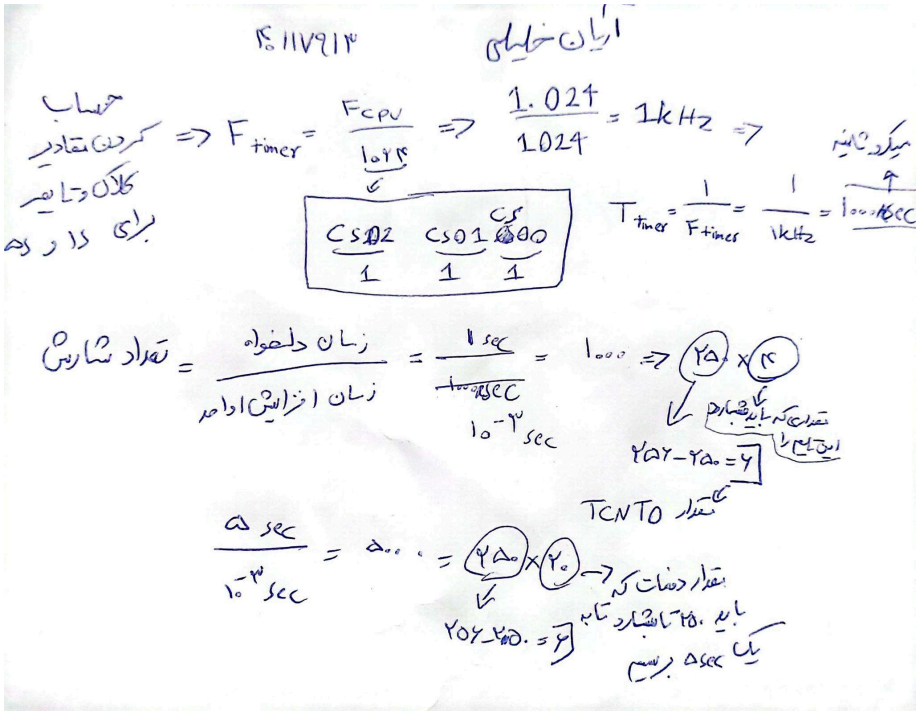
سیستم دیجیتال 2 پروژه نهایی

پروژه بازی حدس عدد

آریان خلیلی بروجنی
40117913

توضیحات پروژه قسمت تایمر:

حساب تایمر: برای حساب تایمر محاسبات داخل عکس انجام شده است و همچنین برای TCNT0وTCCR0وTIMSK را مقادیر میدهیم.



توضیحات قسمت تعاریف اولیه کد:

Initialize های اولیه برای کد مثل تعریف پورت ها که خروجی و ورودی است و وقفه 5 که با لبه پایین رونده کار

میکند پس باید به EIMSK , EICRB مقادیر مناسب داد تا درست فعال سازی شود و برای شمارش تایم های 1 ثانیه و 5 ثانیه هم نیاز به رجیستر داشتیم طبق محاسبات بالا پس دو تا رجیستر برای آن رفته یک کانتر برای شمارش تعداد راند بازی شده و یک کانتر هم برای شمارش اینکه کی بازی برده هر دست و در آخر هم R22 برای جمع کردن برای تعداد برد های بازیکن 1 که باید از بیت 4 یکی یکی زیاد بشه و پشته و ... هم تعریف شده. و در آخر کد بعد تعریف کردن همه این ها SEI میزنیم تا بتواند وارد وقفه شود و بعد وارد یک حلقه بی نهایت میشویم تا وقتی که دکمه استارت وقفه 5 توسط بازیکن 2 زده بشه!

```
.INCLUDE "m64def.inc"
.ORG 0x0000
  JMP Main
.ORG 0x000C
  JMP INT5_ISR
.ORG 0x0020
  JMP TIMER0_OVF_ISR
.ORG 0x0050
Main:
    LDI   R16, HIGH(RAMEND)
    OUT   SPH, R16
    LDI   R16, LOW(RAMEND)
    OUT   SPL, R16
    LDI   R17, 0 ;COUNTER ROUNDS
    LDI   R18, 0 ;COUNTER 1SEC
    LDI   R19, 0 ;COUNTER 5SEC
    LDI   R21, 0 ;COUNTER PLAYER_1,2 WINS
    LDI   R22, 0x10 ;inc hast vali yejoraii baray part 2 register

;-----------TIMER------------
    LDI   R16, 6
    OUT   TCNT0, R16
    LDI   R16, 0x07
    OUT   TCCR0, R16
    LDI   R16, 0x81
    OUT   TIMSK, R16
;-----------INT5------------
    LDI   R16, 0x20
    OUT   EIMSK, R16
    LDI   R16, 0x08
    OUT   EICRB, R16
;-----------PORT------------
    LDI   R16, 0x0F
    OUT   DDRC, R16
    OUT   PORTC, R16
    OUT   DDRE,  R16
    LDI   R16, 0x00
    OUT   DDRA, R16
    LDI   R16, 0x0FF
    OUT   DDRD, R16
```

