



دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

دستور کار آزمایشگاه ریزپردازنده و زبان اسمنبلی

(بهار ۱۴۰۰)

مولفان:

بهداد منصوری

علیرضا صالحی

ارشیا رحیمی

نرگس سدیفی

فهرست مطالب

۱	قوانين آزمایشگاه ریزپردازنده
۲۹	آزمایش ۱: آشنایی با برد Arduino MEGA ۲۵۶۰ و برنامه‌های مورد نیاز
۱۲	آزمایش ۲: کیبورد ورودی و ارتباطات سریال
۱۷	آزمایش ۳: مانیتور خروجی
۲۰	آزمایش ۴: راه اندازی سروو موتور و ورودی آنالوگ
۲۴	آزمایش ۵: راه اندازی رله با Arduino MEGA2560
۳۰	آزمایش ۶: نیم‌پروژه
۳۵	آزمایش ۷: ماشین لباسشویی ساده
۴۱	آزمایش ۸: اتاق تحت کنترل
۴۶	آزمایش ۹: موسیقی و header
۴۹	آزمایش ۱۰: پروژه نهایی
	آزمایش ۱ اسمنبلی
	آزمایش ۲ اسمنبلی
	پروژه های اسمنبلی

قوانين آزمایشگاه ریزپردازند

به منظور افزایش کارایی درس آزمایشگاه ریزپردازند، رعایت عدالت میان همه گروههای آزمایشگاه و آموزش حداکثری مطالب درس به صورت عملی، مدرسین و دانشجویان ملزم به رعایت نکات و قوانین زیر هستند:

۱. مدرسین و دانشجویان موظفند راس ساعت در کلاس آنلاین حاضر شوند.
۲. در تمام مدت زمان کلاس دانشجویان موظفند آنلاین بوده و بنا به صلاح‌حدید مدرسین وب‌کم و میکروفون خود را فعال کرده و همچنین صفحه دسکتاپ خود را به اشتراک بگذارند.
۳. غیبت در هیچ یک از جلسات کلاس‌های آنلاین به هیچ عنوان مجاز نیست.
۴. اگر دانشجویانی نتوانند در طی یک جلسه آزمایش مورد نظر را تمام کنند، در طول هفته موظفند تا آن آزمایش را تکمیل کرده و پیش از آغاز جلسه بعد نتیجه را به مدرس خود تحويل دهند. حداکثر تعداد جلسات که به درازا کشیده می‌شود ۱ جلسه در طول نیمسال است. دانشجو برای تکمیل آزمایش نمی‌تواند به گروههای دیگر ملحق شود.
۵. آزمایش‌ها در گروههای تک نفره انجام می‌شوند و میزان فعالیت افراد تعیین کننده نمره نهایی آن‌ها خواهد بود.
۶. هر آزمایش شامل یک پیش‌گزارش است. این پیش‌گزارش باید به صورت دست‌نویس و پیش از شروع آزمایش‌ها از طریق که مدرس آزمایشگاه اعلام کرده است به ایشان تحويل داده شود. پیش‌گزارش مناسب برای هر آزمایش در دستور کار آمده است.
۷. ارائه گزارش کار آزمایش‌ها الزاماً نیست و تحويل خروجی صحیح به مدرس کفایت می‌کند.

نکات امنیتی:

۱. پیش از انجام هر آزمایش، مبحث تئوری آن آزمایش باید به طور کامل مطالعه شود و پرسش‌هایی که در بخش "آنچه در پیش‌گزارش باید نوشته شود" آمده است باید با دقت پاسخ داده شود؛ چراکه در هنگام انجام آزمایش، وقت کافی برای توضیح و یادگیری قسمت تئوری وجود ندارد.
۲. در انجام هر آزمایش لازم است که مطالب ذکر شده در آزمایش‌های قبل را به خاطر داشته باشید. زیرا به دلیل کمبود وقت، مطالب تکراری توضیح داده نمی‌شود. پس دانشجویان باید علاوه بر گزارش کاری که به تدریس‌یار خود تحويل می‌دهند (چه به صورت شفاهی و چه به صورت کتبی)، برای خود نیز یادداشت بردارند تا بعد از اتمام کلاس‌ها دچار مشکل نشونند.
۳. هیچگاه دیودهای نورانی (یا اصولاً هر قطعه‌ای که در آن دیود نورانی به کار رفته مثل هفت قسمتی‌ها و ماتریس‌های LED) را مستقیماً به خروجی برد یا منبع تغذیه وصل نکنید، بلکه آن را با یک مقاومت بین ۱۰۰ تا ۳۳۰ اهمی سری نموده، و سپس وصل کنید.

۲۵۶. آشنایی با برد Arduino MEGA

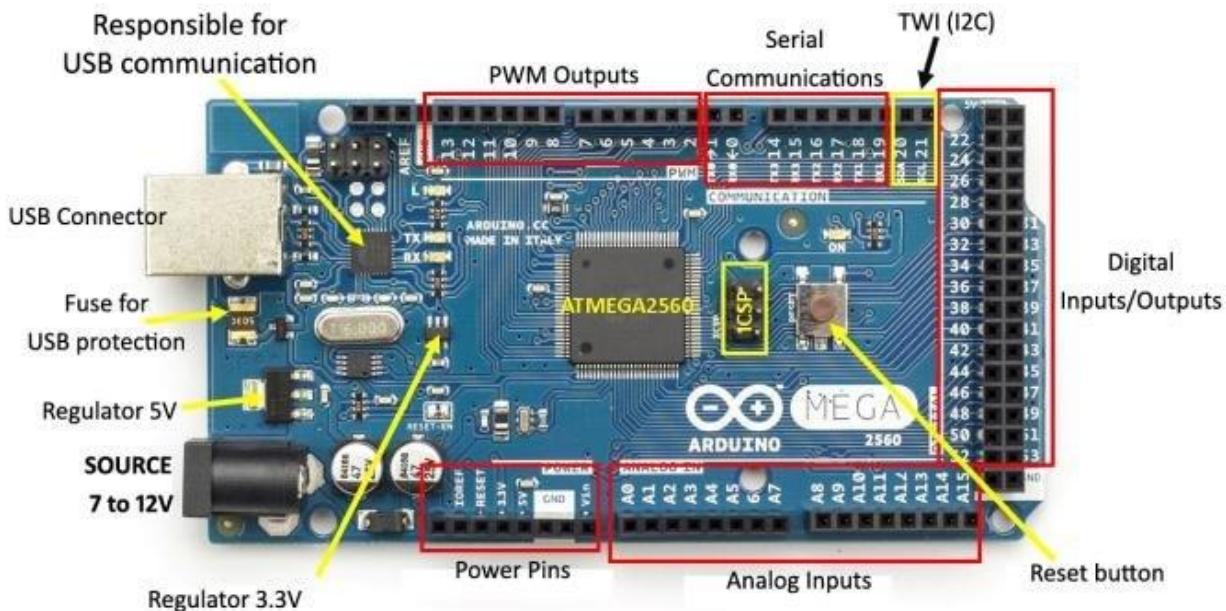
برد ۲۵۶ Arduino MEGA یک میکروکنترلر بر پایه ATmega ۲۵۶ از شرکت Atmel است. این برد دارای ۵۴ پین دیجیتال ورودی/خروجی (که ۱۵ پین می‌تواند به عنوان خروجی PWM استفاده شود)، ۱۶ ورودی آنالوگ، ۴ پورت UARTs (پورت‌های سریال ساخت افزاری)، یک کلاک ۱۶ مگاهرتزی، یک کابل ارتقابی USB، یک پاور جک، یک ICSP header، و یک دکمه ریست است. این برد عملایی چیزهایی که برای کار با میکروکنترلر نیاز است را شامل می‌شود. فقط کافیست با USB به یک کامپیوتر متصل شود یا با آداپتور AC به DC برق بگیرد و یا از باتری استفاده کند تا روشن شود. مگا ۲۵۶ می‌تواند با اکثر شیلد‌های طراحی شده برای Uno و همچنین برد‌های قدیمی Duemilanove or Diecimila کار کند.

۲۵۶ Arduino MEGA همانند دیگر برد‌های Arduino با برنامه [Arduino Software IDE](#) برنامه‌ریزی می‌شود که نحوه نصب و اتصال برد به آن در پیوست آورده شده است.

مشخصات:

ATmega2560

۵ ولت	میکروکنترلر:
۷ تا ۱۲ ولت	ولتاژ عملیاتی:
۲۰-۶ ولت	ولتاژ ورودی (پیشنهادی):
۱۵ تای آن خروجی PWM تولید می‌کنند).	دامنه مجاز ولتاژ ورودی:
۱۶ عدد	پین‌های دیجیتال ورودی/خروجی:
۰.۵ میلی آمپر	پین‌های ورودی آنالوگ:
۰.۲ میلی آمپر	جریان DC هر پین ۳، ۳ ورودی/خروجی:
۸ کیلوبایت	جریان DC هر پین ۵ ولت:
۱۶ مگاهرتز	حافظه فلاش:
	SRAM
	سرعت ساعت:



برد آردوینو Mega2560

نیشانگر LED های

نیشانگر LED تغذیه: هنگام اتصال تغذیه به برد آردوینو روشن می شود. اگر LED روشن نشد، در بخشی از اتصال تغذیه مشکلی وجود دارد.

LED های TX و RX: بر روی برد دو بخش به نام های TX (ارسال) و RX (دریافت) وجود دارد. یکی در بخش پایه های ۱۰ و ۱۴ تا ۱۹ که برای ارتباط سریال هستند و دومی LED های TX و RX. چراغ مربوط به TX هنگام ارسال داده سریال مناسب با میزان سرعت چشمک می زند و چراغ مربوط به RX نیز هنگام دریافت داده چشمک زدن به baud rate برد بستگی دارد.

تغذیه:

ولتاژ مورد نیاز Arduino MEGA 2560 می تواند از طریق اتصال USB و یا یک منبع تغذیه خارجی تامین شود. هنگامی که اتصال برقرار شد، منبع تغذیه به صورت خودکار انتخاب می شود.

منبع تغذیه خارجی غیر از USB می تواند آداپتور AC به DC یا باتری باشد. آداپتور (با سوکت های center-positive به قطر ۲,۱ میلی متر) می تواند به پاورجک موجود بر روی برد متصل شود و سیم های باتری می توانند مستقیماً وارد پین های GND و Vin شوند.

برد می تواند با منبع تغذیه خارجی ۶ تا ۲۰ ولت کار کند. اگر ولتاژ منبع تغذیه پایین تر از ۷ ولت باشد روی ولتاژ پین ها اثر خواهد گذاشت و ممکن است ولتاژ خروجی آنها کمتر از ۵ ولت شود و حتی نوساناتی را به وجود آورد. ولتاژ بیش از ۱۲ ولت نیز می تواند موجب افزایش دمای رگولاتور و در نتیجه آسیب به برد گردد. ولتاژ پیشنهادی مناسب، بین ۷ تا ۱۲ ولت است.

