دانشگاه خوارزمی
دانشکده فنی و مهندسی
رشته مهندسی کامپیوتر

آزمایشگاه شماره ۱: آشنایی با AVR

نگارندگان: دانیال صابر، سعید وجدی
استاد درس: محمد لآلی

آبان ۱۴۰۱

چکیده

AVR که نوعی میکروکنترولر محسوب میشود یک شبه رایانه هوشمند در ابعادی کوچک است که کاربر با برنامه نویسی قادر به کنترل عملیات آن خواهد بود. در واقع می توان گفت میکروکنترلر AVR دارای تمام ویژگی های یک رایانه، با قابلیت پردازش و محاسبات، اما در ابعادی محدودتر است [۱]. هدف ما در این آزمایش پیاده‌سازی چندین برنامه با زبان C بر روی AVR است که در محیط Proteus شبیه‌سازی میشود. همچنین به بررسی مقاومت Pull up و انتقال اطلاعات به صورت موازی بین دو میکروکنترلر خواهیم پرداخت.

۱. مقدمه

میکروکنترلر AVR خانواده ای از میکروکنترلرها است که در سال ۱۹۹۶ توسط Alf-Egil Bogen و Vegard Wollan دو محقق و دانشجوی نروژی طراحی شد و در شرکت ATMEL ساخته شد. نام این میکروکنترلر از مخفف نام طراحان آن گرفته شده است. اولین میکروکنترلری که بر اساس طراحی AVR تولید شد مدل AT90S8515 بود، با این حال در سال ۱۹۹۷ میکروکنترلر AT90S1200 وارد بازار شد، تا این زمان میکروکنترلر های avr هنوز کاملا شناخته شده نبود [۲].
هدف اصلی آزمایش انجام شده در جلسه اول آشنایی با این میکروکنترولر و نحوه پیاده‌سازی ساده‌ترین برنامه بر روی AVR بود. در این آزمایش مشاهده کردیم که چگونه بایستی حین برنامه نویسی، از خروجی‌های AVR استفاده و آن را در Proteus شبیه‌سازی نمود.
در ادامه مشاهدات در آزمایشگاه، تمارینی از قبیل پیاده‌سازی برنامه چشمک‌زن، آشنایی با مدار Reset میکروکنترولر و مقاومت Pull up، و نحوه انتقال اطلاعات به صورت موازی بین دو میکروکنترلر انجام شد.

