دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه درس ریزپردازنده و زبان ماشین

(طراحی سیستم کنترل چراغ راهنمایی دو زمانه یکه چهارراه توسط میکروکنترلر)

تهیه کننده:

سید عرفان نوربخش

استاد درس:

استاد ماهوش

تیر ۱۴۰۱

در این پروژه میخواهیم ولتاژ آنالوگ پتانسیومتر را توسط پورت A ای وی آر تبدیل به دیجیتال بکنیم و از آن مقدار دیجیتال برای روشن کردن چراغ های راهنمایی استفاده بکنیم.

در ابتدا دو تا مقدار value1 و value2 را تعریف کرده که مقدار دیجیتالی ولتاژ آنالوگ دو پتانسیومتر ها در آن ذخیره میشود. (در واقع وقتی جریان داخل ADC می‌شود که ولتاژ آنالوگ به دیجیتال تبدیل کند و مقدار دیجیتالی شده ولتاژ داخل آن ذخیره میشود.)

در ادامه مقادیری برای ذخیره کردن مقادیر سبز و زرد و قرمز ماندن هر چراغ تعریف کرده که مقدار زرد را ثابت فرض کرده و مقدار lossTime هم داریم بدلیل اینکه در واقعیت زمان بسیار کمی دو چراغ روبرو در چهارراه ها هر دو قرمز بوده تا چهارراه خلوت شود.

سپس پورت C را تبدیل به خروجی می‌کنیم. (چون قرار است که از داخل به خارج AVR جریان برود و چراغ ها روشن بشوند که همین پورت را برای وصل کردن به چراغ‌های راهنمایی و رانندگی انتخاب می‌کنیم.) پورت C و تنها ۶ ورودی اول آن را که ۳تای اول به چراغ اول و ۳تای دوم به چراغ دوم متصل است را فعال می‌کنیم.

سپس در ابتدا با فرض اینکه فقط چراغ سبز از چراغ راهنمایی اول و چراغ قرمز از چراغ راهنمایی دوم روشن باشد پورت آن را مقداردهی می‌کنیم. (0b00001100)

```
{
    double value1,value2; // Define 2 variables(value1 & value2) for put the result of each potentiometer
    double green1,green2,red1,red2; //Define red & green for each traffic light delay
    double totalTime; //Define variable to save the value of each period
    double yellow=2,lossTime=1; //Define variable for yellow time and also for loss time in red state of traffic light
    DDRC=0b00111111; // Convert Port C as an output port (1-3 for traffic light1 & 4-6 for traffic light2)
    PORTC=0b00001100; // First time the green light in traffic light 1 & red light in traffic light 2 is on
```

در ادامه یک حلقه while داریم که میخواهیم به ازای هر دوره زمانی که از ابتدای سبز چراغ اول تا دوباره سبز شدن آن این اجرا شود و همینطور این عملیات تکرار می‌شود. در ابتدای آن یک تابع ADC_Init را فراخوانی کرده که در ادامه به شرح آن می‌پردازیم.

در تابع ADC_Init پورت A را تبدیل به ورودی کرده (بدلیل اینکه قرار است از خارج به داخل AVR یک جریان وارد شود همچنین برای مشخص کردن وضعیت هر پورت از رجیسترهای DDR استفاده می‌کنیم که برای هر ۴ تا پورت رجیسترهای جداگانه‌ای وجود دارد. دلیل انتخاب پورت A این است که این پورت ADC بوده و مبدل تبدیل آنالوگ به دیجیتال می‌باشد.) در ادامه باید مبدل آنالوگ به دیجیتال را فعال بکنیم که از طریق رجیستر ADCSRA انجام می‌شود که مقدار آن را برابر 0x87 قرار می‌دهیم.(با استفاده از جدول‌های مبدل آنالوگ به دیجیتال بنابر کاربر می‌توان مشخص کرد که چه مقداری باید داخل آن ریخته شود که در این پروژه ما

