آزمایش 6

طاها موسوی 98243058

نیلوفر مرادی جم 97243063

گروه 2

سوالات تحلیلی:

1- مشخصات یک سیگنال PWM چیست؟

از مشخصات pwm میتوانیم به موارد زیر اشاره کنیم:

فرکانس: نشان میدهد چند پالس در هر ثانیه رخ میدهد همچنین این فرکانس ثابت است.

دوره تناوب: از طریق فرکانس بدست میاید یک تقسیم بر فرکانس دوره تناوب را به ما میدهد.

On time: مدت زمانی که پالس روشن است.

Duty-cycle: تقسیم on time بر دوره تناوب هر چقدر duty-cycle بزرگتر باشد یعنی توان بیشتری منتقل شده.

Adjust on-time : نشان دادن مقادیر انالوگ

2 - تفاوت مد یک و دو PWM در STM32F4 چیست؟

در مد اول در upcounting کانال اول فعال است تا زمانی که TIMx_CNT<TIMx_CCR1 باشد در غیر این صورت غیر فعال است و در downcounting کانال اول غیر فعال است تا زمانی که TIMx_CNT>TIMx_CCR1 در غیر این صورت فعال است.

در مد دوم برعکس است یعنی چنل اول غیر فعال است تا زمانی که TIMx_CNT<TIMx_CCR1 در غیر این صورت فعال است و در downcounting کانال اول فعال است تا زمانی که TIMx_CNT>TIMx_CCR1 در غیر این صورت غیر فعال است.

همچنین فاز شکل موج وقتی duty-cycle تغییر بدیم در مود اول تغییر نمیکند و در دومی تغییر میکند.

3 – چه تفاوتی در ساختارLED ها با رنگهای متفاوت وجود دارد؟ آیا می‌توان در هر مداری آنها را جایگزین یکدیگر نمود؟ شرح دهید.

TIMx_CCMR1(capture compare mode register 1) برای تنظیم کانال های یک و دو استفاده میشه و TIMx_CCMR2(capture compare mode register 2) برای تنظیم کانال های سه و چهار استفاده میشود.

capture compare mode register 1 هشت بیت سمت راست آن برای تنظیمات کانال یک و هشت بیت سمت چپ برای تنظیمات کانال دو میباشد.

capture compare mode register 2 هشت بیت سمت راست آن برای تنظیمات کانال سه و هشت بیت سمت چپ برای تنظیمات کانال چهار میباشد.

دو مد مختلف input capture و output compare دارند که

Input capture برای گرفتن سیگنال ورودی داده شده به میکرونترلر و اندازه گیری فرکانس و عرض پالس آن کاربرد دارد.

Output capture برای کنترل شکل موج خروجی یا نشان دادن دوره زمانی که سپری شده و همچنین برای تولید پالس های خروجی کاربرد دارد.

رفرنس های سوالات تحلیلی:

- کلاس درس و اسلاید های درسی

https://resources.pcb.cadence.com/blog/2020-pulse-width-modulation-characteristics-and-the-effects-of-frequency-and-duty-cycle#:~:text=PWM%20Characteristics&text=There%20are%20two%20primary%20components,a%20full%20ON-OFF%20cycle.

https://en.wikipedia.org/wiki/Pulse-width_modulation

http://www.ledified.com.au/understanding-the-colour-temperature-of-led-lights/#:~:text=LED%20lights%20come%20in%20a,light%20appears%20crisp%20and%20white.

دستور کار:

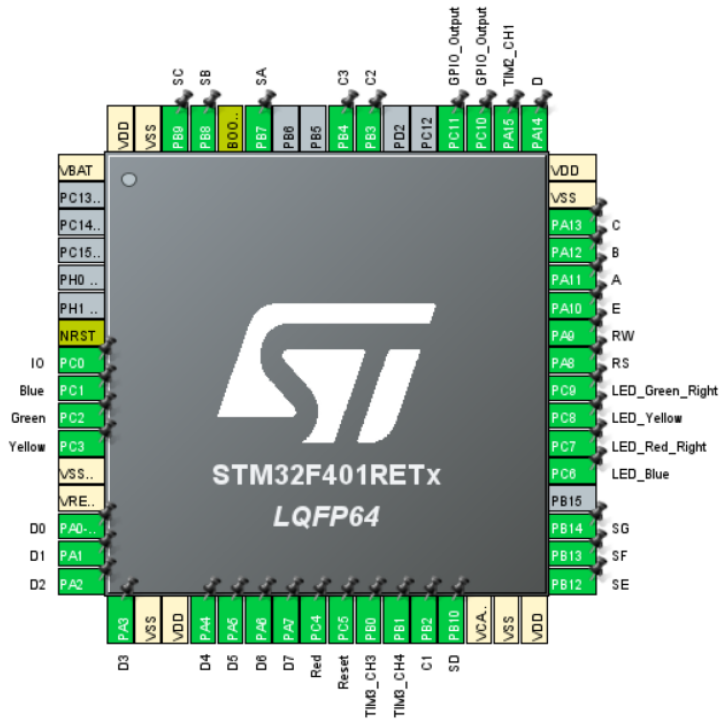
ورودی خروجی‌ها:

pin	PA15	PA 14	PA1 3	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
symbol	RESET	D	C	B	A	E	RW	RS	D7	D6	D5	D4	D3	D2	D1	**D0**
type	IC	O	O	O	O	O	O	O	O	O	O	O	O	O	O	**O**

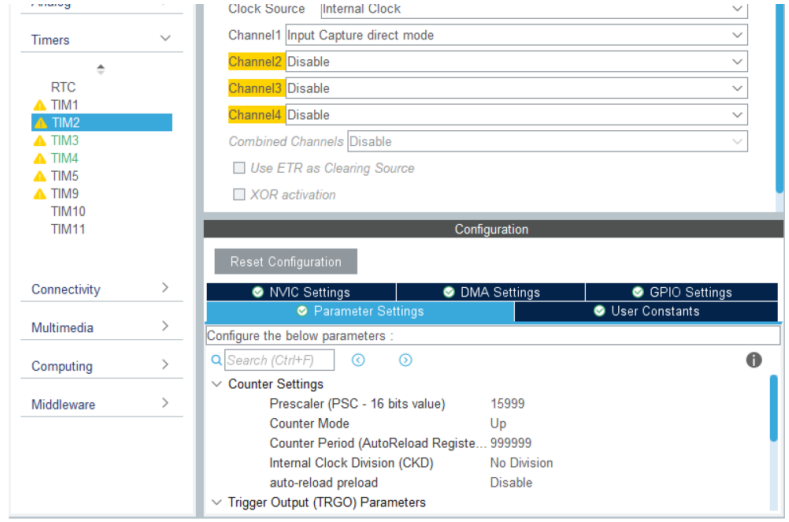
PIN	PB14	PB13	PB12	PB10	PB9	PB8	PB7	PB4	PB3	PB2	PB1	PB0
SYM	SG	SF	SE	SD	SC	SB	SA	C3	C2	C1	RED PWM	GREEN PWM
TYPE	O	O	O	O	O	O	O	I	I	I	TIM3-CH3-PWM	TIM3-CH4-PWM

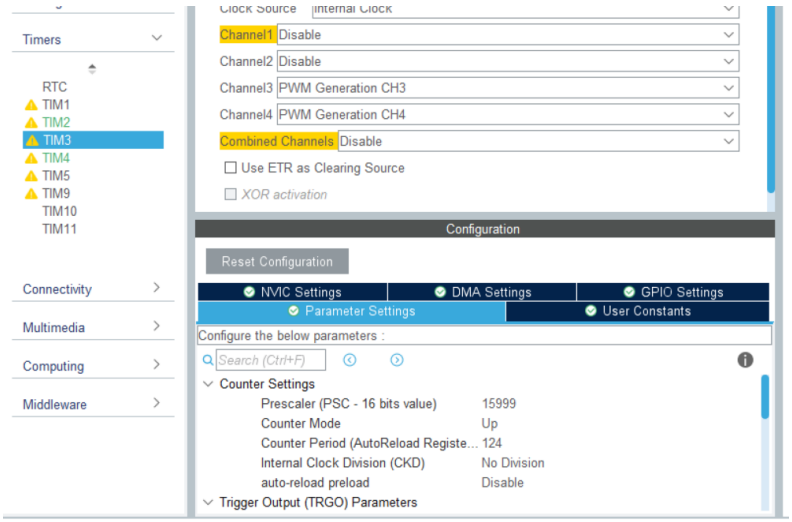
PIN	PC11	PC10	PC9	PC8	PC7	PC6	PC 5	PC 4	PC3	PC2	PC1	PC 0
SYM	RED_ PWM_ EN	GREEN_PW M_ EN	LED_ RED	LED_ YELLOW	LED_ GREEN	LED_BL UE	RES ET	RE D	YELL OW	GREEN	BLU E	I0
TYPE	o	O	O	O	O	O	I	I	I	I	I	int err up t