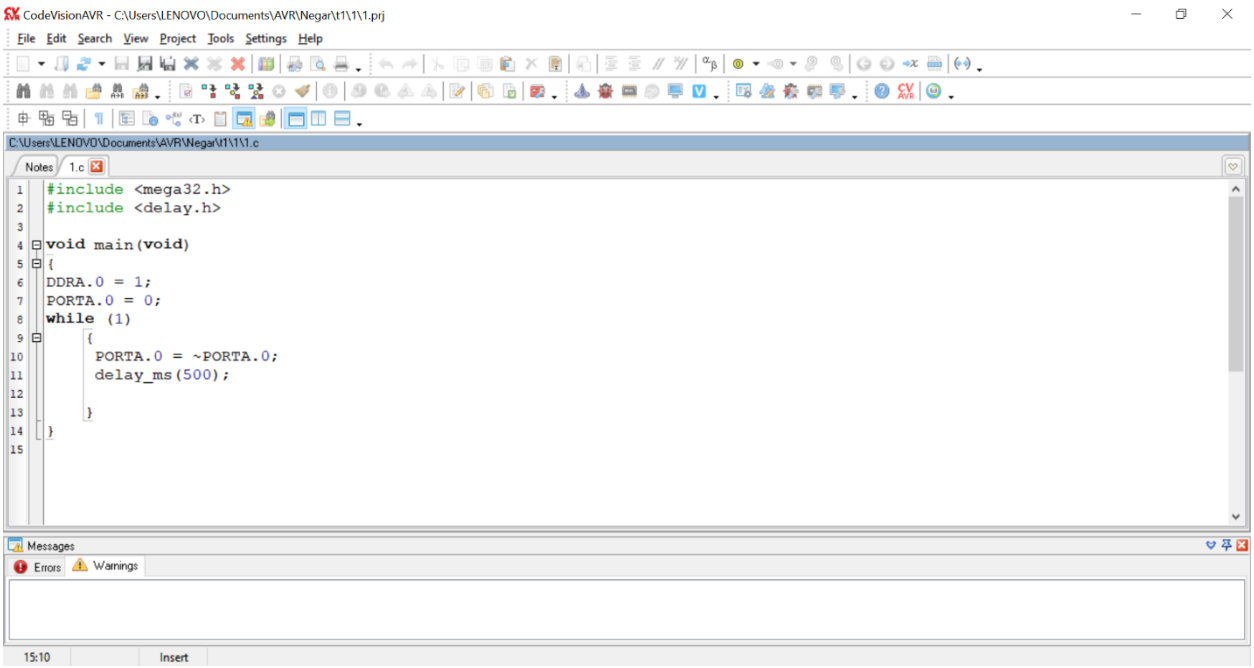
۲. روش‌ها و تجهیزات مورد استفاده

برای انجام تمارین، از زبان برنامه‌نویسی C، برنامه Codevision، و Proteus استفاده شده است.

۳. تمارین

۳/۱. برنامه چشمک‌زن

برنامه‌ی چشمک زن به زبان C نوشته شده است. کد و توضیحات آن در زیر آمده است:



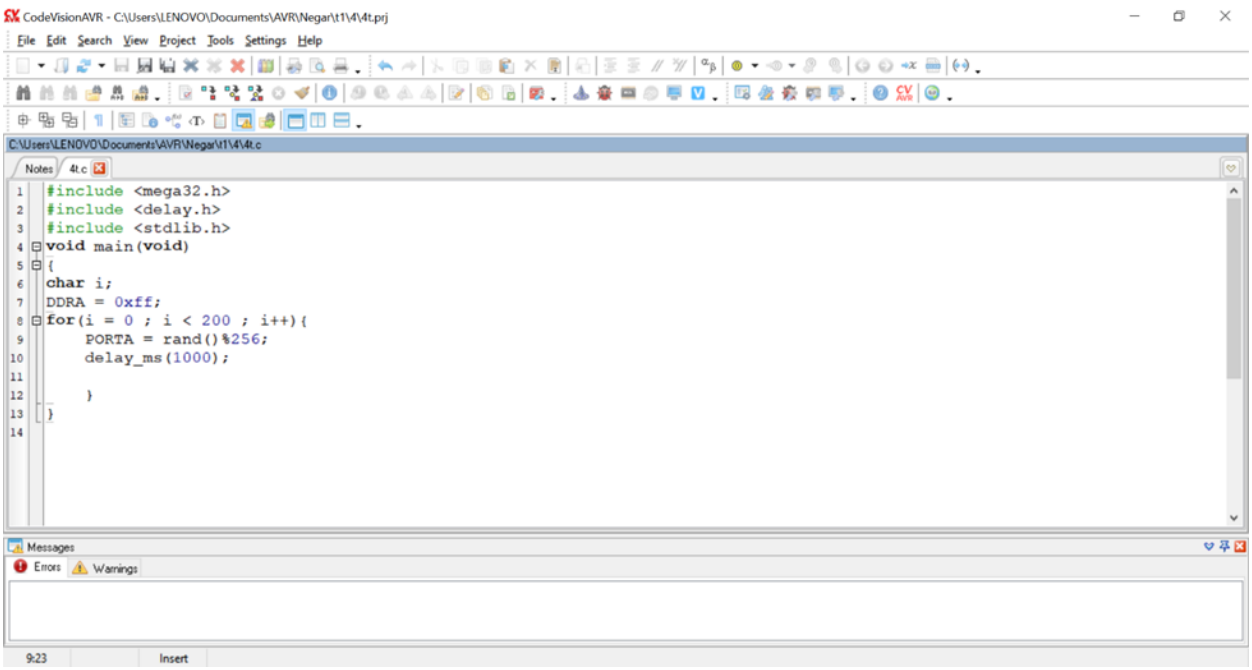
شکل ۱. برنامه چشمک‌زن

در شروع کار کتابخانه های میکروکنترلر و تاخیر زمانی را اضافه می کنیم.
همان طور که می دانیم اجرای هر برنامه ی C با تابع main شروع می شود. در این تابع، در رجیستر DDRA پین ۰ به عنوان خروجی تعریف شده و مقدار اولیه آن داخل رجیستر PORTA صفر ریخته شده است. در داخل حلقه while(1) (حلقه برنامه) دائما مقدار خروجی مکمل شده (یعنی خاموش و روشن می شود) و ۵۰۰ میلی ثانیه تاخیر ایجاد می شود. یعنی فرکانس چشمک زدن برابر با ۱ هرتز است. برای تغییر فرکانس چشمک زن، باید مقدار داخل پرانتز در دستور را تغییر داد .نتیجه شبیه سازی ضمیمه شده است.