پین های مربوط به منبع تغذیه به شرح زیر است:

- V_{IN} : پین ورودی ولتاژ آردوینو است که به هنگام استفاده از منبع تغذیه خارجی (به جای منبع تغذیه تنظیم شده یا اتصال USB با ۵ ولت) از آن استفاده می شود و چنانچه برد از طریق پاورجک به منبع تغذیه وصل شده باشد، می توانید از طریق این پین (به عنوان خروجی) به ولتاژ منبع تغذیه دسترسی داشته باشید.

- ۵V: این پین یک ولتاژ تنظیم شده ۵ ولت را از طریق رگولاتور موجود بر روی برد فراهم می کند. برد می تواند از طریق پاورجک ۷-۱۲ ولت)، پورت USB (به اندازه ۵ ولت) و یا پین V_{IN} برد (۷-۱۲ ولت)، تغذیه گردد. ولتاژ پین های ۵ ولت، ۳،۳ ولت از رگولاتور عبور می نماید و استفاده از ولتاژ این پین ها ممکن است باعث صدمه دیدن برد شود، از همین رو، استفاده از این پین ها توصیه نمی شود.

- ۳.3V: یک ولتاژ ۳،۳ ولتی، به وسیله ی رگولاتور روی برد فراهم می گردد که حداقل جریان آن ۵۰ میلی آمپر است.

- GND: پین هایی که با اتصال به زمین، ولتاژ صفر را فراهم می کنند.

- IOREF: این پین میزان ولتاژ مرجعی که میکروکنترلر با آن کار می کند را مشخص می نماید. این پین اجازه می دهد یک شیلد را با پیکربندی مناسب، جهت تطبیق با ولتاژی که توسط برد فراهم شده است، به برد متصل کنید. یک شیلد که به درستی تنظیم شده باشد، می تواند مقدار ولتاژ را از پین IOREF خوانده، منبع تغذیه مناسب خود را انتخاب نماید و یا این که مبدل های ولتاژ را برای کار کردن با ولتاژ های ۵ یا ۳،۳ ولت، بر روی خروجی ها فعال نماید. این قابلیت، به شیلد ها امکان می دهد تا با برد ۳،۳ ولت همچون DUE و برد های AVR-based که با ولتاژ ۵ ولت کار می کنند، خود را تطبیق دهند.

Arduino

آردوینو یک بستر متن باز، برای توسعه سیستم های نهفته می باشد که کار توسعه سیستم های سخت افزاری را ساده تر می کند. این بستر یک فریم ورک توسعه متن باز به زبان C/C++ برای AVR و ... فراهم میکند. این فریم ورک API یکسانی را به ازای میکروکنترلهای متفاوت فراهم میکند و مدیریت سطح پایین سخت افزار میکروکنترلر را انجام میدهد. از این رو برنامه ای که بر بستر آردوینو نوشته می شود قابلیت آن را دارد که بر میکروکنترلهای مختلف با سخت های افزار های متفاوت اجرا شود.

در پروژه آردوینویی توان زبانهای C, C++ و Assembly را به کاربرد. برای نمونه یک روال را به زبان اسمبلی نوشته و در کد C/C++ آن را فراخوانی کرد. از این رو می توان مدیریت سطح پایین سخت افزار را به جای قابلیت های عمومی فریم ورک آردوینو، به طور ویژه برای پروژه خود پیاده سازی کرد.

همچنین می توان از قابلیت های برنامه نویسی شی گرا در C++ مانند کلاس ها، ارث بری، اینترفیس و... بهره مند شد. البته باید توجه داشت که در برنامه نویسی شی گرا در میکروکنترلهای دلیل منابع سخت افزاری محدود، همه قابلیت های زبان C++ پیاده سازی نشده است.

- برای راهنمایی در بخش های مختلف گزارش کارکدهای نمونه ای در گیت هاب آزمایشگاه گذاشته می شود که به آدرس زیر می باشد.

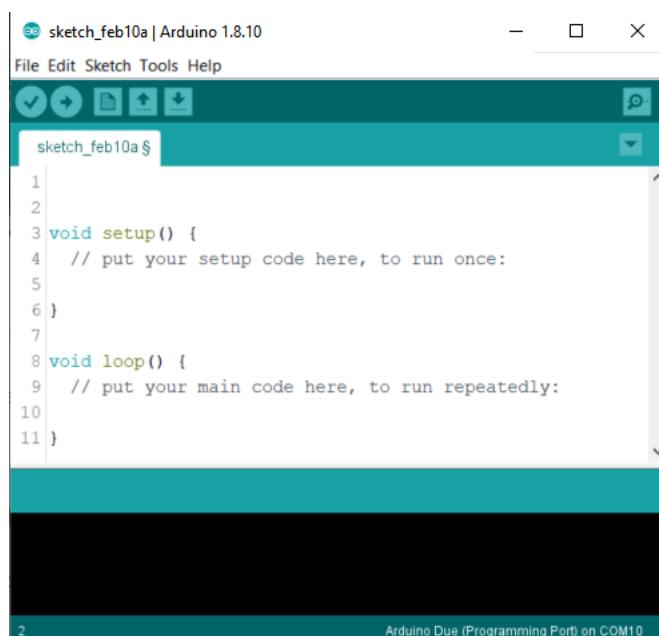
<https://github.com/MicroprocessorAUT/Lab/tree/main/Samples>

- پروژه های نمونه ای برای نشان داده شیوه استفاده از کلاس های C++ و ساختاریندی مناسب کد در آدرس بالا آورده شده است. از آنجا که رعایت این ساختاریندی در تحویل کد های آزمایشگاه لازم می باشد، این پروژه های نمونه را بررسی کنید.

پلتفرم آردوینو یک IDE نیز برای برنامه نویسی بردهای آردوینو و همچنین ارتباط سریال با برد به نام Arduino IDE فراهم میکند. این برنامه متن باز و رایگان است. می توانید نسخه دسکتاپ آن را نصب نمایید یا از نسخه برخط آن در آدرس زیر برای نوشتن برنامه و آپلود آن بر روی برد استفاده کنید.

<https://create.arduino.cc/>

شکل زیر محیط برنامه Arduino IDE را نشان می دهد.



محیط برنامه Arduino IDE

البته این IDE برعی توانایی های محیط توسعه مدرن را ندارد، اما خوشبختانه تنها محیط توسعه آردوینو موجود نیست و برای نمونه که بر روی ویرایشگر هایی مانند [Atom](#) و [VS Code](#) اجرا می شود، قابلیت های بیشتری را ارائه میکند.

همچنین می توان از محیط [Eclipse C/C++](#) برای توسعه پروژه های آردوینو بهره برد.

عملکرد منوهای موجود در Arduino IDE

-	Verify: برای بررسی خطاهای برنامه‌ی نوشته شده از این گزینه استفاده می‌کنیم.
-	Upload: برای آپلود کردن برنامه‌ی نوشته شده روی برد از این گزینه استفاده می‌کنیم. (بروگرم کردن میکروکنترلر)
-	New: کلید میان بر برای ایجاد یک پروژه جدید.
-	Open: کلید میان بر برای باز کردن نمونه پروژه‌های موجود در نرمافزار.
-	Save: ذخیره‌ی پروژه ایجاد شده.
-	Serial Monitor: ترمینال سریال برای دریافت دیتای پورت سریال از برد و ارسال اطلاعات به آن.

حالا با یک کلیک ساده روی منوی Upload شروع به آپلود برنامه روی برد آردوینو خواهید کرد. چند ثانیه صبر نمایید، در زمان آپلود دو عدد Led با نام‌های Rx و Tx روی برد چشمک خواهند زد. در صورتی که آپلود برنامه با موفقیت انجام شود، در نوار وضعیت، پیغام "Done Uploading" را خواهید دید.

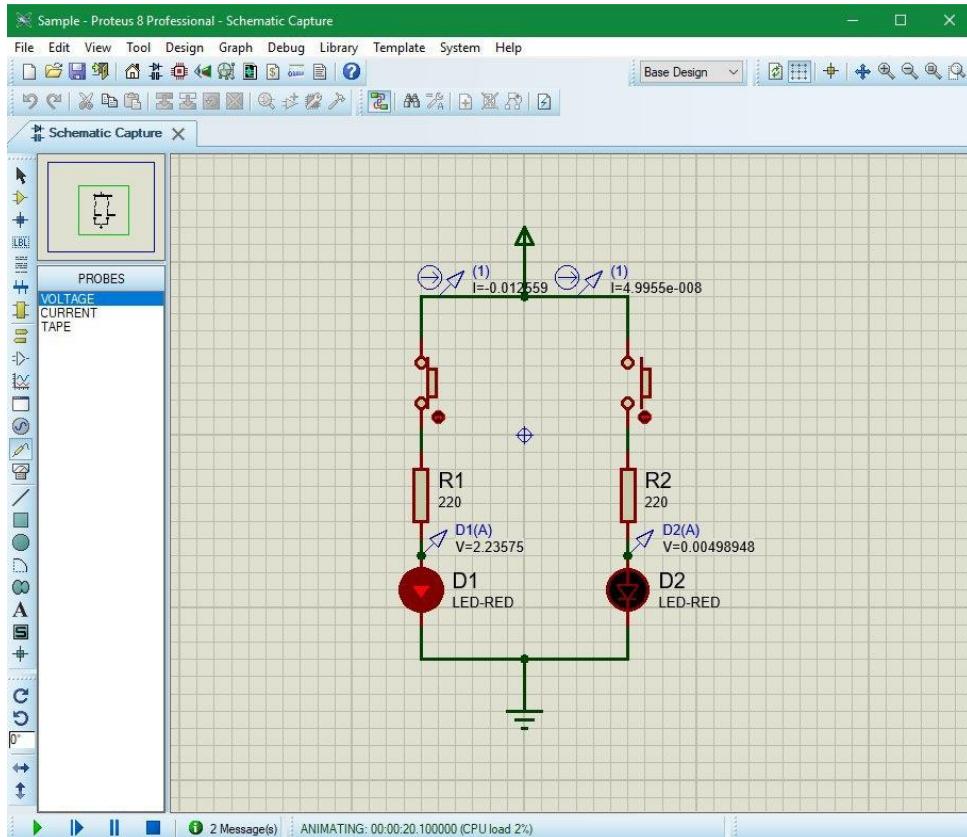
برنامه‌نویسی در آردوینو ۳ بخش کلی دارد:

۱. پیش‌پردازش‌ها: در بخش پیش‌پردازش، کتابخانه‌ها و یا متغیرهایی که محلی نیستند تعریف و فراخوانی می‌شوند.
۲. حلقه `setup`: در این بخش کارهایی که باید یک بار انجام شوند را تعریف می‌کنیم. به طور مثال، تنها کافی است یک بار تعریف کنیم که یک پین دیجیتال ورودی باشد یا خروجی و یا این که به برد اطلاع دهیم که می‌خواهیم از پورت سریال استفاده کنیم.
۳. حلقه `loop`: حلقه `loop` یک حلقه بی‌نهایت است و کارهایی که باید به طور متناوب انجام شوند در این حلقه نوشته می‌شود. به طور مثال، اگر بخواهیم یک سنسور دما را به برد وصل کنیم و برد دمای محیط را بر روی نمایش گر نشان دهد، عمل خواندن دما از سنسور و نوشتمن اطلاعات در صفحه نمایش را باید در این حلقه بنویسیم.