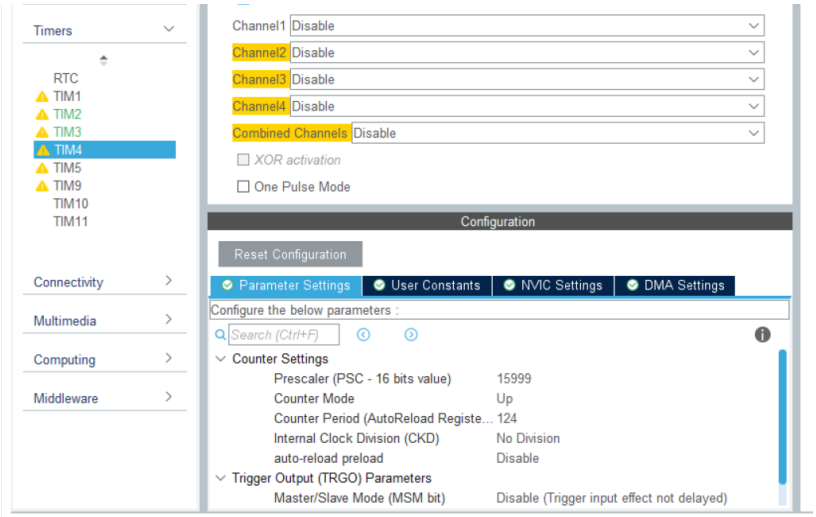
طرح کلی:



تایمر ۴ برای محاسبه فاصله زمانی میان toggle کردن نام است و از چنل اول تایمر ۲ نیز برای input-capture روی خط reset و از چنلهای ۳ و ۴ تایمر را برای pwm استفاده میشود.







numberOfAllowedMisses تعداد خطای مجاز که برابر 3 است.

timeOfGame تایم کلی بازی را نشان میدهند.

آرایه inputID برای شماره دانشجویی میباشد.

تعدادخطاها، متغیری برای نشاندادن و حفظ اسم متناظر با شماره دانشجویی ورودی، حالت برنامه، تایم باقیمانده، و شکل نتیجه نیز تعریف میشوند.

اینام هایی برای مشخص کردن استیت بازی و همچنین برنده و بازنده و در حال بازی بودن تعریف شده است.

Current color نیز رنگ ال ای دی است که در حال حاضر روشن است.

```
62
63  static bool printedEND_OF_GAMEType = false;
64  volatile static unsigned int zero_counter = 0;
65  const uint16_t timeOfGame = 60, numberOfAllowedMisses = 3;
66  enum State { START,
67               PRINTING_NAME,
68               RUNNING,
69               END_OF_GAME };
70  enum END_OF_GAMEType {  PLAYING,
71                          WIN,
72                          LOOSE};
73  volatile enum State gameState = START;
74  volatile enum END_OF_GAMEType END_OF_GAMEType = PLAYING;
75  volatile char inputID[8] ,currentColor='b';
76  volatile uint16_t misses = 0, inputIDIndex = 0, remainingTime = timeOfGame, currentRow = 0;
77  char showingName[30];
78  uint32_t count = 0;
79  bool errorPassed = false;
80  uint32_t lastCaptured = 0;
81
82  void clearFirstLine();
83  int32_t charToInt(char ch);
84  char intToChar(int32_t digit);
85  void printInLine(char* string);
86  void printNum(int N);
87  void init_lcd();
88  void delay(uint16_t ms);
89  void commitCommand(unsigned char command);
90  void commitChar(char data);
91  uint16_t sevenSegment(uint16_t value);
92  void loose();
93  void win();
94
95  /* USER CODE END Includes */
```