۳/۲. مدار ریست میکروکنترولر AVR

پایه ۹ از میکروکنترلر ATmega32 به پایه ی ریست اختصاص دارد. پایه reset همان طور که از نامش پیداست برای راه اندازی مجدد میکرو به کار می رود. این پایه در صورتی که صفر منطقی شود میکرو reset می گردد. برای جلوگیری از افتادن نویز روی این پایه این پایه را با یک مقاومت در حدود 5k تا 10k به Vcc وصل می شود به اصطلاح pull up می گردد. در صورتی که بخواهیم میکرو reset شود توسط کلید موجود پایه میکرو به زمین متصل شده و میکرو reset می شود. و در

نهایت خازن به کار برده شده در مدار باید مقدار 10uF داشته باشد، برای reset شدن میکرو در لحظه ی قطع وصل شدن برق dc میکرو می باشد.



شکل ۲. برنامه Reset میکروکنترولر

۳٫۳. مقاومت Pull up

مقاومت نشان داده شده در شکل سوال، برای محدود کردن جریان led و نسوختن آن است. از آنجاکه led یک دیود نورانی است، ولتاژ وصل آن حدود ۰٫۷ ولت است. معمولا حداکثر جریان قابل تحمل led ۱۶ میلی آمپر است. در این صورت مقدار مقاومت به سادگی محاسبه می شود:

$$I_{LED} = \frac{5-0.7}{R} \rightarrow R_{min} = \frac{5-0.7}{Imax} = \frac{4.3V}{16mA} = 268.75^{\Omega} \rightarrow R > 270^{\Omega}$$

در نتیجه مقدار مقاومت باید حداقل ۲۷۰ اهم باشد. هر چقدر مقدار مقاومت افزایش یابد، جریان LED کاهش یافته و در نتیجه نور آن کم می شود. معمولا مقاومت ۳۳۰ اهم با LED سری می کنند.

۳٫۴. انتقال داده بین AVR ها

در این سوال هدف انتقال اطلاعات به صورت موازی بین دو میکروکنترلر، تولید عدد تصادفی در میکروکنترلر سمت راست و ذخیرهی کاراکتر دریافتی در میکروکنترلر سمت چپ می باشد.
کد فرستنده در حالتی که کاراکتر ارسالی در فرستنده مشاهده نمی شود به صورت زیر است:
در ابتدا کتابخانه های stdlib , delay , mega32 را فراخوانی می کنیم. با کاربرد دو کتابخانه ی اول در سوال ۱ آشنا شدیم. کتابخانه ی stdlib برای استفاده از تابع ()rand فراخوانی شده که اعداد تصادفی را می سازد.

در تابع main ، یک کاراکتر به نام i تعریف شده که متغیر شمارش حلقه ی for است. سپس پورت A با دستور DDRA = 0xff به عنوان خروجی تعریف شده است. داخل حلقه ی for که ۲۰۰ بار اجرا می شود ، باقیمانده تقسیم عددی تصادفی بر ۲۵۶ داخل PORTA قرار گرفته و ۱ ثانیه تاخیر ایجاد می شود.

لازم بذکر است در این برنامه ، بعد از پایان ارسال داده ها، کار میکروکنترلر تمام می شود. در صورت نیاز می توان بعد از حلقه ی for یک حلقه ی while(1) قرار داد.

کد گیرنده به صورت زیر است:

در ابتدا علاوه بر دو کتابهانه mega32 و delay ، کتابخانه ی alcd.h فراخوانی شده که دستورات کار با LCD را در داخل خود دارد.