توضیحات قسمت وقفه 5 :

در وقفه 5 کار خاصی نکرده و فقط اول SEI را فعال میکنیم و بعد مقدار مانتر برد های بازیکن 1 و 2 را در پورت بی ذخیره میکنیم همونجوری که گفته شد 4 بیت اول برای بازیکن دوم و 4 بیت دوم برای بازیکن اول هست بخاطر محدودیت پورت ها در ATMEGA هر دو رو در یک بخش زدم و بعد همه رجیستر های که بعد هر بازی باید ریست بشوند را پاک کردم و وارد محسابات و .... میشویم

```
    SEI
LOOP:
    JMP   LOOP
;-------------------------------------------------------------------------
INT5_ISR:
    SEI
    OUT  PORTB, R21
    CLR  R17
    CLR  R18
    CLR  R19
    CLR  R23
    JMP  CHECK_ANSWER

;-------------------------------------------------------------------------
```

توضیحات بیس کد و چک کردن جواب و بعد مقایسه کردن دو عدد بازیکن 1 و 2 در پورت ها به کجا رفته:

در قسمت چک انسر ابتدا رجیستر 17 را زیاد کرده چون شمارنده تعداد راند ها هستش و بعد داخل پورت E ریخته و بعد با استفاده از پورت سی LED سفید را روشن کرده که بیت اولش هست و بهش مقدار صفر داده و بعد با پاک کردن مقدار رجیستر 19 که برای 5 ثانیه صبر کردن است وارد لیبل WAIT شده و تا وقتی به 20 برسد که طبق محاسبات اولیه حساب شده در بالای PDF هست برمیگردد به خود لیبل بعد از صبر کردن تقریبا 5 ثانیه مقادیر پورت های A,D را خوانده که از بازیکن های 1 و 2 هست و نسبت به مقایسه کردن آن دو عدد به لیبل های مختلف میرود. اگر مقدار داده شده توسط

بازیکن دو کمتر بوده به AS , اگر مساوی بوده به AE و در آخر اگر بزرگتر

بوده به AB می رود و در آنجا کار نهایی انجام میشود.

```
CHECK_ANSWER:
    INC   R17
    OUT   PORTE,  R17
    LDI   R16, 0x0E
    OUT   PORTC, R16
    CLR   R19
WAIT_JAVAB:
    CPI   R19,20
    BRNE  WAIT_JAVAB
    IN    R16, PINA
    IN    R23, PIND
    CP    R16, R23
    BRLO  AS
    CP    R16, R23
    BREQ  AE
    JMP   AB
```

در AS چه کرده:

در آن اول کل LED ها را خاموش کرده و بعد بیت 4 ام یا در اینجا بهش 3 اشاره شده چون از صفر شروع میکنیم در برنامه نویسی پس آن بیت را صفر کرده تا روشن شود و بعد از پاک کردن رجیستر 18 با استفاده ازش 1 ثانیه صبر کرده در لیبل WAIT که باز هم این مقدار 4 در محاسبات بالا انجام شده و بعد از آن چک میکند ببیند بازی 10 راندش تموم شده یا نه وگرنه ادامه میدهد.

```
;-----------------------------
AS:
    LDI   R16, 0x0F
    OUT   PORTC, R16
    CBI   PORTC, 3
    CLR   R18
WAIT_S:
    CPI   R18,4
    BRNE  WAIT_S
    CPI   R17,10
    BRNE  CHECK_ANSWER
```

در AB چه کرده:

در آن اول کل LED ها را خاموش کرده و بعد بیت 3 ام یا در اینجا به2 اشاره شده چون از صفر شروع میکنیم در برنامه نویسی پس آن بیت را صفر کرده تا روشن شود و بعد از پاک کردن رجیستر 18 با استفاده ازش 1 ثانیه صبر کرده در لیبل WAIT که باز هم این مقدار 4 در محاسبات بالا انجام شده و بعد از آن چک میکند ببیند بازی 10 راندش تموم شده یا نه وگرنه ادامه میدهد.

```
AB:
    LDI   R16, 0x0F
    OUT   PORTC, R16
    CBI   PORTC, 2
    CLR   R18
WAIT_B:
    CPI   R18,4
    BRNE  WAIT_B
    CPI   R17,10
    BRNE  CHECK_ANSWER
    JMP   ETMAM_GAME
```

در AE چه کرده:

در آن اول کل LED ها را خاموش کرده و بعد بیت 2 ام یا در اینجا به 1 اشاره شده چون از صفر شروع میکنیم در برنامه نویسی پس آن بیت را صفر کرده تا روشن شود و بعد با پاک کردن رجیستر 19 با استفاده ازش 5 ثانیه صبر میکنیم در لیبل WAIT که باز هم این مقدار 20 در محاسبات بالا انجام شده و بعد از آن پورت سی را همه را یک کرده تا تمام LED ها خاموش شود و بعد تعداد راند ها را ریست کرده چون بازی تموم شده با برد بازیکن 2 پس یکی هم به مقدار شمارنده تعداد برد های بازیکن 2 اضافه کرده و از وقفه 5 بیرون آمده و به حلقه برمیگردیم تا وقفه بعدی!

```
;-------------------------
AE:
    LDI   R16, 0x0F
    OUT   PORTC, R16
    CBI   PORTC, 1
    CLR   R19
WAIT_E:
    CPI   R19,20
    BRNE  WAIT_E
    SBI   PORTC, 1
    LDI   R16,  0x00
    OUT   PORTE,  R16
    INC  R21
    RETI
```

بعد اتمام مقایسه ها:

بعد تموم شدن مقایسه ها و روشن کردن LED لازمه و صبر کافی باید بریم راند بعدی مگر اینکه سبز آمده باشد که بازی کلا تمام شود ولی اگر زرد یا قرمز آمده بود به لیبل ETMAM_GAME رفته تا کار های نهایی انجام داده تیکه اول برای امتیازی است که بعدا توضیح داده میشود وقتی به اینجا میاییم که که ده راند تموم شده پس تمام LED ها باید به مدت 5 ثانیه روشن بماند و بعد بازی تموم شود و یک دست به بازیکن 1 برسد که پس یک WAIT 5 ثانیه ای می خواهیم که تا 20 باید بره و بعدش تمام LED هارا ریست کرده و راند رو به صفر برگردونده و بعد از جمع کردن برد بازیکن 1 به 7SEGMENT خودش از وقفه 5 بیرون اومده و در لوپ چرخیده تا دست جدید شروع بشه

```
ETMAM_GAME:
;-----------------------------------------
    IN      R30, OCR2
    SUBI    R30,10
    OUT     OCR2, R30
;--------WAS FOR BOUNES THE PREVIOUS LINES
    LDI   R16,0x00
    OUT   PORTC,  R16
    CLR  R19
    WAIT_ETMAM:
    CPI   R19,  20
    BRNE  WAIT_ETMAM
    LDI   R16,  0x0FF
    OUT   PORTC,  R16
    LDI   R16,  0x00
    OUT   PORTE,  R16
    ADD  R21, R22
    RETI
```

لیبل تایمر در آن چی نوشته؟:

در تایمر دوباره مقدار 6 را در TCNT0 ریخته چون هر بار از اول ریست میشود طبیعتا وقتی دور اول تا 256 رفته بعد از صفر برای دور بعد شروع میکند که 5 ثانیه صبر خراب میشود بعد مقادیر رجیستر 18و19 اضافه کرده تا در لیبل های WAIT بشود دقیق به مقادیر لازم برسیم و در آخر هم با RETI کردن به وقفه 5 برمیگردیم از وقفه تایمری که رفته بودیم داخلش.

```
;
TIMER0_OVF_ISR:
    LDI   R16, 6
    OUT   TCNT0, R16
    INC   R18
    INC   R19
;---------------------
END:
    RETI
```

امتیازی:

در قسمت امتیازی باید با استفاده از fast pwm بعد هر راند درخشش LED سفید را کاهش داده نکته مهم این هست که با کار های زیر این کار امکان پذیر است:

برای فعال کردن Fast PWM روی تایمر صفر، مراحل کلی به شکل زیر است:

1. **انتخاب مد Fast PWM**
   - در تایمر صفر (هشت‌بیتی)، بیت‌های WGM21 و WGM20 (در رجیستر TCCR2) تعیین‌کنندهٔ حالت کاری هستند.
   - برای Fast PWM باید این دو بیت برابر 1 باشند: (WGM21=1, WGM20=1).

