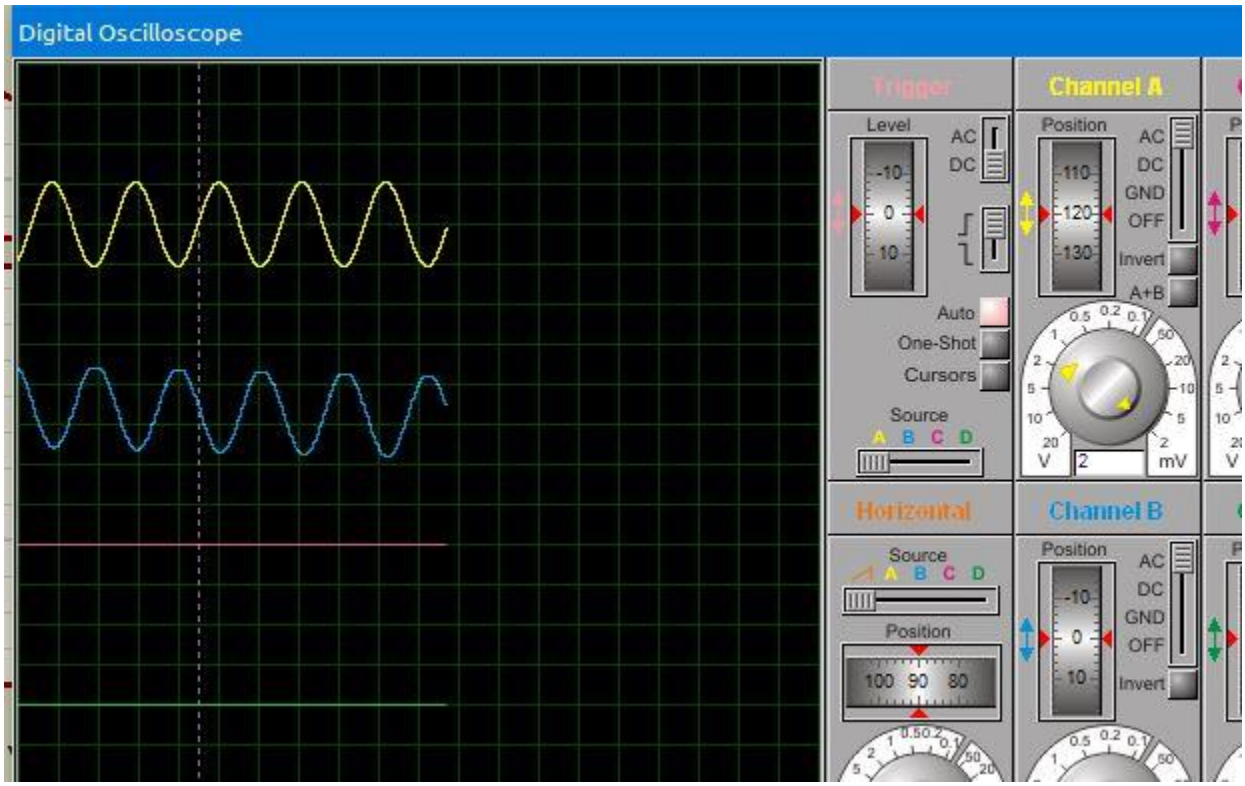
در حلقه اصلی برنامه مرتب چک می‌کنیم که آیا وارد حالت خطر شده‌ایم یا خیر؛ اگر اینطور بود اقدامات لازم را با تابعی که پیشتر شرح داده شد انجام میدهیم.

اگر سیستم در حالت OK بود نیز از موج ورودی نمونه‌گرفته و در خروجی قرار می‌دهیم.

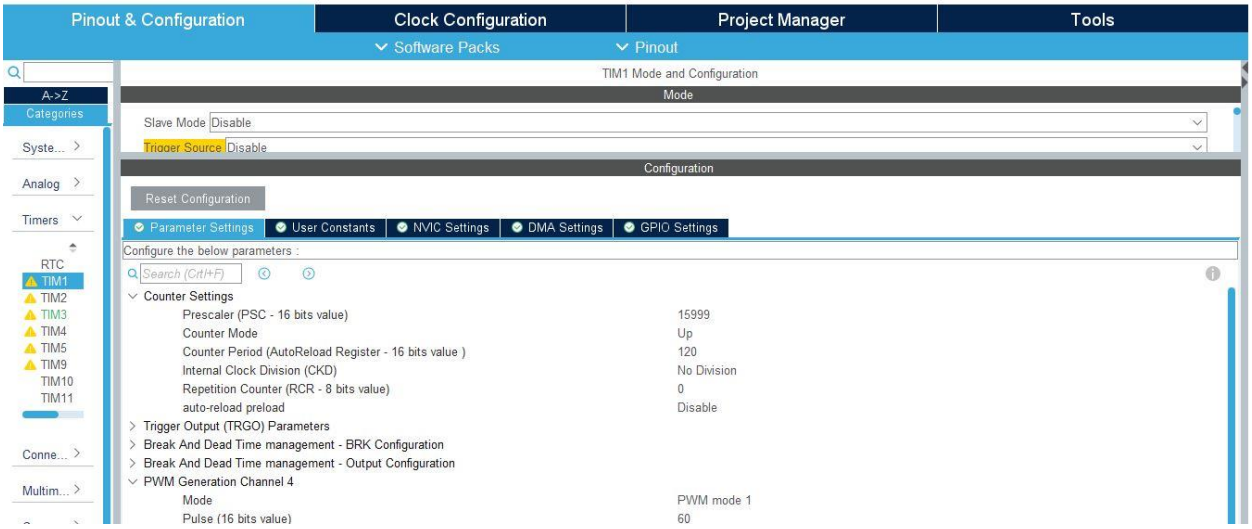
Proteus



InputWave-Reverse



تنظیمات تایمر 1 (برای PWM)



تایمر 3 (برای تعیین میانگین دوره‌ای دما با وقفه)