برای گرفتن فایل hex از Arduino IDE که به Proteus داده شود، بعد از کامپایل شدن کد، با انتخاب گزینه Export Compiled Sketch Folder Binary از تب Sketch می‌توانید فایل بازیزی کامپایل شده برای پروتئوس را بسازید که در Sketch Folder ذخیره می‌شود و می‌توانید با انتخاب Show Sketch Folder در زیر این گزینه، به آن دسترسی پیدا کنید.

نرم افزار شبیه سازی Proteus

این ترم به دلیل برگزاری مجازی آزمایشگاه، آزمایش ها در بستر شبیه سازی Proteus انجام خواهد شد. این نرم افزار مانند نرم افزار ORCAD که در آزمایشگاه مدارهای الکتریک با آن آشنا شدید، امکان شبیه سازی مدارهای الکتریک را ارائه می کند. افزون بر آن، شبیه سازی کارکرد خانواده هایی از میکروکنترلرها در مدار را فراهم می کند. از این رو، می توان از این نرم افزار برای شبیه سازی آزمایش های درس ریز پردازند بهره برد.



محیط برنامه Proteus

چگونگی انجام شبیه سازی آزمایش ها در محیط Proteus

نخست مدار آزمایش را در نرم افزار Proteus طراحی کنید، سپس در محیط توسعه آردوینو در تابع `setup()` (پیکربندی ورودی/خروجی پایه ها را انجام دهید و در تابع `loop()` منطق کنترلی برنامه را پیاده سازی کنید.

پس از آن، برنامه را برای ATmega 2560 کامپایل کنید. بدین منظور، ابتدا برد را از طریق Tools -> Board -> Arduino Mega انتخاب کنید و سپس گزینه Verify را بزنید.

حال بر روی برد در Proteus کلیک راست کرده و گزینه Edit Properties را انتخاب کرده و سپس در بخش Program File آدرس فایل هگز کامپایل شده را قرار دهید و شبیه سازی را اجرا کنید.

همچنان راه دیگری نیز برای دریافت فایل باینری در محیط Arduino IDE وجود دارد، پس از کامپایل شدن برنامه، گزینه Export Compiled Binary در منوی Sketch را انتخاب کنید. این کار، فایل HEX برنامه را در پوشه Sketch می ریزد و می توان با Show Sketch Folder گزینه Show Sketch Folder در همان منو به آن دست یافت.

آزمایش ۱: کار با پایه های ورودی/خروجی (PIO) و وقفه ورودی (Input Interrupt)

هدف آزمایش:

- آشنایی واحد PIO
- آشنایی با روش های سرکشی (Interrupt-Driven) و وقفه محور (Polling) برای مدیریت واحد های جانبی
- مقایسه دو روش سرکشی و وقفه محور

قطعات آزمایش:

- برد ۲۵۶۰ Arduino MEGA
- دیود نورانی LED
- کلید
- مقاومت $\Omega 220$
- مقاومت $K\Omega 10$

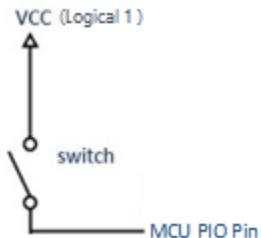
آنچه باید در پیش گزارش نوشته شود:

- با توجه به درس سیستم عامل، تفاوت روش های سرکشی و وقفه محور را بیان کنید.
- به پرسش هایی که در مقدمه آزمایش آورده شده است پاسخ دهید.

مقدمه:

مدارهای Pull-up و Pull-down

مدار زیر را در نظر بگیرید:



فرض کنید میخواهیم از این مدار برای دریافت اینکه چه زمانی کلید (Switch) بسته شده است استفاده کنیم. برای این کار سطح ولتاژ منطقی (۱ یا ۰) بر روی پایه میکرو (MCU PIO Pin) را پیوسته بررسی میکنیم و زمانی که برابر با ۱ شد را زمان بسته شدن کلید در نظر میگیریم. به عبارتی دیگر، زمانی که کاربر دکمه مربوط را فشار داده است.

پرسش: چرا این روش برای فهمیدن اینکه چه زمانی کلید بسته شده درست نیست؟ در این مدار پایه میکرو در چه حالتی می باشد؟

برای اینکه مشکل روش بالا بر طرف گردد می توان کلید را در مدار Pull-up یا Pull-down قرار داد.



بديهي است تفاوت دو مدار فوق در سطح ولتاژ پایه ميكروكنترلر در دو حالت فشرده شده يا آزاد ميباشد. دو مدار فوق را ببررسی كرده و از آنها در طراحی مدار خود بهره ببريد.

پرسش: درباره چگونگی کارکرد مدار های بالا توضیح دهید. به چه دلیل نیاز به مقاومت (Pull-up/pull-Down) داریم؟

برای نوشتن برنامه این آزمایش می باشد از دستورات `pinMode`, `digitalRead()`, `digitalWrite()`, `delay()` استفاده کنید.

وقفه:

وقفه پاسخی است که پردازنده به هنگام رخدان یک اتفاق (Event) می دهد. اين پاسخ به اين صورت است که پردازنده اجرای کنونی خود را متوقف کرده و روال سرويس وقفه (Interrupt Service Routine) متناظر با آن رخداد را اجرا خواهد کرد. پس از به پایان رسیدن سرويس وقفه پردازنده اجرای متوقف شده خود را دنبال خواهد کرد. واحد مدیریت وقفه مسئولیت اجرای این روند را برعهده دارد. به این صورت که یک اتفاق رخ می دهد در صورت لزوم روال سرويس وقفه متناظر با آن را اجرا خواهد کرد.

واحد های گوناگونی می توانند تنظیم شوند تا رخدان یک اتفاق مشخص را اعلام کنند (Assertion). یکی از این واحد ها، GPIO می باشد که می تواند حالت های مختلف سطح ولتاژ منطقی یک پایه ورودی را به عنوان اتفاق دلخواه در نظر گرفته و رخدان آن را به واحد مدیریت وقفه اعلام کند. از این روی توان این واحد را به گونه ای پیکربندی کرد که فشرده شدن کلید را اعلام کند.

سپس هر بار که دکمه فشار داده می شود، روال سرويس وقفه متناظر با آن اجرا خواهد شد. می توان این روال یا تابع را به گونه ای برنامه ریزی کرد که پاسخ مناسب به فشرده شدن کلید داده شود. در این صورت رخدان اتفاق مورد نظر یعنی فشرده شدن کلید در شرایط گوناگون مدیریت پذیر خواهد بود.

پرسش: آیا رخدان یک اتفاق در صورت اعلام شدن (Assertion) لزوما منجر به اجرای روال سرويس وقفه متناظر با آن می شود؟

همه یا برعی از پایه های یک واحد GPIO برای ثبت رخداد های ورودی در نظر گرفته شده است. شمار این پایه ها بسته به مدل ميكروكنترلر متفاوت است.

پرسش: پایه های وقفه در برد ATmega 2560 و شیوه پیاده سازی وقفه ورودی را بدست آورید.

دستوری که برای فعال سازی مدیریت وقفه روی پایه مدد نظر در آردوینو وجود دارد (`attachInterrupt()`) میباشد. برای اطلاعات بیشتر در این باره لینک زیر از مستندات آردوینو را بررسی کنید:

<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/?setlang=it>

پرسش: اگر بخواهیم در زمان تغییر مقدار پایه، وقفه فعال شود از چه mode ای درون تابع `attachInterrupt` استفاده می شود؟

پرسش: انواع اتفاق های ورودی را که واحد GPIO در برد آردوینو ATmega ۲۵۶۰ بتواند رخداد آن ها را بفهمد و اعلام کند بنویسید.

شرح آزمایش:

این آزمایش دربردارنده چندین LED (دلخواه اما بیشتر از ۵ عدد) و سه دکمه می باشد. که مدار آن بر روی برد در شکل ۱-۱ و مدار شماتیک آن در شکل ۲-۱ نمایش داده شده است. در ابتدا LED ها خاموش می باشند. با هر بار فشردن دکمه ی یک، LED ها از سمت چپ یک روشن می شوند، با هر بار فشردن دکمه ی دو، LED ها بصورت همزمان به تعداد کاراکتر های نام شما شروع به چشمک زدن می کنند (فراخوان `strlen` باید درون کد شما قابل مشاهده باشد) و پس از آن در حالت تمام روشن قرار می گیرند. و با فشردن دکمه سوم همه LED ها خاموش می شوند.

۱. برنامه آزمایش را به روش سرکشی بنویسید و پس از آن که از درستی کارکرد مدار و برنامه خود مطمئن شدید گام دوم را انجام دهید.

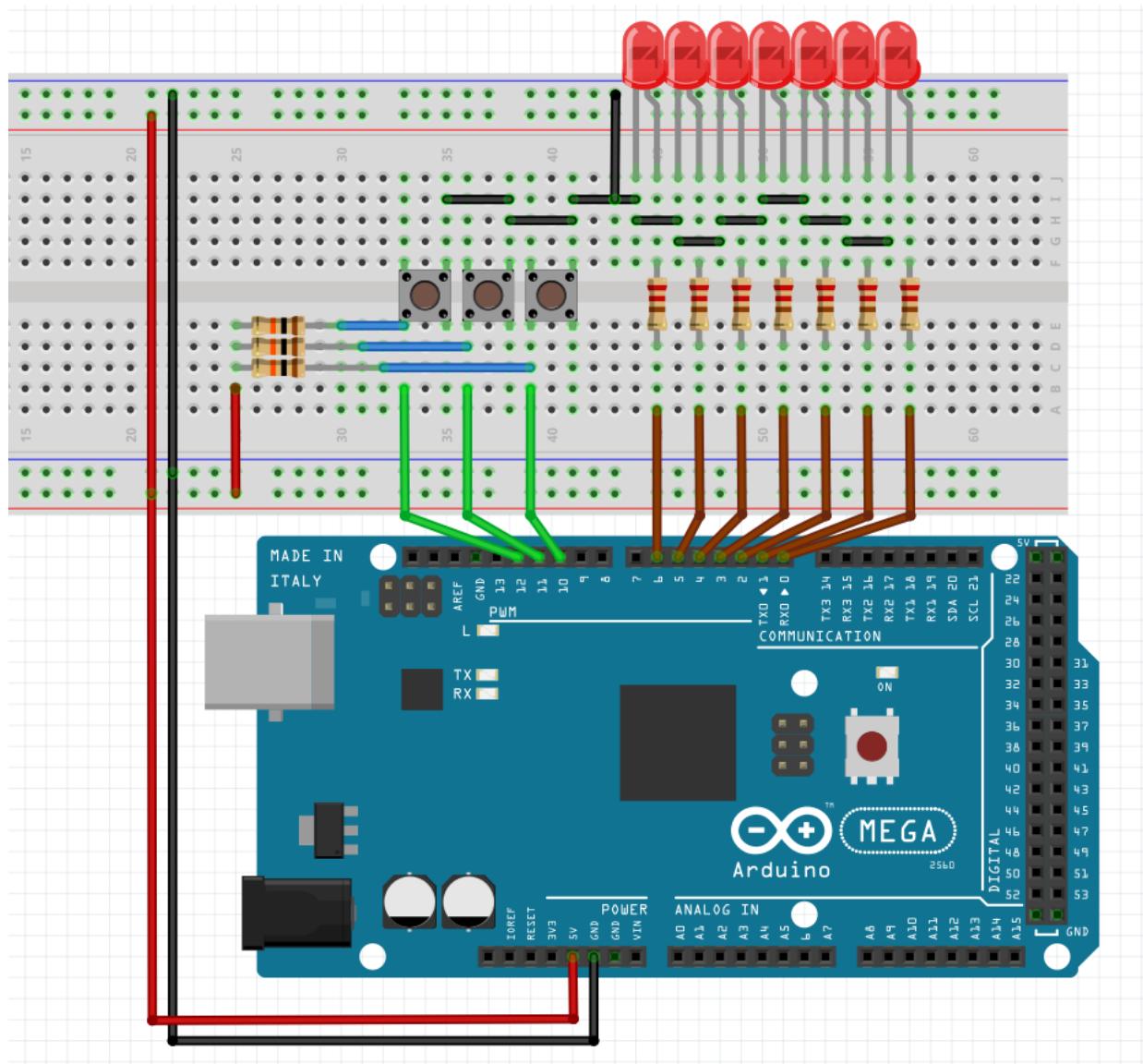
۲. به پرسشهای زیر پاسخ دهید:

- اگر دکمه را در حالت فشرده برای زمان طولانی نگه داریم چه اتفاقی خواهد افتاد؟ آیا با منطق کارکرد خواسته شده سازگار است؟ چه راه حلی برای این مشکل (در صورت وجود) می توان پیشنهاد کرد؟
- فرض کنید می خواهیم برد مورد نظر علاوه بر فراهم کردن کارکرد خواسته شده در بالا، عمل دیگری را نیز به صورت زمان دار انجام دهد. برای نمونه در کنار کارکرد بالا، وضعیت روشن یا خاموش بودن یک LED را نیز هر ۵ ثانیه یک بار تغییر دهد. روشی برای افزودن این کارکرد تازه به برنامه پیشنهاد دهید.

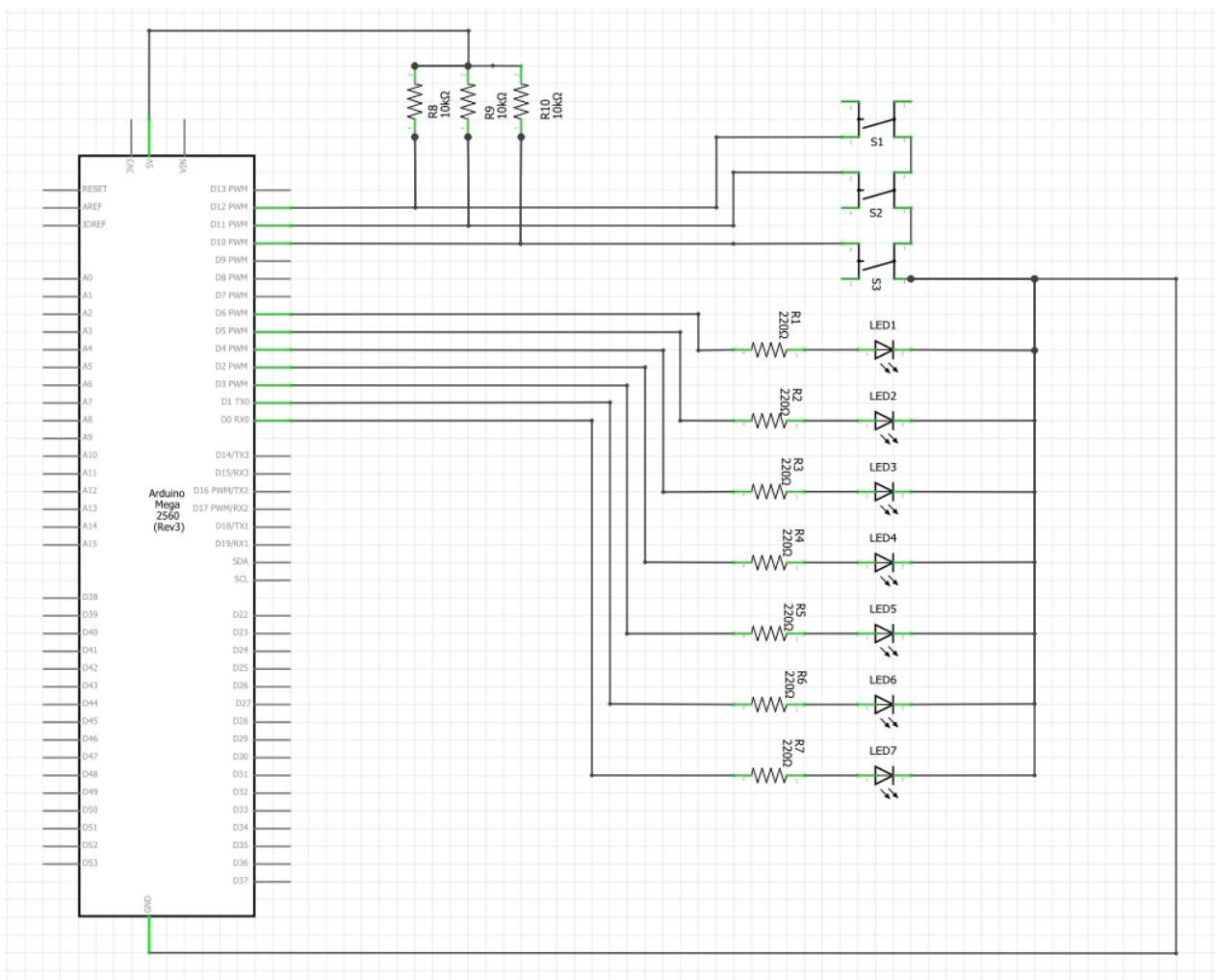
فرض کنید می خواهیم کارکرد دیگری را به دستگاه اضافه کنیم به این صورت که در صورت یک شدن یک پایه عملیات مشخصی را به عنوان پاسخ انجام دهد. (محدودیت زمانی برای پاسخ دادن وجود دارد) هیچ یک از اتفاق های یک شدن پایه نباید از دست برود (بی پاسخ بماند). و یک شدن پایه نیز در هر زمانی ممکن است رخ دهد. آیا برنامه شما که به روش سرکشی واحد های جانی را بررسی می کند- می تواند در هر شرایطی (مثل هنگام فشرده شدن کلید) این کارکرد را فراهم کند؟

فرض کنید به دلیل محدودیت در توان مصرفی می خواهیم پردازنده در هنگام بیکاری به خواب برود. در زمان خواب پردازنده هیچ دستوری را اجرا نمی کند. روش سرکشی چه قدر با این نیازمندی سازگاری دارد؟ آیا می توان با این روش هم به خواب رفت و هم کارکرد درست آزمایش را فراهم کرد؟

۳. با پاسخ به پرسش های بالا می توان دریافت که روش سرکشی برای کنترل واحد های جانی با اینکه در برنامه های کوچک و به نسبت ساده قابل پیاده سازی است، همواره روش خوبی نیست و گاهی نمی تواند نیازمندی های ما را فراهم کند. اکنون آزمایش را به روش وقفه محور انجام دهید. پیاده سازی نیازمندی های خواسته شده در گام دوم را به روش سرکشی و وقفه محور مقایسه کنید.



شكل ١١



شكل 1-2

آزمایش ۲: کیبورد ورودی و ارتباطات سریال

هدف آزمایش:

اتصال صفحه کلید ماتریسی ساده به میکروکنترلر به عنوان رابط کاربر برای وارد کردن اطلاعات

راه اندازی یک ترمینال مجازی با Arduino Mega ۲۵۶۰ و آشنایی با ارتباطات Serial

قطعات مورد نیاز:

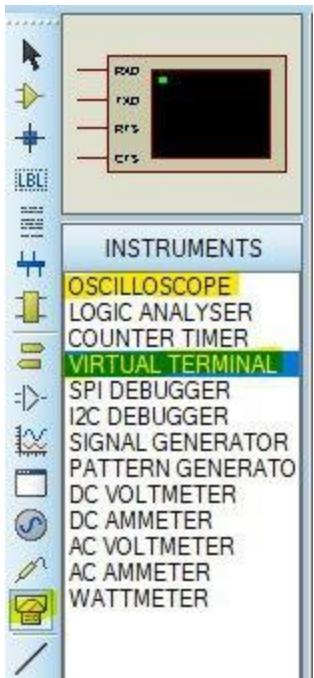
- برد Arduino Mega
- صفحه کلید (Keypad) ماتریسی
- LED
- ترمینال مجازی برای سریال مانیتور
- اسیلوسکوپ

آنچه باید در پیش گزارش نوشته شود:

- کدهای مورد نیاز برای برنامه ریزی برد
- پاسخ به پرسش های دستور کار
- انواع Keypad ماتریسی و چگونگی کارکرد آنها
- پدیده‌ی نوسان (bounce) کلید چیست و چگونه می‌توان از بروز اشکالات ناشی از آن جلوگیری کرد؟
- تعریف مختصر توابع مورد نیاز از کتابخانه Keypad.h مانند:

- [Keypad\(makeKeymap\(userKeymap\), row\[\], col\[\], rows, cols\)](#)
- [Char getKey\(\)](#)
- [Char getKeys\(\)](#)
- [char waitForKey\(\)](#)
- [KeyState getState\(\)](#)
- [boolean keyStateChanged\(\)](#)

- نحوه و کاربردهای ارتباط سریال در آردوینو
- تعریف مختصر و نحوه کار با توابع ارتباط سریال مانند:
-

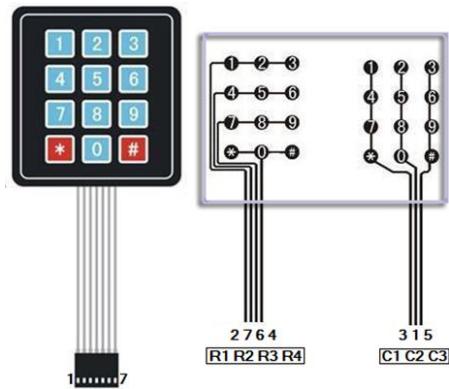
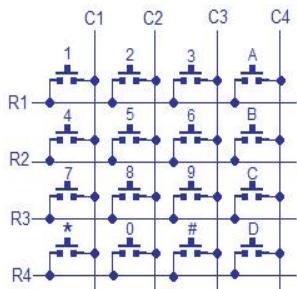


- begin()
- end()
- find()
- parseInt()
- println()
- read()
- readStringUntil()
- write()

: مقدمه

صفحه کلید مجموعه‌ای از دکمه‌ها است که به صورت بلوک یا "پد" تنظیم شده‌اند و به صورت ماتریسی روی هم قرار گرفته‌اند. ماتریسی بودن کلیدها این امکان را می‌دهد تا علاوه بر بهره مندی از تعداد زیادی کلید، تعداد کمی از پایه‌های بردها را اشغال کنیم.