در ATmega64 رجیستر TCCR2 به این صورت است:

```
:TCCR2

 0      1      2      3      4      5      6      7
| FOC2 | WGM20 | COM21 | COM20 | WGM21 | CS22 | CS21 | CS20 |
```

   - پس برای انتخاب Fast PWM، باید WGM20=1 و WGM21=1 باشد.
2. **تنظیم نوع خروجی مقایسه (Compare Output Mode)**
   - برای اینکه سیگنال PWM روی پایه OC2 مشاهده شود، باید از مد غیرمعکوس‌کننده (Non-inverting) یا معکوس‌کننده (Inverting) استفاده کنیم.
   - برای غیرمعکوس‌کننده (مثبت) باید COM21=1 و COM20=0 تنظیم شود.
   - این کار باعث می‌شود که وقتی شمارنده تایمر از مقدار OCR2 کمتر است، پایه OC2 در سطح بالا و وقتی از OCR2 عبور می‌کند، پایه OC2 به سطح پایین برود.
3. **تنظیم Prescaler (بیت‌های CS2 [2:0])**
   - برای انتخاب فرکانس PWM و سرعت شمارش تایمر از تقسیم‌گر کلاک استفاده می‌شود.
   - مثلاً اگر بخواهیم تقسیم‌گر 1 (بدون تقسیم) داشته باشیم، CS22=0, CS21=0, CS20=1.
   - بسته به فرکانس کاری میکروکنترلر (مثلاً 16MHz، 8MHz، 1MHz) باید این بخش را طوری انتخاب کنید که فرکانس خروجی مطلوب به‌دست آید. که من با 256 حلش کردم
4. **تنظیم مقدار اولیه‌ی دیوتی‌سایکل (OCR2)**
   - رجیستر OCR2 در مد Fast PWM مقدار Compare را مشخص می‌کند.
   - هرچه مقدار OCR2 بزرگ‌تر باشد، سیگنال PWM مدت بیشتری در سطح بالا خواهد ماند (Duty Cycle بیشتر) و LED پرنورتر خواهد بود.

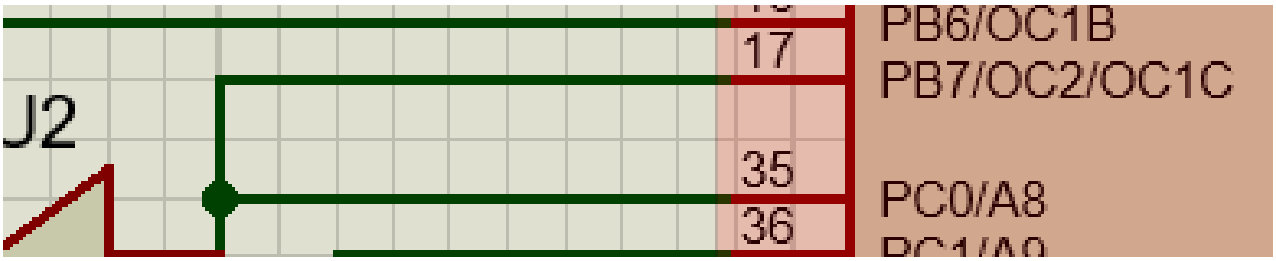
بنابراین مثلاً اگر بخواهیم در ابتدای کار دیوتی‌سایکل ما ۱۰۰٪ باشد (حداکثر نور)، مقدار OCR2 را ۰xFF می‌گذاریم. اگر بخواهیم حدود ۵۰٪ باشد (نیمه‌روشن)، OCR2 را ۰x7F قرار می‌دهیم و الی‌آخر.