تابع مربوط به آپدیت سون سگمنت:

```
/* USER CODE BEGIN 0 */
uint16_t sevenSegment(uint16_t value) {
  if(value == 0){
    return MASK(SA) | MASK(SB) | MASK(SC) | MASK(SD) | MASK(SE) | MASK(SF);}
  else if (value==1){
    return MASK(SB) | MASK(SC);}
  else if (value==2){
    return MASK(SA) | MASK(SB) | MASK(SD) | MASK(SE) | MASK(SG);}
  else if (value==3){
    return MASK(SA) | MASK(SB) | MASK(SC) | MASK(SD) | MASK(SG);}
  else if (value==4){
    return MASK(SB) | MASK(SC) | MASK(SF) | MASK(SG);}
  else if (value==5){
    return MASK(SA) | MASK(SC) | MASK(SD) | MASK(SF) | MASK(SG);}
  else if (value==6){
    return MASK(SA) | MASK(SC) | MASK(SD) | MASK(SE) | MASK(SF) | MASK(SG);}
  else if (value==7){
    return MASK(SA) | MASK(SB) | MASK(SC);}
  else if (value== 8 ){
    return MASK(SA) | MASK(SB) | MASK(SC) | MASK(SD) | MASK(SE) | MASK(SF) | MASK(SG);}
  else if (value==9){
    return MASK(SA) | MASK(SB) | MASK(SC) | MASK(SD) | MASK(SF) | MASK(SG);}
  else{
    return MASK(SA) | MASK(SB) | MASK(SC) | MASK(SD) | MASK(SE) | MASK(SF);}
}
```

این تابع برنامه و متغیرها را ریست میکند و درواقع شرایط را به حالت اولیه میبرد:

```
// reset all program states & vars.
void resetTheProgram(){
  END_OF_GAMEType = PLAYING;
  HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_4);
  HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_3);
  HAL_GPIO_WritePin(GPIOC, MASK(RED_PWM_ENABLE), GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, MASK(GREEN_PWM_ENABLE), GPIO_PIN_RESET);
  gameState = START;
  clearFirstLine();
  clearLEDs();
  remainingTime = timeOfGame;
  misses= count= inputIDIndex = 0;
  //Show misses
  HAL_GPIO_WritePin( GPIOB, 0xFF << 7, GPIO_PIN_RESET);
  HAL_GPIO_WritePin( GPIOB, sevenSegment(misses), GPIO_PIN_SET);

  commitCommand(0x80);
  MX_TIM3_Init();
  HAL_TIM_Base_Start_IT(&htim3);
}
```

در این تابع ال ای دی با رنگ فعلی روشن و بقیه ال ای دی ها خاموش میشوند:

```
void showColor(char color){
  if(color == 'g'){ // show green color on leds.
      HAL_GPIO_WritePin(GPIOC, MASK(LED_BLUE) | MASK(LED_RED) |MASK(LED_YELLOW) , GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOC, MASK(LED_GREEN), GPIO_PIN_SET);
  }else if(color == 'y'){// show yellow color on leds.
      HAL_GPIO_WritePin(GPIOC, MASK(LED_BLUE) | MASK(LED_RED) |MASK(LED_GREEN) , GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOC, MASK(LED_YELLOW), GPIO_PIN_SET);
  }else if(color == 'b'){// show blue color on leds.
      HAL_GPIO_WritePin(GPIOC, MASK(LED_GREEN) | MASK(LED_RED) |MASK(LED_YELLOW) , GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOC, MASK(LED_BLUE), GPIO_PIN_SET);
  }else if (color =='r'){// show red color on leds.
      HAL_GPIO_WritePin(GPIOC, MASK(LED_BLUE) | MASK(LED_GREEN) |MASK(LED_YELLOW) , GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOC, MASK(LED_RED), GPIO_PIN_SET);
  }
}
```