در یک صفحه کلید استاندارد، کلیدهای اصلی به صورت ۴×۷ ماتریس قرار گرفته‌اند که در حالت معمول به صورت ردیفی و ستونی به یکدیگر متصل هستند. اگر یک صفحه کلید، ۱۲ کلید داشته باشد این صفحه کلید به صورت ۳ ستون و ۴ ردیفی به یکدیگر متصل می‌شوند. بد آردوینو می‌تواند از طریق پین‌های دیجیتال به این سطرهای و ستون‌ها دسترسی داشته باشد. وقتی یک کلید فشار داده می‌شود یک سطر و ستون به هم متصل می‌شوند. در غیر این صورت هیچ گونه اتصالی بین سطرهای و ستون‌ها وجود ندارد. شکل زیر ساختمان داخلی یک صفحه کلید ۴×۳ را نشان می‌دهد.



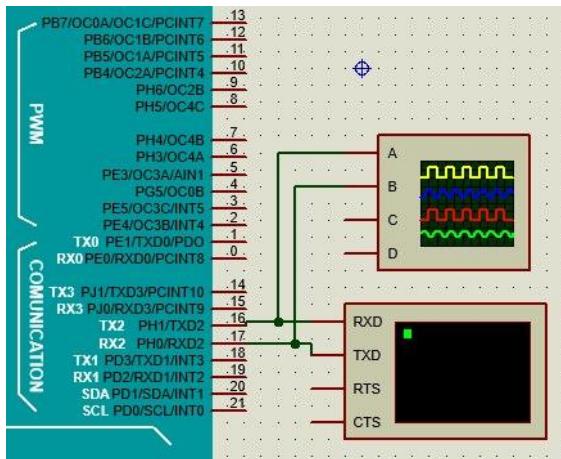
نمایی از ساختار درونی و بیرونی دو کیپد

برای برنامه‌ریزی برد می‌توان از کتابخانه Keypad.h استفاده کرد. در صورتی که کتابخانه Keypad.h در نرمافزار Arduino IDE موجود نباشد، باید کتابخانه‌ی مورد نظر را نصب کنید. برای این کار به مسیر زیر رفته، سپس عبارت Keypad.h را جستجو کنید و کتابخانه مورد نظر را نصب کنید.

Sketch->Include Libraries->manage libraries

و یا آن را از لینک زیر دانلود کنید و در مسیر پوشش \libraries\arduino\libraries ذخیره کنید.

<https://playground.arduino.cc/uploads/Code/keypad/index.zip>

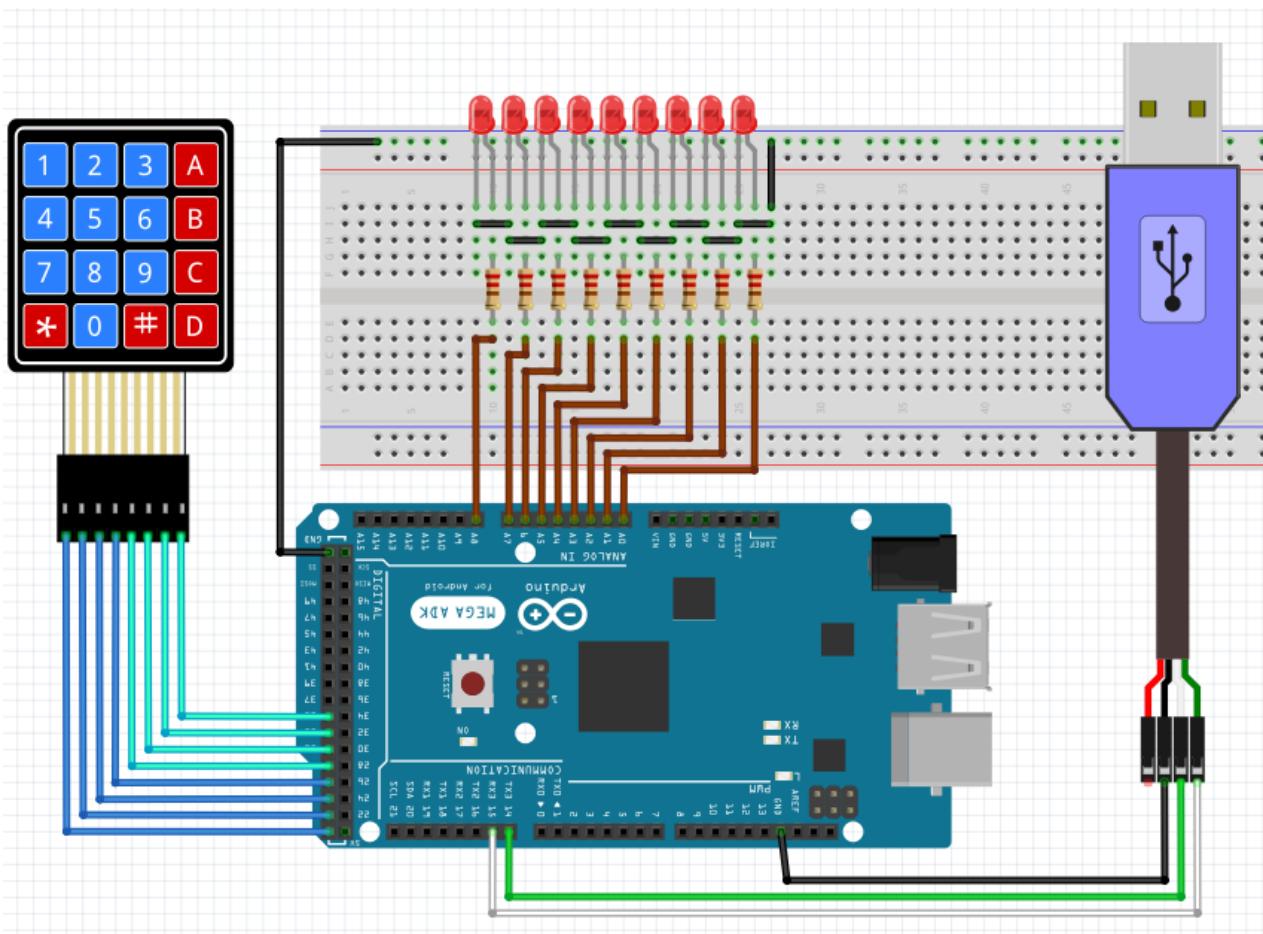


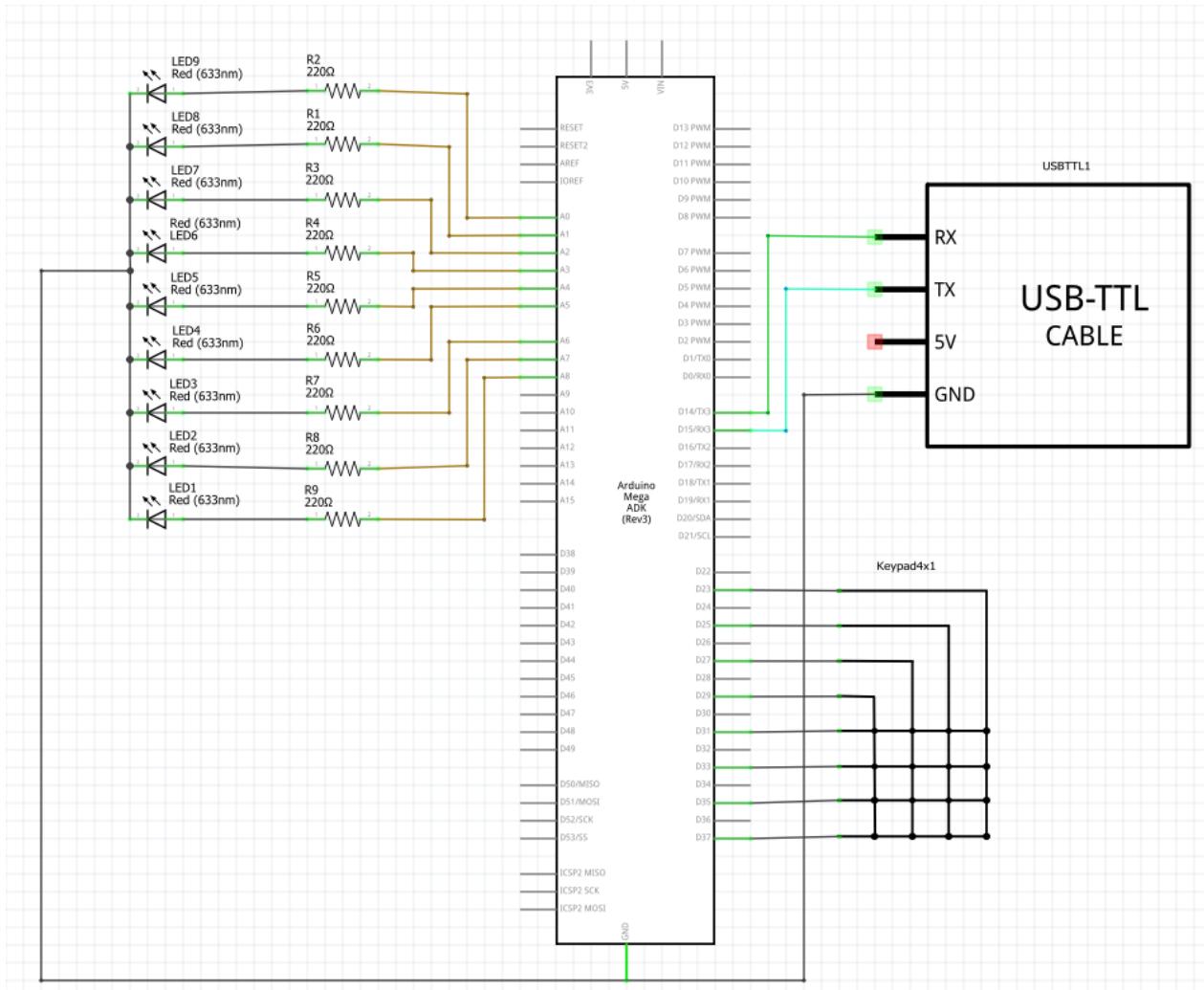
همچنین در این آزمایش قابلیت ارتباط سریال آردوینو را بررسی می‌کنیم. از نوار ابزار پروتئوس (سمت چپ) و از تب INSTRUMENTS یک اسیلوسکوپ و یک ترمینال مجازی را انتخاب می‌کنیم. ترمینال را به یک اسیلوسکوپ را برای مشاهده سیگنال‌های ارتباطی، به همان دو سر ترمینال مجازی متصل می‌کنیم. در آردوینو واحدهای USART و UART با نام Serial به کار بدهد می‌شوند و چنانچه از واحد USART یا USART شماره n استفاده می‌کنید (TXn, RXn) و همچنین اگر $n = 0$ باشد در غیر این صورت Serials نام دارد.