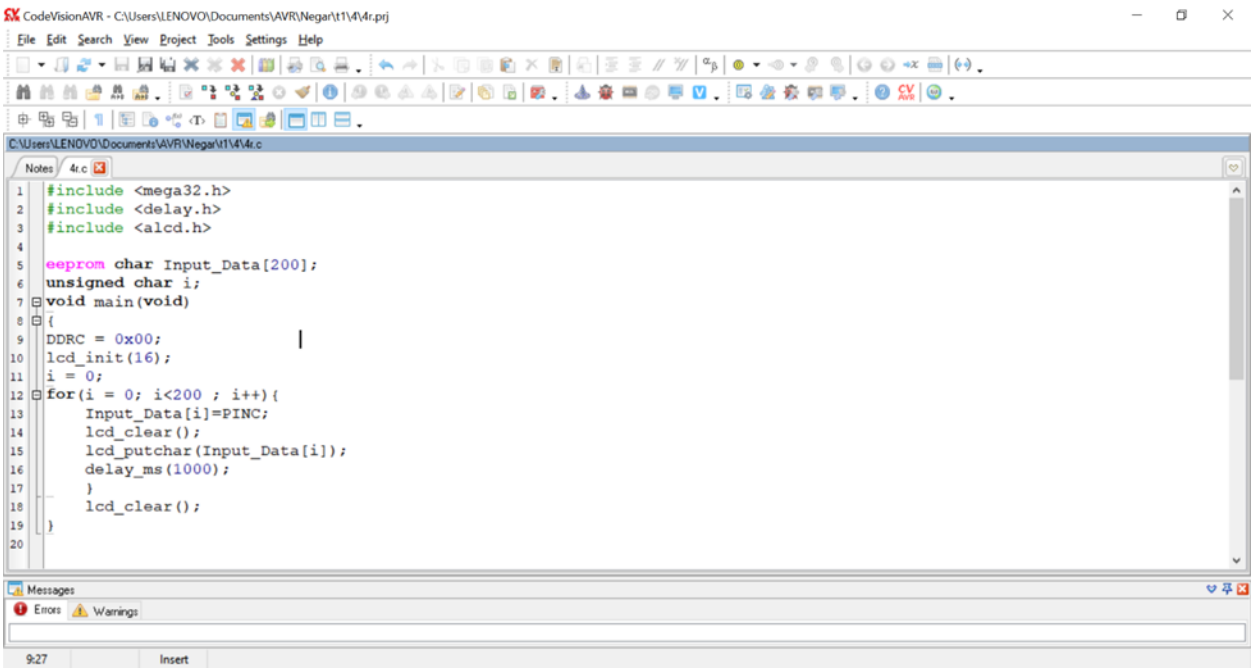
در ادامه آرایه ای از نوع char (داده ی ۸ بیتی) به نام Input_Data با ۲۰۰ عنصر در حافظه eeprom تعریف شده است. متغیر i که متغیر حلقه است.

داخل تابع main ابتدا پورت C با دستور PORTC = 0x00 به عنوان ورودی تعیین شده و دستور lcd_init(16) نمایشگر lcd را با ۱۶ ستون راه اندازی می کند.