این تابع یک کارکتر رندوم برای رنگ ال ای دی به ما برمیگرداند:

```
431
432 char generateRandomColor(){
433   int val = (int) rand() % 4;
434   int val2 = 0;
435   switch(currentColor){
436     case 'b':
437       val2 = 0;break;
438     case 'y':
439       val2 = 1;break;
440     case 'g' :
441       val2 = 2;break;
442     case 'r':
443       val2 = 3;break;
444   }
445   if(val == val2){ // Different color
446     val = (val+1) % 4;
447   }
448   switch(val){
449     case 0:
450       currentColor= 'b';break;
451     case 1:
452       currentColor= 'y';break;
453     case 2:
454       currentColor= 'g'; break;
455     case 3:
456       currentColor= 'r';break;
457   }
458   return currentColor;
459 }
```

این تابع بررسی میکند که کاربر کلید کدام رنگ فشرده است:

```
420
421 char clickedLED(){
422   if( HAL_GPIO_ReadPin( GPIOC, MASK(BLUE)) == GPIO_PIN_SET)
423     return 'b';
424   else if (HAL_GPIO_ReadPin( GPIOC, MASK(GREEN)) == GPIO_PIN_SET)
425     return 'r';
426   else if (HAL_GPIO_ReadPin( GPIOC, MASK(YELLOW)) == GPIO_PIN_SET)
427     return 'y';
428   else if (HAL_GPIO_ReadPin( GPIOC, MASK(RED)) == GPIO_PIN_SET)
429     return 'g';
430 }
431
```

این تابع همه LED هارا خاموش میکند:

```
182 void clearLEDs(){
183   HAL_GPIO_WritePin(GPIOC, MASK(LED_GREEN) | MASK(LED_RED) |MASK(LED_YELLOW) | MASK(LED_BLUE) , GPIO_PIN_RESET);
184 }
```

اینتراپت درصورت فشردن کیپد یا Reset اجرا میشود که در حالت START صرفا ورودی‌هارا نشان میدهیم وقتی ۸ کاراکتر وارد شد، نام متناظر با آن را متعاقب ورود به حالت PRINTING_NAME به‌صورت چشمک زن نشان می‌دهیم:

```
501 }
502 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
503   if(GPIO_PIN_0) {
504       char clickedButton = clickKeyPadBtn();
505       if( gameState != START ){
506         if(clickedButton == '1' || clickedButton == '*')
507           resetTheProgram();
508       }
509       else{
510         if( clickedButton != 'e')
511           inputID[inputIDIndex++] = clickedButton;
512         if(inputIDIndex == 8){
513           clearFirstLine();
514           commitCommand(0x80);
515           if(inputID[0] == '9' && inputID[1] == '7'
516             && inputID[2] == '2' && inputID[3] == '4'
517             && inputID[4] == '3'&& inputID[5] == '0'
518             && inputID[6] == '6' && inputID[7] == '3')
519             setName("Moradi Jam");
520           else if(inputID[0] == '9' && inputID[1] == '8'
521                   && inputID[2] == '2' && inputID[3] == '4'
522                   && inputID[4] == '3' && inputID[5] == '0'
523                   && inputID[6] == '5' && inputID[7] == '8')
524             setName("Mousavi");
525
526           gameState = PRINTING_NAME;
527         }
528         else if( clickedButton != 'e')
529             commitChar(clickedButton);
530
531       }
532   }
533 }
```

در این تابع زمانیکه ۴ لبه (پایین رونده و بالارونده) دیده شود، به‌طوریکه اگر فاصله بین لبه‌های بالارونده بیش از ۲ ثانیه باشد، سیستم وارد حالت RUNNING شده و بازی اجرا می‌شود (runGame) :

```
614
615 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
616   if( !errorPassed){
617     errorPassed = true;
618     return;
619   }
620   if( gameState != PRINTING_NAME)
621     return;
622
623   count++;
624   // first and third edges (rising)
625   if(count == 1 ){
626     lastCaptured = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
627   }
628   // second edge  (falling)
629   else if(count == 2){
630     uint32_t currentCaptured = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
631     if( currentCaptured - lastCaptured < 2000/16)
632       count = 0;
633       lastCaptured = 0;
634   }
635   else if(count == 3){
636     lastCaptured = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
637   }
638   // check wether appropriate pulse has been seen or not.
639   else if(count == 4) {
640     uint32_t currentCaptured = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
641     if( currentCaptured - lastCaptured >= 2000/16)
642      {
643       count = 0;
644       currentRow = 3;
645       lastCaptured = 0;
646       gameState = RUNNING;
647       HAL_GPIO_WritePin(GPIOA, MASK(D), GPIO_PIN_SET);
648       HAL_GPIO_WritePin(GPIOA, MASK(A)|MASK(B)|MASK(C) , GPIO_PIN_RESET);
649       remainingTime = timeOfGame;
650       runGame();
651     }
652     else
653       count = 0;
654       lastCaptured = 0;
655   }
656
657
658 }
```

تایمر 4 که تایمر Base است مسئولیت چشمک زدن نام را دارد.