در آخر کد پس ما مقدار OCR2 را با 10 کم کرده و مقدار جدید را ذخیره کرده . و در عکس سخت افزار سخت افزار همونجوری که قابل مشاهده هست باید از خروجی OC2 به LED سفید هر جا که ورودی میگیرد وصل کرد به همان سیم پس نتیجه نهایی به این شکل میشود و نکته نهایی مهم این است که در PROTEUS این کم شدن نور خوب نمایش داده نمیشود پس برای همین در فیلم اشاره نشده بهش ولی مفهومش پیاده سازی شده:

```
;----------------------------------------
    IN      R30, OCR2
    SUBI    R30,10
    OUT     OCR2, R30
;--------WAS FOR BOUNES THE PREVIOUS LINES
```

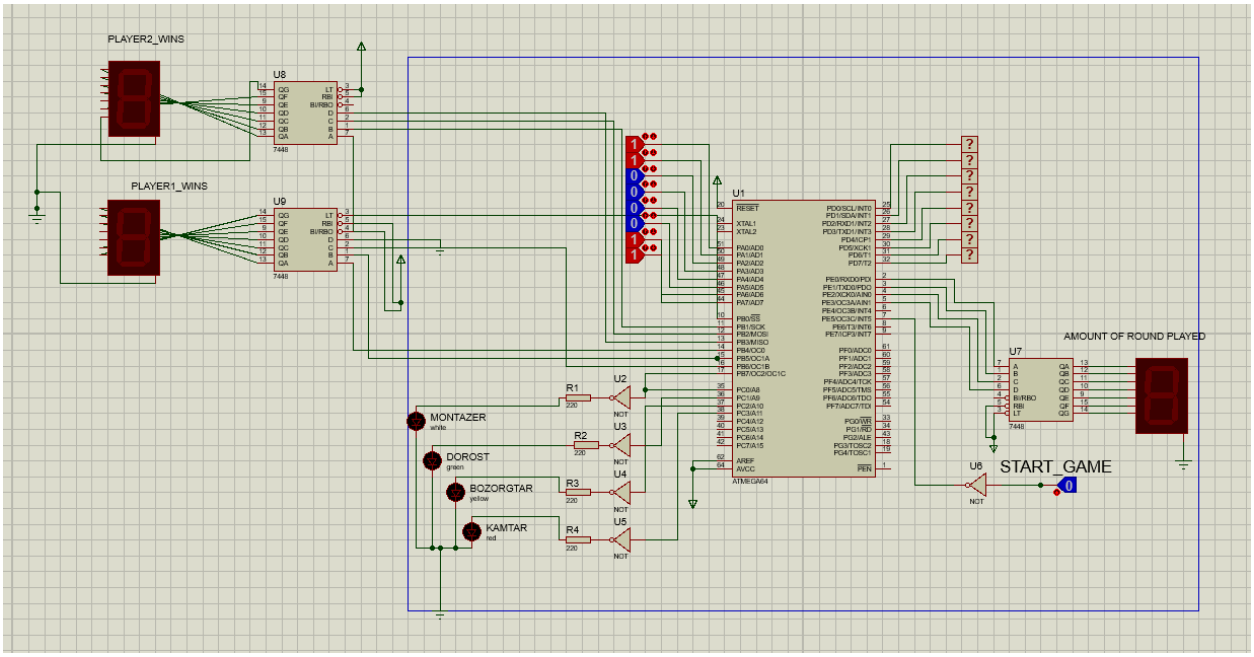
```
;----------BOUNES-------
    LDI     R16,  0x78
    OUT     TCCR2, R16

    ldi     R16, 0x77
    OUT     OCR2, R16
```



نکات نهایی:

در آخر توضیحات سخت افزار در فیلم داده شده و همچنین نحوه ران و اجرا شدن پروژه همچنین برای مقدار اولیه f cpu سعی کردم دیتاشیت atmega 64 بخوانم که 414 صفحه از مطالب بود پس بعد پیدا نکردنش سعی کردم با تست کردن مقادیر مختلف نزدیک ترین جواب را پیدا کنم که دیدم با 1.024Mhz یا 1MHZ خوب کار میکنه و نزدیک زمان های 1 ثانیه و 5 ثانیه هست داخل فیلم میتونید زمان تقریبی ببینید که نزدیک هست.


PLAYER2_WINS
PLAYER1_WINS
U8
U9
U1
U2
U3
U4
U5
U6
U7
7448
ATMEGA64
R1
R2
R3
R4
220
NOT
MONTAZER
white
DOROST
green
BOZORGTAR
yellow
KAMTAR
red
START_GAME
AMOUNT OF ROUND PLAYED

"خسته نباشید"