پرسش: در رابطه با نحوه عملکرد ارتباط سریال و در آردوینو و همچنین پروتکل‌های ارتباطی UART و USART توضیح مختصری ارائه دهید.

شرح آزمایش:

۱. نخست ۹ عدد LED و یک صفحه کلید را از میان component‌های پروتئوس اضافه و به برد ۲۵۶۰ Arduino Mega وصل کنید. برنامه‌ای بنویسید که به تعداد عدد انتخاب شده در صفحه کلید از سمت چپ به راست LED‌ها را روشن کند.
۲. سپس ترمینال مجازی را (که در شکل زیر به صورت یک تبدیل USB-TTL نشان داده است) به پین‌های ارتباطی برد وصل کنید. برنامه‌ای بنویسید که کارکتر روی دکمه فشرده شده را در ترمینال مجازی نشان دهد. حال این آزمایش را با اسیلوسکوپ متصل شده به سیم‌های ترمینال مجازی تکرار کنید. سیگنال فرستاده شده به ترمینال روی اسیلوسکوپ را ببینید. آیا می‌توانید آن را بررسی کنید؟
۳. برنامه‌ای بنویسید که ترمینال مجازی، یک عدد بین ۱ تا ۹ را به عنوان ورودی بگیرد و به تعداد آن LED‌ها را از سمت چپ به راست روشن کند. در صورتی که عدد وارد شده بزرگتر از ۹ بود پیام "Invalid number" را به عنوان خطا نمایش دهد.





آزمایش ۳: مانیتور خروجی

هدف آزمایش:

اتصال صفحه نمایش کاراکتری به میکروکنترلر برای نمایش اطلاعات

قطعات مورد نیاز:

- برد ۲۵۶۰ Arduino Mega
- صفحه نمایش کاراکتری ۱۶×۲
- مقاومت 22Ω یا پتانسیومتر $10k\Omega$
- ترمینال مجازی برای سریال مانیتور
- کیبورد ۴x۶

آنچه باید در پیش گزارش نوشته شود:

- کدهای مورد نیاز برای برنامه‌ریزی برد
- مشخصات فنی ماژول نمایشگر ال‌سی‌دی کاراکتری ۱۶×۲ و دلیل استفاده از پتانسیومتر در مدار
- پاسخ به پرسش‌های مقدمه
- تعریف مختصر توابع مورد نیاز از کتابخانه LiquidCrystal مانند:

- [LiquidCrystal\(\)](#)
- [begin\(\)](#)
- [clear\(\)](#)
- [setCursor\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [noDisplay\(\)](#)
- [scrollDisplayLeft\(\)](#)
- [autoscroll\(\)](#)

مقدمه:

در بسیاری از پروژه‌ها ما نیاز داریم تا برخی اطلاعات را توسط نمایشگرها به کاربران نمایش دهیم. به صورت کلی LCD‌های موجود به دو دسته LCD گرافیکی و LCD کاراکتری تقسیم می‌شوند.

پرسش: در رابطه با LCD گرافیکی توضیح دهید.



LCD کاراکتری 16×2 یکی از پایه‌ای‌ترین نمایشگرهای الکترونیکی است. مژول نمایشگر دارای یک صفحه نمایش LCD با قابلیت نمایش ۲ سطر و ۱۶ ستون کاراکتر است. این مژول در شکل (۱-۳) نشان داده شده است. پایه‌های LCD کاراکتری از چپ به راست به صورت زیر است:

پرسش: هر یک از پایه‌های زیر برای چه هدف استفاده می‌شوند؟

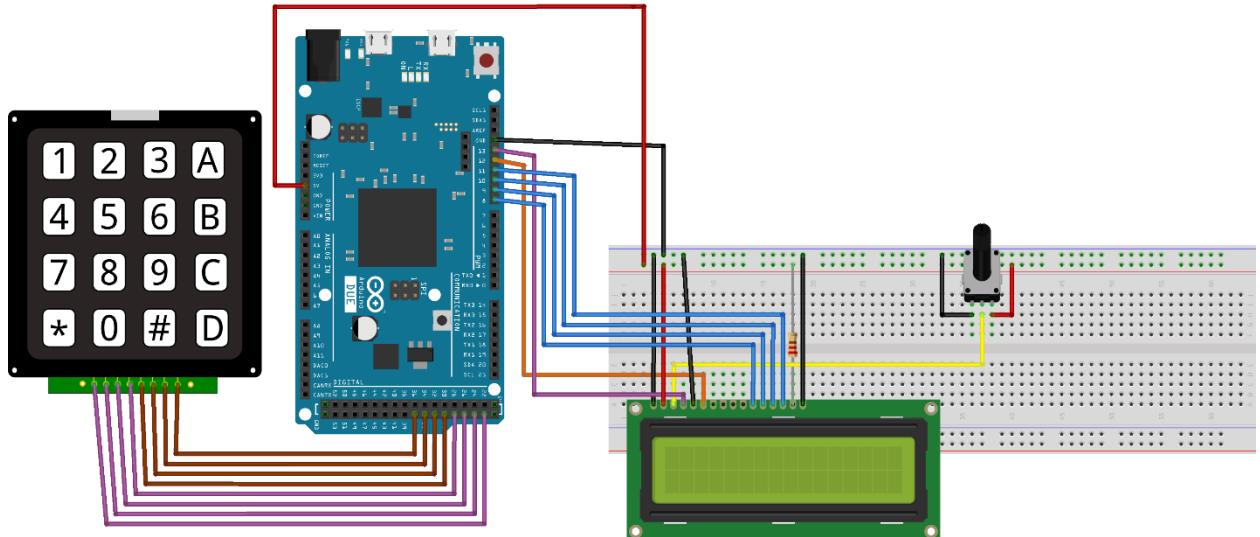
۱. GND = زمین
۲. VCC = تغذیه ۵ ولت
۳. VO (Display Contrast Pin) = تنظیم شدت نور صفحه
۴. RS (Register Select) = انتخاب رجیستر
۵. RW (Read/Write) = پایه Read و Write
۶. E = پایه Enable
۷. D0 – D7 = پایه‌های دیتا (داده ۸ بیتی)
۸. A = پایه Anode
۹. K = پایه Cathode

برای برنامه‌ریزی LCD موجود می‌توانید از کتابخانه LiquidCrystal استفاده کنید که به صورت پیش‌فرض در نرم افزار IDE آردوینو وجود دارد. لازم به ذکر است که در محیط شبیه سازی پروتئوس، مقاومت/پتانسیومتر تاثیری ندارد و صفحه نمایش همواره اعداد را نشان میدهد، در حالی که در عمل میزان ولتاژ آن VO، میزان نمایان بودن اعداد و شدت نور صفحه را تنظیم می‌کند.

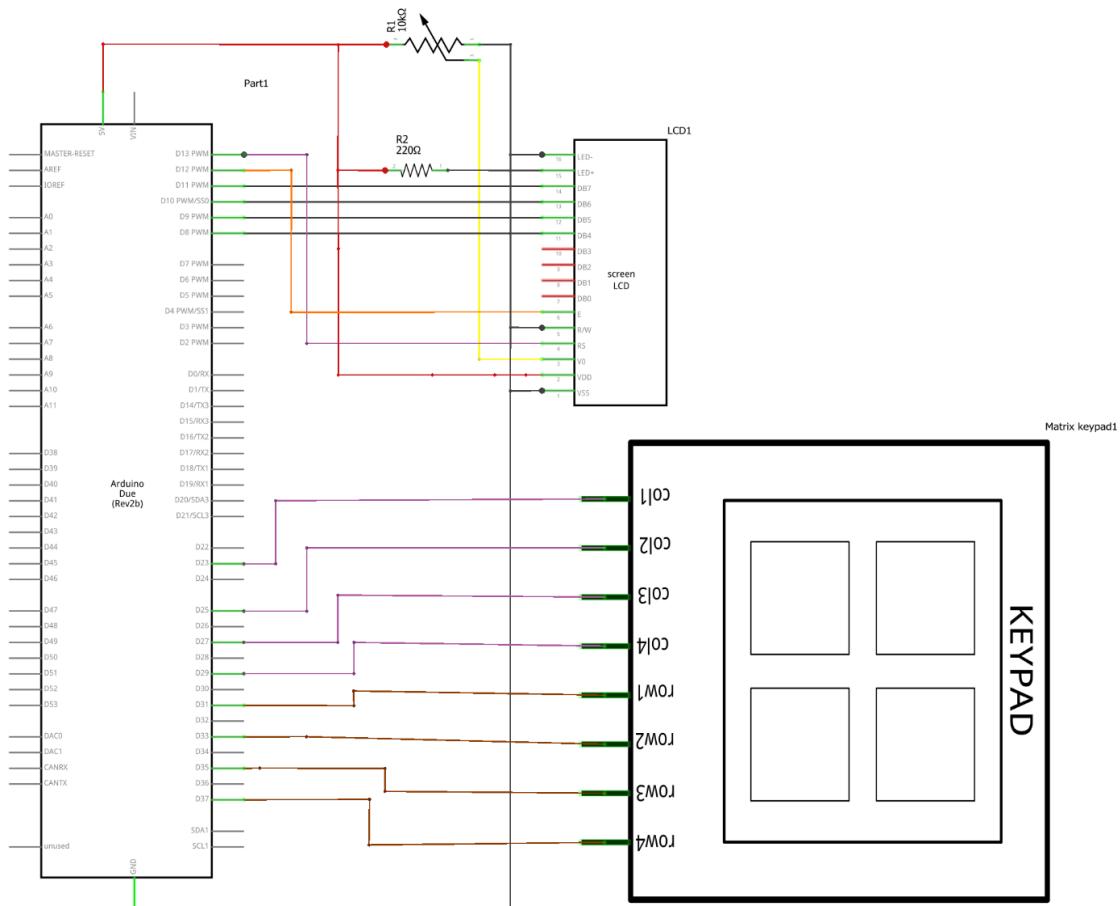
شرح آزمایش:

۱. همانطور که در عکس زیر می‌بینید، ابتدا یک صفحه نمایش را از بین component‌های پروتئوس اضافه کرده و به برد Arduino Mega ۲۵۶۰ وصل کنید. با استفاده از توابع کتابخانه‌ای برنامه‌ای بنویسید که اسم خودتان در ابتدای سطر اول LCD کاراکتری نوشته شده و هر ثانیه یک خانه به جلو حرکت کند و وقتی به انتهای خط اول رسید به خط دوم رفته و باز به سمت جلو حرکت نماید. سپس به طور پیوسته این کار را تکرار نماید.
۲. یک صفحه کلید از پیش‌فرض‌های پروتئوس اضافه کنید و به برد نصب کنید. با استفاده از آن، رمزی را دریافت کنید و روی LCD نمایش دهید، و با زدن کلید * باید درستی این رمز را (مقدار صحیح رمز به صورت پیش‌فرض باید شماره دانشجویی شما باشد) برسی کنید. در صورت درستی رمز عبور پیغام "correct password" در خط دوم نمایش داده شود و در غیر اینصورت "wrong password" نمایش داده شود.
۳. یک کد ماشین حساب بنویسید که بر اساس نوشته‌های کلید (که در LCD مانند $5+23=9$ نوشته شود) دستورات محاسباتی بگیرد و در خط دوم LCD به صورت خروجی نشان دهد. ماشین حساب باید جمع، تفریق، ضرب و تقسیم را انجام دهد.
۴. با استفاده از توابع کتابخانه‌ای برنامه‌ای بنویسید که یک کاراکتر را در ابتدای سطر اول LCD بگیرد و هر ثانیه در حالی که یک خانه به جلو حرکت می‌کند، به خط بعدی برود (اگر در خط اول بود به خط دوم برود و برعکس). سپس بطور پیوسته این کار را تکرار نماید.