تایمر 3 نیز در حالت Base اگر سیستم درحالت RUNNING باشد و تایم تمام نشده باشد، در هرثانیه آنرا Decrement میکند و نشان می‌دهد. اگر تایم تمام شود و تعداد خطاها کمتر از آستانه تعریف شده باشد؛ بازی با برد و در غیر اینصورت با باخت خاتمه یافته است:

```
561
562 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
563   // decrementing time and checking win-conditions
564   if( htim->Instance == TIM3){
565     if( gameState == RUNNING) {
566       clearFirstLine();
567       remainingTime--;
568       if(remainingTime >0){
569         printNum(remainingTime);
570       }
571       else {
572         if(misses == numberOfAllowedMisses )
573             loose();
574         else
575             win();
576       }
577     } else if(gameState == END_OF_GAME){
578       if(END_OF_GAMEType == WIN){
579         if(printedEND_OF_GAMEType)
580           clearFirstLine();
581         else
582           printInLine("Winner");
583         printedEND_OF_GAMEType = !printedEND_OF_GAMEType;
584       }else{
585         if(printedEND_OF_GAMEType)
586         clearFirstLine();
587       else
588         printInLine("Looser");
589       printedEND_OF_GAMEType = !printedEND_OF_GAMEType;
590       }
591     }
592   }
593
594   // for toggling name
595   if( htim->Instance == TIM4) {
596     if( gameState == PRINTING_NAME){
597       static bool printed = false;
598       if(printed)
599         clearFirstLine();
600       else
601         printInLine(showingName);
602       printed = !printed;
603     }
604   }
605
```

با توجه به دستورکار، در تابع runGame ۳ مرتبه ۳ ثانیه‌ای LED روشن و ورودی چک می‌شود و سپس ۳ مرتبه ۲ثانیه‌ای و پس آن طبق دستورکار ۱۰ مرتبه ۱۰ مرتبه با کسر ۱۰۰ میلی‌ثانیه روال انجام می‌شود. در تابع checkingAndShowStatus نیز یک رنگ رندوم ایجاد و LED مربوطه را روشن میکنیم و زمان مشخصی مرتبا ورودی کاربر را چک میکنیم درصورتیکه کاربر کلید درست در تایم مربوطه وارد نکند misses اضافه میشود، اگر به تعداد مجاز خطا برسد، loose کال می‌شود:

```
485 void runGame(){
486   checkingAndShowStatus(3000/8);
487   checkingAndShowStatus(3000/8);
488   checkingAndShowStatus(3000/8);
489   checkingAndShowStatus(2000/8);
490   checkingAndShowStatus(2000/8);
491   checkingAndShowStatus(2000/8);
492   uint16_t delay = 1000;
493     while(time>0) {
494       if(gameState != RUNNING)
495         break;
496       for(int i=0; i < 10; i++)
497         checkingAndShowStatus(delay/8);
498       delay -= 100;
499     }
500 }
```