می‌خواستیم ADC را فعال بکنیم.) رجیستر بعدی ADMUX بوده که مبدل تبدیل آنالوگ به دیجیتال یک مالتی‌پلکسر دارد که از طریق آن مشخص می‌کنیم گه ADC ما قرار است چه کاری انجام بدهد و از کجا قدرت (Power) بگیرد و همچنین چگونه تبدیل بکند. (از این رجیستر برای مقداردهی مالتی‌پلکسر استفاده می‌شود که این رجیستر ۸ بیتی بوده که ۲ بیت پرارزش آن مشخص کننده این است که مالتی پلکسر از چه منبع AVR قدرت (Power) بگیرد که بر فرض مثال در این پروژه می‌توان هم از AREF و هم از AVCC قدرت گرفت که ما از AVCC استفاده می‌کنیم. علاوه بر آن، از طریق آن می‌توان مشخص کرد از که کدام پین‌های پورت A فعال باشند تا از آنها جریان عبور کند و در مالتی‌پلکسر ۴ بیت کم‌ارزش آن مشخص کننده این است که کدام پین‌ها می‌توانند مقدار بگیرند یعنی اینکه از این ۴بیتی که داخل رجیستر است می‌توان پین‌ها را فعال کرد که برای شروع پین صفر را قرار می‌دهیم که منظور همان channel=0 می‌باشد که باز هم این مورد با استفاده از جدول مخصوص به خود مقداردهی می‌شود.)

```
/*
    ** FunctionName: ADC_Init
    * Description: Initialize the ADC
*/
void ADC_Init()
{
    DDRA=0x0;   // Make ADC port as input
    ADCSRA = 0x87;  // Enable ADC-fr/128
    ADMUX = 0x40;   // AVCC-ADC channel: 0
}
```

در ادامه دو بار تابع ADC_Read با مقادیر ورودی مختلف ۰ و ۱ فراخوانی می‌شوند که عملیات تبدیل آنالوگ به دیجیتال و برگرداندن محتوا انجام می‌شود که ورودی گرفته شده مشخص می‌کند که از کدام پین‌های پورت A میخواهیم جریان را عبور بدهیم. در ادامه توسط عملیات بیتی، ۴ بیت مالتی‌پلکسر مقداردهی می‌شود و پین موردنظر فعال می‌شود که حتی می‌توانیم نگذاریم و همان مقداری صفری که از اول مقداردهی کرده بودیم را قرار بدهیم و فقط از پین صفر جریان را عبور بدهیم. در ادامه عملیات تبدیل آنالوگ به دیجیتال انجام شده که صبر می‌کنیم تا این تبدیل به اتمام برسد. بعد از به اتمام رسیدن آن، دو رجیستر ADCL و ADCH را داریم که مقادیر دیجیتالی داخل این دو رجیستر ۸ بیتی ذخیره می‌شوند که مقدارهای کم‌ارزش داخل ADCL و مقدارهای پرارزش داخل ADCH ذخیره می‌شوند. سپس بعد از ذخیره شدن، این دو رجیستر با همدیگر جمع شده و مقدار نهایی برگردانده می‌شود.

```
/*
    ** FunctionName: ADC_Read
    * Description: Operation of convert Analog to Digital
*/
int ADC_Read(char channel) // Input of port as an input of function
{
    int Ain,AinLow; // Define 2 variable to read bytes
    ADMUX=ADMUX|(channel & 0x0f);   // Set input channel to read
    ADCSRA |= (1<<ADSC);    // Start conversion
    while((ADCSRA&(1<<ADIF))==0);   // Monitor end of conversion interrupt-Wait until conversion finished
    _delay_us(10);  //Take 10microSeconde delay
    AinLow = (int)ADCL;     // Read lower byte
    Ain = (int)ADCH*256;        // Read higher 2 bits & multiply with weight */
    Ain = Ain + AinLow;     //Add 2 variables Ain & AinLow
    return(Ain);        // Return digital value
}
```

در ادامه باتوجه به گفته مسئله درمورد پتانسیومتر اول زمان کل یک دوره را محاسبه می‌کنیم. (به این صورت محاسبه می‌شود که ما مقدار جریان از ۰ تا ۱۰۲۳ بوده و مقدار مینیمم و ماکزیمم را ۱۴ و ۲۴ بوده که اگر مقدار lossTime را در این مورد درنظر بگیریم ۱۶ و ۲۶ می‌شود چون باید ۲ ثانیه یکی برای چراغ اول یکی برای چراغ دوم درنظر بگیریم که جمعا ۲ثانیه به کل دوره اضافه می‌شود. که این دو مقدار به همدیگر تبدیل میشوند که با استفاده از معادله درجه ۱ بدست می‌آید.)

$0*a+b=14 \Rightarrow b=14$

$1023*a+b=24 \Rightarrow a=10/1023$

$totalTime=((10*X)/1023)+14$

سپس مقدار سبز ماندن چراغ اول را بدست می‌آوریم که مقدار مینیمم آن را ۵ درنظر گرفته و باز هم مقدار ۰ تا ۱۰۲۳ جریان را به مقدار که باز هم مانند قسمت قبلی محاسبه می‌شود که مقدار جریان آن که به کل نسبت داده می‌شود (یعنی value/1023) را در مقداری که برای سبز میخواهیم (totalTime -5-5-2-2) میکنیم که در واقع خواهیم داشت:

$green1=(((totalTime-14) * value) / 1023) + 5$

برای مدت زمان سبز بودن چراغ دوم هم باید زمان کل دوره را از کل سبز بودن چراغ اول و همچنین جمع زرد بودن دو تا چراغ کم کرد تا مقدار آن بدست آید، پس داریم:

$green2=totalTime-green1-2-2$

در ادامه زمان‌های قرمز بودن را هم میتوان بدست آورد که دقیقا معکوس همدیگر می‌شوند. یعنی برای مدت زمان قرمز بودن چراغ دوم برابر است با زمان سبز بودن چراغ اول + زمان زرد بودن چراغ اول و برای مدت زمان قرمز بودن چراغ اول برابر است با تفریق کل زمان از زمان سبز و زرد بودن آن.

red1=totalTime-green1-2

red2=green1+2

```
while (1) // While loop for each duration of traffic light (1 iteration of each traffic light)
{
    ADC_Init(); // Call ADC_Init function
    value1=ADC_Read(0); // Put returned value of function ADC_Read to value1 by passing 0(first input of port A)
    value2=ADC_Read(1); // Put returned value of function ADC_Read to value2 by passing 1(second input of port A)
    totalTime=( ( 10 * value1 ) / 1023 ) + 14; // Calculate totalTime from linear equation(multiply the descent of current and the value1 and after that add to
    green1=  (((totalTime-14) * value2) / 1023) + 5; // Calculate the green time of traffic light 1 (multiply the descent of current and the totalTime - minimum
    green2=totalTime-green1-4; // Calculate the green time of traffic light 2(totalTime - the green time of traffic light 1 - yellow time of traffic light 1 - y
    red1=totalTime-2-green1; // Calculate the red time of traffic light 1(total Time - the yellow time of traffic light 1 - the green time of traffic light 1)
    red2=green1+2; // Calculate the red time of traffic light 2(sum of the green time of traffic light 1  and the yellow time of traffic light 1 => because it s
```

در ادامه که به منطق اصلی کد می‌رسیم به این صورت است که باتوجه به زمان‌هایی که بدست آوردیم در هر موقعیت تاخیر مربوط به آن را اعمال می‌کنیم.

- ✓ حالت اولیه
  - ❖ چراغ اول: سبز
  - ❖ چراغ دوم: قرمز
- ✓ حالت دوم (زمان تاخیر green1)
  - ❖ چراغ اول: زرد
  - ❖ چراغ دوم قرمز
- ✓ حالت سوم (زمان تاخیر yellow که ثابت است)
  - ❖ چراغ اول: قرمز
  - ❖ چراغ دوم قرمز
- ✓ حالت چهارم (زمان تاخیر lossTimeکه مقداری است که هر دو قرمز می‌مانند)
  - ❖ چراغ اول: قرمز
  - ❖ چراغ دوم قرمز
- ✓ حالت پنجم (زمان تاخیر green2)
  - ❖ چراغ اول: قرمز
  - ❖ چراغ دوم سبز
- ✓ حالت ششم (زمان تاخیر yellow که ثابت است)
  - ❖ چراغ اول: قرمز
  - ❖ چراغ دوم زرد
- ✓ حالت هفتم (زمان تاخیر lossTimeکه مقداری است که هر دو قرمز می‌مانند)
  - ❖ چراغ اول: قرمز

❖ چراغ دوم قرمز

✓ حالت هشتم (دیگر مقدار آن به حالت اولیه برگشته و دوباره این چرخه تکرار می‌شود. این مورد بهتر بود حالت نباشد اما برای درک بهتر مراحل آورده شده است.)