fritzing



Matrix keypad1

fritzing

آزمایش ۴: راه اندازی سروو موتور و ورودی آنالوگ

هدف آزمایش

راه اندازی سروو موتور با Arduino Mega ۲۵۶۰ و آشنایی با سروو موتور و مفهوم PWM

خواندن پین های ورودی آنالوگ و تحلیل و استفاده از آنها

قطعات مورد نیاز

- برد Arduino Mega
- سروو موتور (MOTOR-PWMSERVO در پروتئوس)
- پتانسیومتر (POT در پروتئوس) با مقاومت ۱۰ کیلو
- کیبورد X4
- اسیلوسکوپ

آنچه باید در پیش گزارش نوشته شود:

- مفهوم PWM و استفاده های آن
- پاسخ به پرسش های دستور
- کاربردهای سروو موتور
- کدهای مورد نیاز برای برنامه ریزی برد
- توضیح در مورد ورودی آنالوگ و تحلیل آن در آردوینو وتابع مورد استفاده این آزمایش:

- **analogRead()**

تعریف مختصر توابع مورد نیاز از کتابخانه Servo.h مانند:

- attach()
- write()
- read()
- writeMicroseconds()
- readMicroseconds()

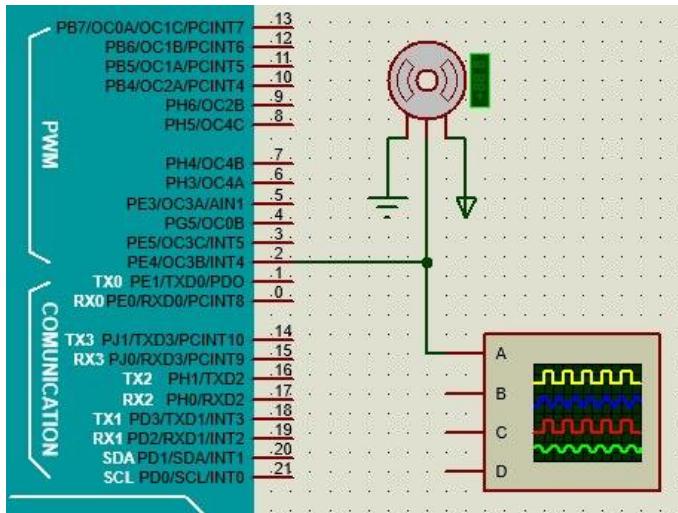
مقدمه:

سروو موتور یک چرخ دنده و مدار کوچک الکترونیکی است و نوعی از موتورهای الکتریکی است که می‌تواند موقعیت زاویه را به صورت دقیق کنترل کند.

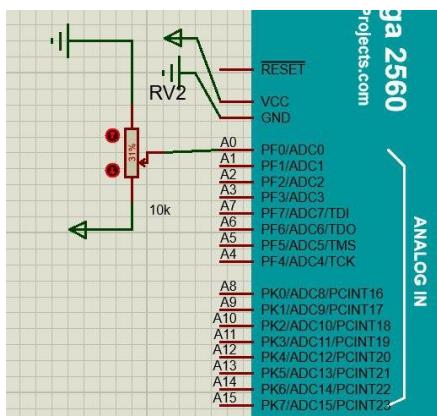
برای مدیریت سروو موتور، نیاز به موج مربی با PWM است. به عبارت دیگر، سروو موتور یک موتور کوچک دارای یک محور یا شفت خروجی است. این محور خروجی قادر است در یک موقعیت و زاویه خاص با سیگنال دریافتی قرار گیرد.

سروو موتور دارای سه پایه به ترتیب GND، VCC، Signal است. سیم تغذیه به +۵ ولت، سیم زمین به زمین مدار، و در آخر سیم سیگنال به یک پین دیجیتال آردوینو که قابلیت PWM داشته باشد متصل می‌شود.

لازم به ذکر است که در محیط آزمایشی پروتئوس، سروو موتور با کمی خطأ عمل می‌کند و با تنظیم و کالیبره کردن PWM می‌توان از مقدار آن کاست.



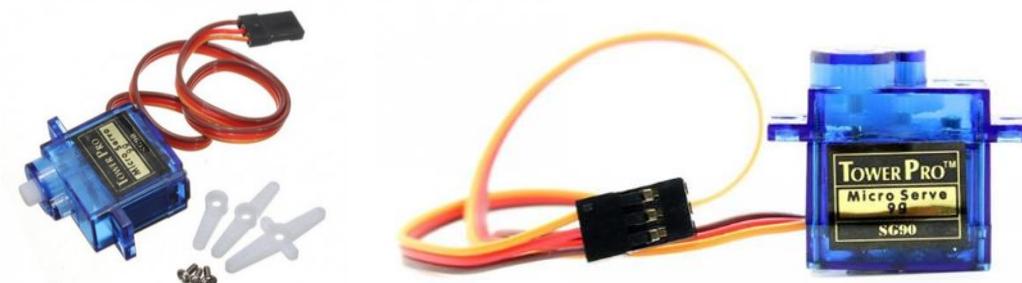
سپس برای اتصال پتانسیومتر، پایه‌های کناری را به تغذیه ۵+ ولت و زمین متصل کرده و سیم وسط را به یک پین آنالوگ آردوینو (مانند A0) می‌زنیم تا ورودی آنالوگ (میزان ولتاژ) را برای تحلیل دریافت کند.



پرسش: ابزار دیگری همانند سروو موتور به نام stepper وجود دارد. در رابطه با نحوه‌ی عملکرد و تفاوت آن با سروو توضیحاتی ارائه دهید.

پیشنهاد: تابع map آردوینو در کار با سروو موتور و خواندن داده آنالوگ پتانسیومتر میتواند به شما کمک شایانی نماید.

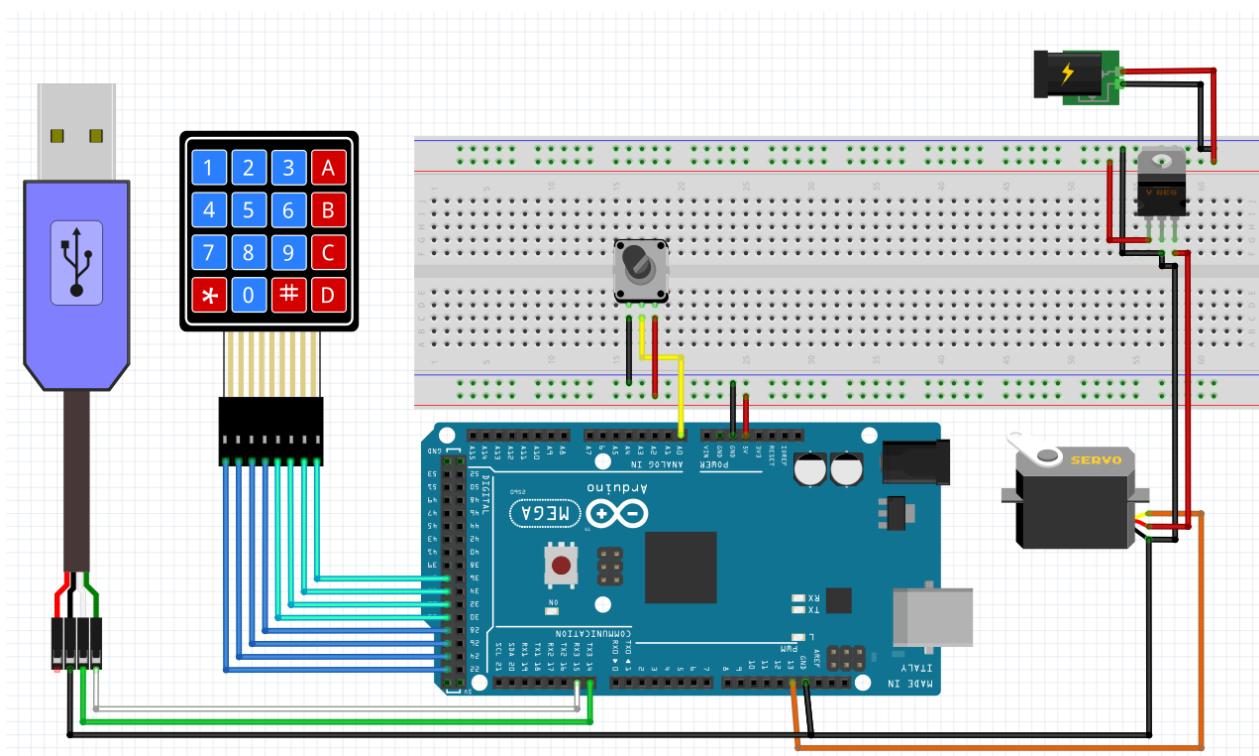
پرسش: در بخش آخر شرح آزمایش به duty cycle و fundamental period اشاره شده است در رابطه با هر یک توضیح دهید.