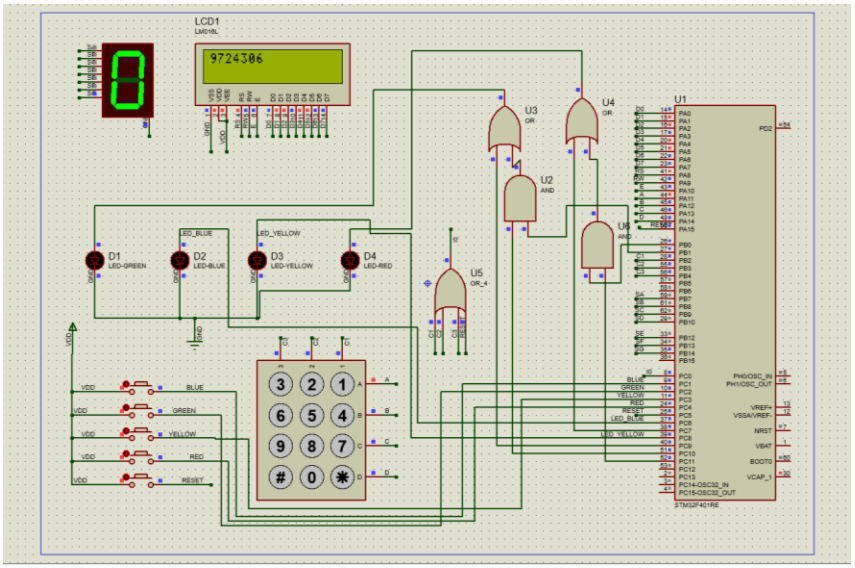
```
461 void checkingAndShowStatus(uint16_t delayms){
462   if( gameState != RUNNING)
463     return;
464   bool passed = false;
465   char rand = generateRandomColor();
466   showColor(rand);
467   uint32_t start = HAL_GetTick();
468   while( (HAL_GetTick() - start) < delayms){
469     if(clickedLED() == rand)
470       passed = true;
471   }
472   if(!passed){
473     misses++;
474
475     //show misses
476     HAL_GPIO_WritePin( GPIOB, 0xFF << 7, GPIO_PIN_RESET);
477     HAL_GPIO_WritePin( GPIOB, sevenSegment(misses), GPIO_PIN_SET);
478
479     if(misses >= numberOfAllowedMisses){
480       loose();
481     }
482   }
483 }
```

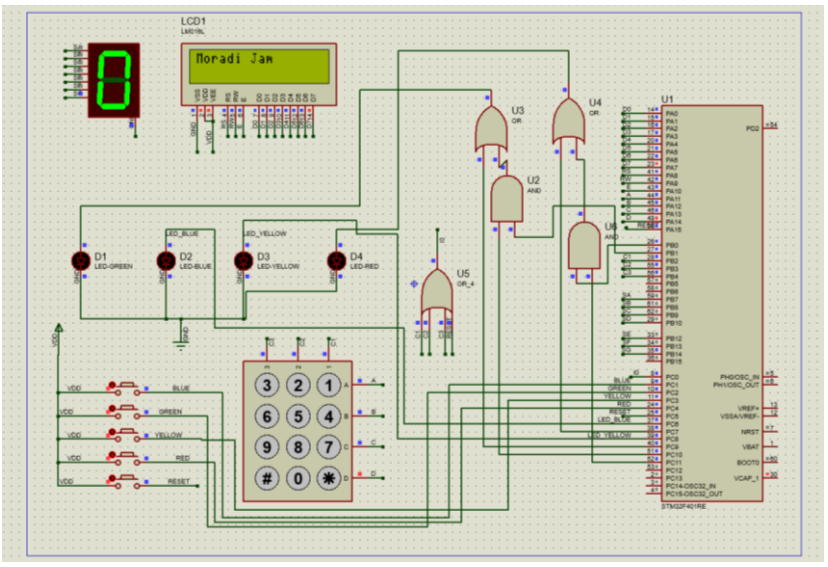
در تابع loose() اگر بازی به حالت WIN باشد برای جلوگیری از اختلال زمانی loose نبایستی اجرا شود .در غیر اینصورت این تابع حالت نتیجه را LOOSE و حالت سیستم را در حالت END_OF_GAME میگذارد.
سپس PWM مربوط به باخت (قرمز) را فعال میکن و عبارت لوزر را چاپ میکند.