❖ چراغ اول: سبز

❖ چراغ دوم قرمز

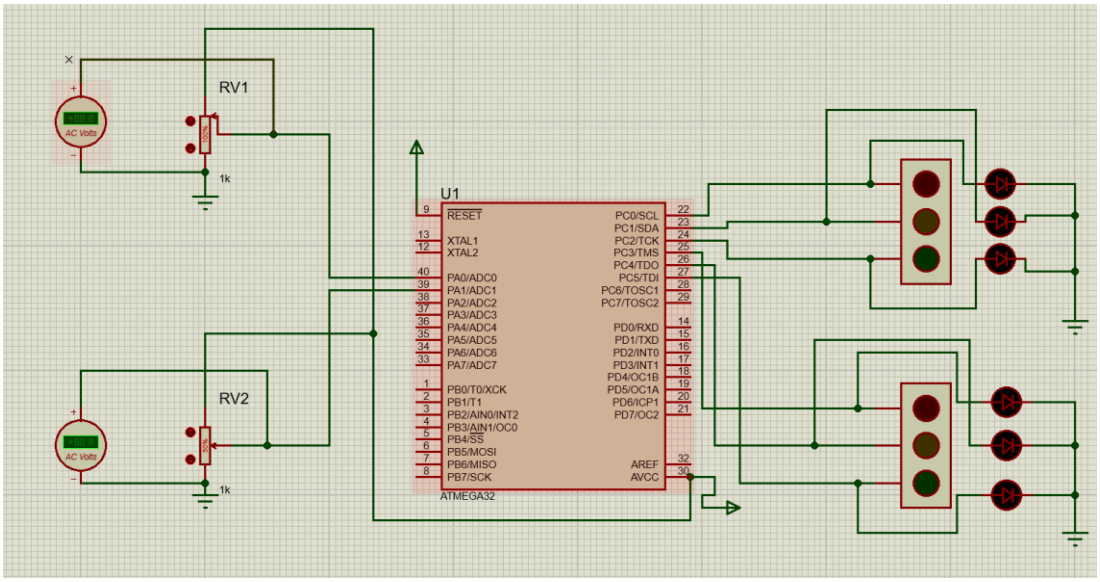
```
    Delay(green1); // Take delay of green time of traffic light 1
    PORTC=0b00001010; // traffic light 1(yellow) - traffic light 2(red)
    Delay(yellow); // Take delay of yellow time of traffic light 1
    PORTC=0b00001001; // traffic light 1(red) - traffic light 2(red)
    Delay(lossTime); // Take delay of lossTime of traffic light 1
    PORTC=0b00100001; // traffic light 1(red) - traffic light 2(green)
    Delay(green2); // Take delay of green time of traffic light 2
    PORTC=0b0010001; // traffic light 1(red) - traffic light 2(yellow)
    Delay(yellow); // Take delay of yellow time of traffic light 2
    PORTC=0b0001001; // traffic light 1(red) - traffic light 2(red)
    Delay(lossTime); // Take delay of lossTime time of traffic light 2
    PORTC=0b00001100; // traffic light 1(green) - traffic light 2(red)
```

مقدار Delay راه هم با نوشتن یک تابع پیاده سازی می‌کنیم که در آن با حلقه while کنترل می‌شود که چه مقدار باید تاخیر وجود داشته باشد و همچنین به ازای هر کم شدن یک مقدار از آن delay از تابع خودش یعنیdelay_ms استفاده می‌کنیم تا مقدار واقعی تر بدست آید.

```
/*
    ** FunctionName: Delay
    * Description: Custom delay that working with _delay_ms for each second
*/
void Delay (double delay){
    while (delay>0){ // Wait until delay will be zero or negative
        _delay_ms(100); // 100 miliSecond delay for each decrease of delay
        delay=delay-1; // Decrease delay on each iteration
    }
}
```

در ادامه آن را در پروتئوس پیاده‌سازی کرده که ۳ورودی اول پورت C به چراغ اول و ۳ ورودی دوم پورت C به چراغ دوم متصل شده و همچنین دو پتانسیومتر هم به ترتیب به دو ورودی اول پورت A متصل شده و یک پایه آن به زمین و پایه دیگر آن به AVCC متصل کرده که همزمان آن هم به Power متصل است. همچنین برای اینکه جریان را هم متوجه بشویم از ولت سنج استفاده کرده که لحظه ای جریان را مشاهده بکنیم. (بدلیل اینکه

واضح تر بشه و هم اینکه هم با LED و هم با چراغ های راهنمایی پورتئوس استفاده کرده باشیم هر دو را قرار دادم ولی کار اضافه ای است و میتوان به همان چراغ های راهنمایی بسنده کرد.)



در ادامه برای تست کردن آن میتوان پتانسیومتر اولی را روی ۱۰۰٪ تنظیم کرد که مقدار پتانسیومتر اول برابر ۲۴ می‌شود و اگر هم پتانسیومتر دومی را روی ۵۰٪ تنظیم بکنیم مقدار سبز بودن چراغ اول ۱۰ و مقدار سبز بودت چراغ دوم هم ۱۰ خواهد بود که نسبت هر دو برابر بوده است. و اگر هم پتانسیومتر دوم را روی ۱۰۰٪ تنظیم بکنیم مقدار سبز بودن چراغ اول ۱۵ و مقدار سبز بودن چراغ دوم ۵ خواهد بود و اگر به پایین نزدیک شویم یعنی روی ۰ ٪ تنظیم شود مقدار سبز بودن چراغ اول ۵ و مقدار سبز بودن چراغ دوم ۱۵ خواهد بود. دیگر مقادیر را باتوجه به مقدار تنظیم شده برای پتانسیومتر اول و همچنین پتانسیومتر دوم هم می‌توان بدست آورد که هر کدام خروجی متناسب با آن را روی چراغ‌های راهنمایی و رانندگی را می‌توان مشاهده کرد.