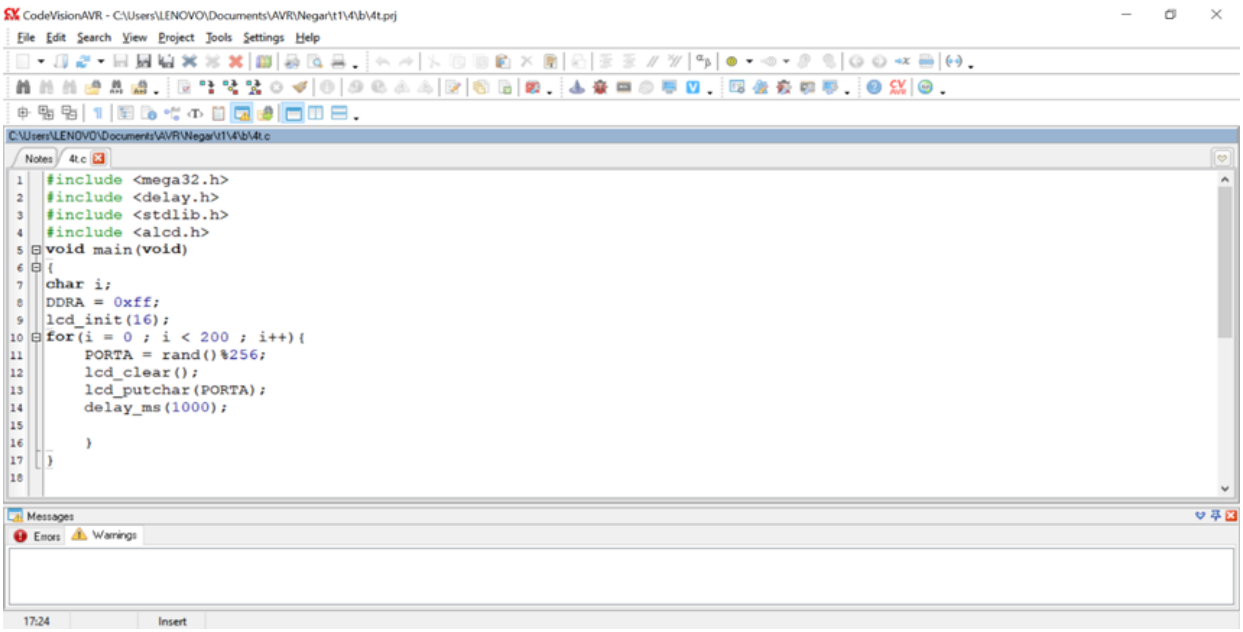
داخل حلقه ی for که ۲۰۰ بار اجرا می شود، مقدار ورودی پورت C از رجیستر PINC خوانده شده و داخل عنصر شماره i از آرایه Input_Data قرار می گیرد. سپس با دستور lcd_clear() نمایشگر پاک شده و با دستور lcd_putchar(Input_Data[i]) کاراکتر دریافتی روی lcd چاپ می شود. در نهایت ۱ ثانیه صبر کرده و به ابتدا ی حلقه بازگشته و کاراکتر بعدی را دریافت می‌کند.

نهایتا بعد از دریافت همه ی کاراکترها، برنامه از حلقه ی for خارج شده و lcd با دستور lcd_clear() پاک می شود.

کد فرستنده در حالتی که کاراکتر ارسالی در سمت فرستنده چاپ شود، در زیر مشاهده می شود:



شکل ۳. کد فرستنده



شکل ٤. کد گیرنده

همانطور که می بینیم ساختمان اصلی کد مانند حالت قبل است. فقط کتابخانه ی alcd هم اضافه شده و داخل حلقه ی for بعد از تولید کاراکتر تصادفی و قرار دادن آن در خروجی، lcd با دستور lcd_clear() پاک شده و با دستور lcd_putchar(PORTA) کاراکتر ارسالی روی lcd چاپ می شود.

کد گیرنده مانند حالت قبل است و تفاوتی ندارد و ویدئوی شبیه سازی ضمیمه شده است.

لازم به توضیح است که دلیل تاخیر و فاصله ی یک کاراکتری بین نمایش کاراکتر روی LCD فرستنده و گیرنده، احتمالا خواندن و نوشتن گیرنده از eeprom است که حافظه ی کندی است.

۴. نتیجه‌گیری

در این آزمایش با نحوه کار با میکروکنترولر AVR آشنا شدیم. در تمرین اول، نحوه پیاده‌سازی برنامه چشمک‌زن با زبان C را دیدیم که چگونه بایستی از پورت‌های AVR استفاده نمود. در تمرین دوم درباره نقش Reset در AVR که برای راه اندازی مجدد میکرو به کار می‌رود تحقیق کردیم. این پایه در صورتی که صفر منطقی شود میکرو reset می گردد. سپس درباره مقاومت Pull up که برای جلوگیری از سوختن LED ها استفاده میشود بحث شد. در نهایت، دیدیم که چگونه انتقال اطلاعات به صورت موازی بین دو میکروکنترلر، تولید عدد تصادفی در میکروکنترلر سمت راست و ذخیرهی کاراکتر دریافتی در میکروکنترلر سمت چپ صورت میگیرد.

۵. مراجع

[1] -https://namatek.com/%D9%85%DB%8C%DA%A9%D8%B1%D9%88%DA%A9%D9%86%D8%AA%D8%B1%D9%84%D8%B1-avr-%DA%86%DB%8C%D8%B3%D8%AA/

[2] -https://paytakhtfanavari.com/microcontroller-avr/