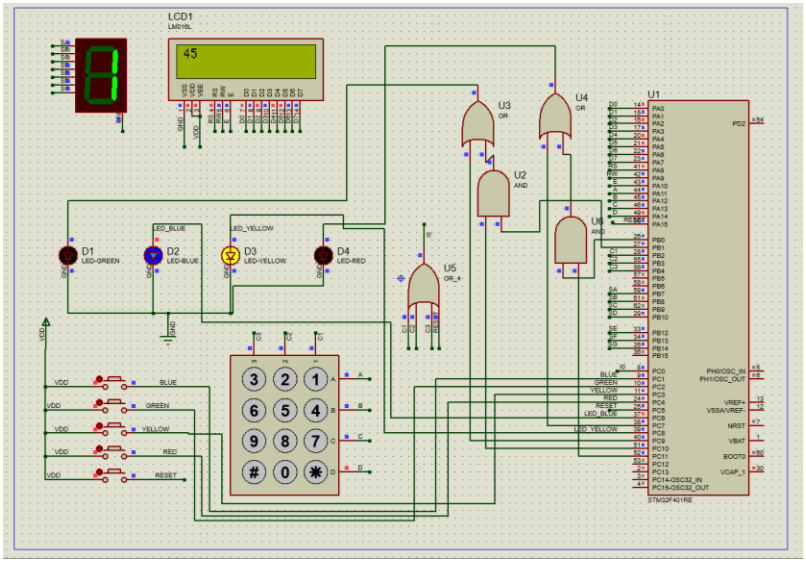
متقارن همین تابع WIN وجود دارد.

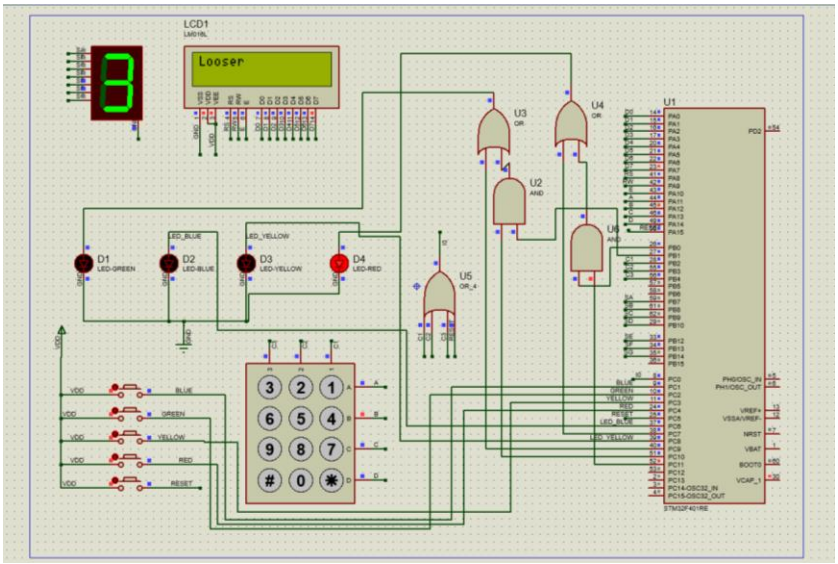
```
534
535 void win(){
536   if(END_OF_GAMEType == LOOSE)
537     return;
538   END_OF_GAMEType = WIN;
539   clearLEDs();
540   gameState = END_OF_GAME;
541   HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4);
542   HAL_GPIO_WritePin(GPIOC, MASK(GREEN_PWM_ENABLE), GPIO_PIN_SET);
543   clearFirstLine();
544   //printInLine("Winner");
545 }
546
547 void loose(){
548   if(END_OF_GAMEType == WIN)
549     return;
550   END_OF_GAMEType = LOOSE;
551   clearLEDs();
552   gameState = END_OF_GAME;
553   HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);
554   HAL_GPIO_WritePin(GPIOC, MASK(RED_PWM_ENABLE), GPIO_PIN_SET);
555   clearFirstLine();
556   //printInLine("Looser");
557 }
558
```

نتیجه کار در پروتئوس:




LCD1
Moradi Jam
U1
U2
U3
U4
U5
D1
LED-GREEN
D2
LED-BLUE
D3
LED-YELLOW
D4
LED-RED
BLUE
GREEN
YELLOW
RED
RESET
STM32F401RE


LCD1
45
U1
U2
U3
U4
U5
D1
LED-GREEN
D2
LED-BLUE
D3
LED-YELLOW
D4
LED-RED
BLUE
GREEN
YELLOW
RED
RESET
STM32F401RE


LCD1
Looser
U1
U2
U3
U4
U5
D1
LED-GREEN
D2
LED-BLUE
D3
LED-YELLOW
D4
LED-RED
BLUE
GREEN
YELLOW
RED
RESET
STM32F401RE