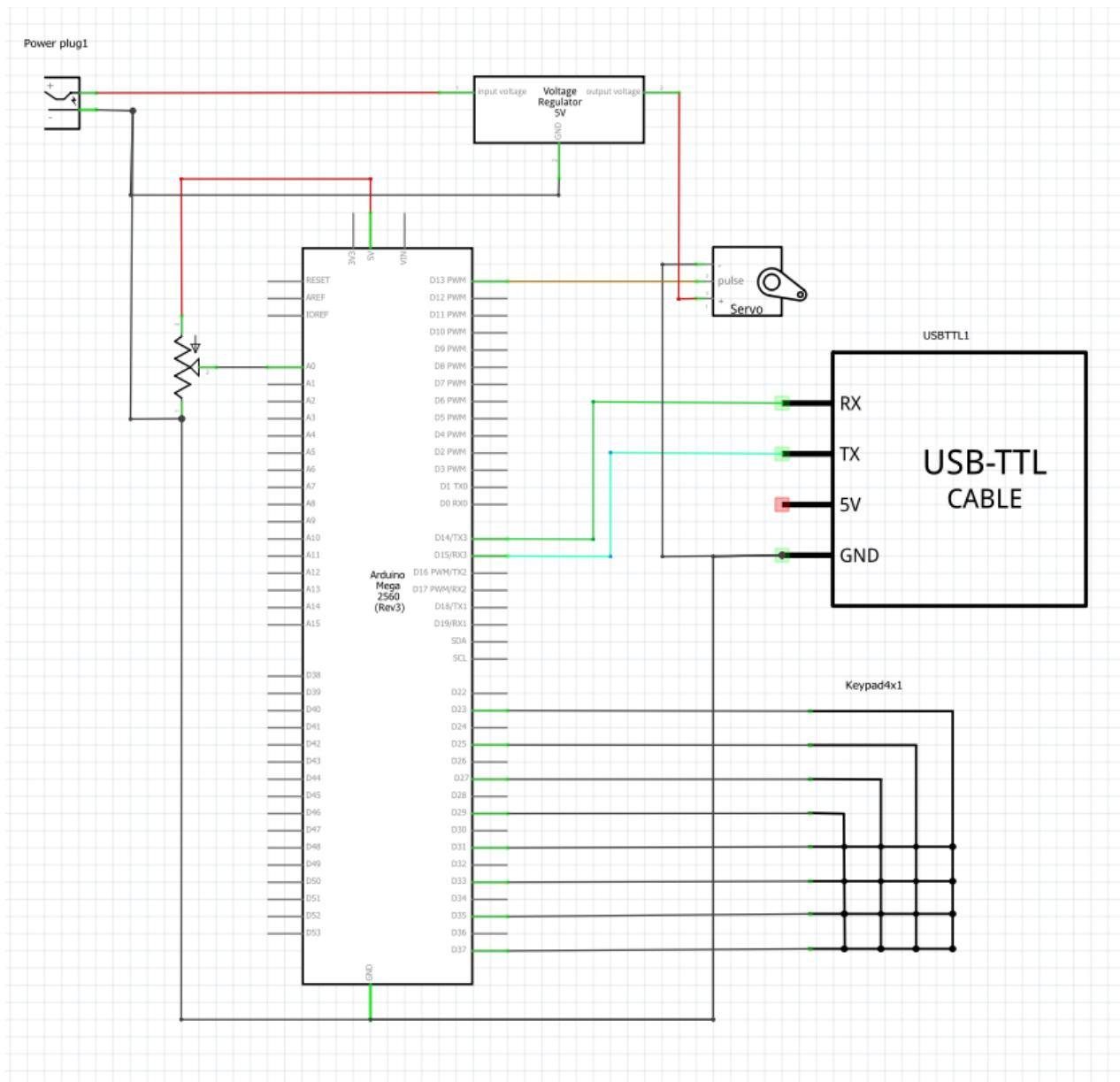


سرورو موتور SG 90

شرح آزمایش:

۱. برنامه‌ای بنویسید که به صورت خودکار سرورو از زاویه ۰ تا ۹۰ درجه تغییر کند و سپس از زاویه ۹۰ به ۰ بازگردد. سپس به صورت متناوب این حرکت را تکرار کند.
۲. برنامه‌ای بنویسید که کاربر با کیبورد یک عدد بین ۰ تا ۳۶۰ انتخاب کرده و سرورو موتور آن را بین -۱۸۰ و +۱۸۰ درجه نشان دهد.
۳. برنامه‌ای بنویسید که با استفاده از سریال مانیتور، مقدار زاویه مورد نظر را وارد کنیم و سرورو موتور به اندازه‌ی قرینه‌ی آن عدد تغییر زاویه دهد.
۴. برنامه باید به گونه‌ای نوشته شود که با تغییر مقدار پتانسیومتر که به یک پایه آنالوگ برد متصل است، زاویه سرورو موتور تغییر کند.
۵. اسیلوسکوپ را به خط سرورو موتور متصل کنید. چه چیزی متوجه می‌شود؟ آیا می‌توانید دوره پایه (Fundamental Duty Cycle) و همچنین دوره کاری (Period) را به ازای زاویه‌های گردش مختلف سرورو موتور به دست آورید؟





آزمایش ۵: راه اندازی رله با Arduino MEGA ۲۵۶

هدف:

آشنایی با سویچ الکتریکی یا رله (Relay) و راه اندازی آن با Arduino Mega

اتصال رله به برد و فعال سازی آن با سیگنال های خروجی از آردوینو برای راه اندازی قطعات مانند لامپ که نیازمند جریان یا ولتاژ بالای هستند و مستقیم با پایه های برد قابل فعال سازی نیستند.

قطعات مورد نیاز:

Arduino Mega	بورد	•
	رله	•
	موتور	•
	LED	•
	مقاومت $1\text{ k}\Omega$ و $10\text{ k}\Omega$	•
	ترانزیستور NPN	•
	دیود	•
	باتری	•
	دکمه	•
	ولت سنج (DC Voltmeter)	•
	آمپرسنج (DC Ammeter)	•

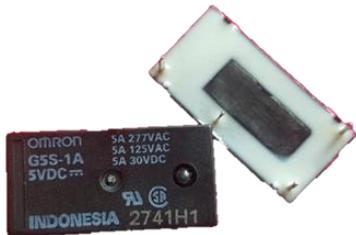
آنچه باید در پیش گزارش نوشته شود:

- رله چیست؟ انواع رله و کاربردهای آن را ذکر کنید.
- آشنایی با پایه های رله، نحوه کار کرد آن و نحوه تشخیص پایه های رله
- تشخیص پایه های رله چگونه انجام می شود؟
- پاسخ به پرسش های مقدمه

مقدمه

کلیدها می توانند با فرمان های مختلف تحريك شوند. فشار مکانیکی، نور، لرزش و صد البته جریان الکتریکی! وقتی فرمان یک کلید جریان الکتریکی است، نام سوئیچ یا رله (Relay) را برای کلید انتخاب می کنیم. در واقع رله یک کلید تبدیل است، با این تفاوت که در کلید تبدیل به یک انسان نیاز است تا با دست خود، کلید تبدیل را فشار دهد. ولی در رله یک جریان برق این کلید را تغییر حالت می دهد. یعنی ما یک ولتاژ برق را به رله می دهیم و رله، کلید تبدیلی که در داخل آن تعییه شده است را برای ما خاموش و روشن می کند. از آنجا که رله ها می توانند جریان قوی تر از جریان ورودی را هدایت کنند، به معنی وسیع تر می توان آنها را نوعی تقویت کننده نیز دانست.

رله ها چندین ساختار مختلف دارند. که با توجه به تعداد قطب ها و خروجی های آن، دسته بندی می شوند. اما ساده ترین و پر کاربرد ترین نوع آنها، رله های SPDT هستند. نمونه ای از این نوع رله در شکل (۱-۶) نشان داد شده است. این حروف مخفف عبارت تک قطبی دو خروجی (Single Pole Double Throw) است. به این معنا که این نوع رله ها دارای ۵ پایه هستند که دو پایه coil برای فرمان (قسمت فرمان) و سه پایه برای خروجی (مدار قدرت) دارند.



دو نمونه از رله های ۵ ولت SPDT



تمامی ارائه‌های درسی را می‌توانید در [دانلود](#) کنید.

۱. پایه Common (COM): پایه مشترک میان دو پایه NO و NC می باشد که بسته به شرایط به یک از این دو پایه متصل است. می توانیم این پایه را به یک سر موتور متصل کنیم.
 ۲. پایه های COIL: این پایه ها مربوط به COIL در رله است که هنگامی که به این دو پایه یک ولتاژ مثبت و منفی DC وصل شود کلید تغییر وضعیت می دهد.
 ۳. پایه Normally closed (NC): پایه ای که در حالت عدم تحریک کویل به COM متصل است. می توانیم این پایه را به منفذ تغذیه متصل کنیم.
 ۴. پایه Normally open (NO): پایه ای که در حالت تحریک کویل به COM متصل می شود. می توانیم آن را به مثبت تغذیه متصل کنیم.

بنابراین آن سر موتور نیز به منفی تغذیه متصل می‌کنیم.

پرسش: درباره چگونگی کارکرد این مدار توضیح دهید.

پرسش: از جمله دیگر ساختارهای معروف رله DPDT، SPST، DPST را می‌توان نام برد. ساختار درونی آن‌ها به چه صورت می‌باشد؟

شرح آزمایش:

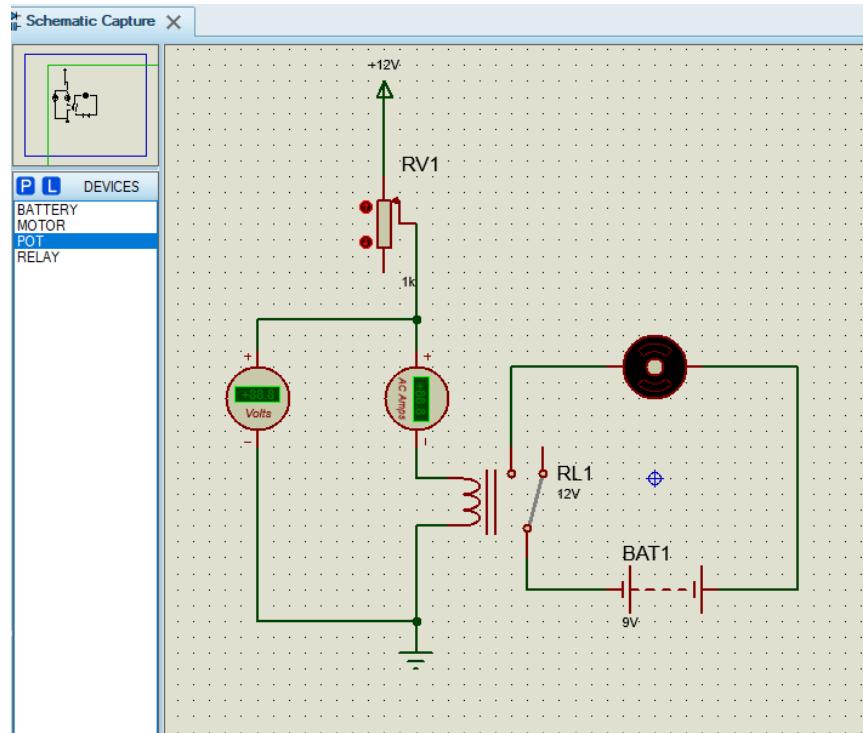
در این آزمایش همانند مدارهای شکل ۱ و ۲ که در زیر آورده شده است، می‌خواهیم با رله یک موتور را راه اندازی کنیم، هنگامی که دکمه را فشار میدهیم موتور روشن و LED خاموش می‌شود و هنگامی که دکمه را رها می‌کنیم موتور خاموش و LED روشن می‌شود.

ابتدا مدار زیر را با پتانسیومتر(POT)، رله(Relay)، موتور(Motor)، باتری، آمپرmetr و ولت متر رسم کنید. و آن را اجرا نمایید. ابتدا POT را بر روی بیشترین مقاومت قرار دهید. در این حالت ولت متر و آمپرmetr کمترین مقدار را نشان خواهند داد. همچنین رله غیرفعال است.

سپس رفته مقدار POT را افزایش دهید و آستانه ای را که در آن رله فعال می شود را به دست آورید. و مقدارهایی را که دستگاه های ولت متر و آمیر نشان می دهند بررسی کنید.

همان گونه که دیده می شود ولتاژ آستانه مورد نیاز برای فعال شدن رله به مراتب بیشتر از ولتاژ یک منطقی ۲۵۶۰ ATMega است.
(v0)

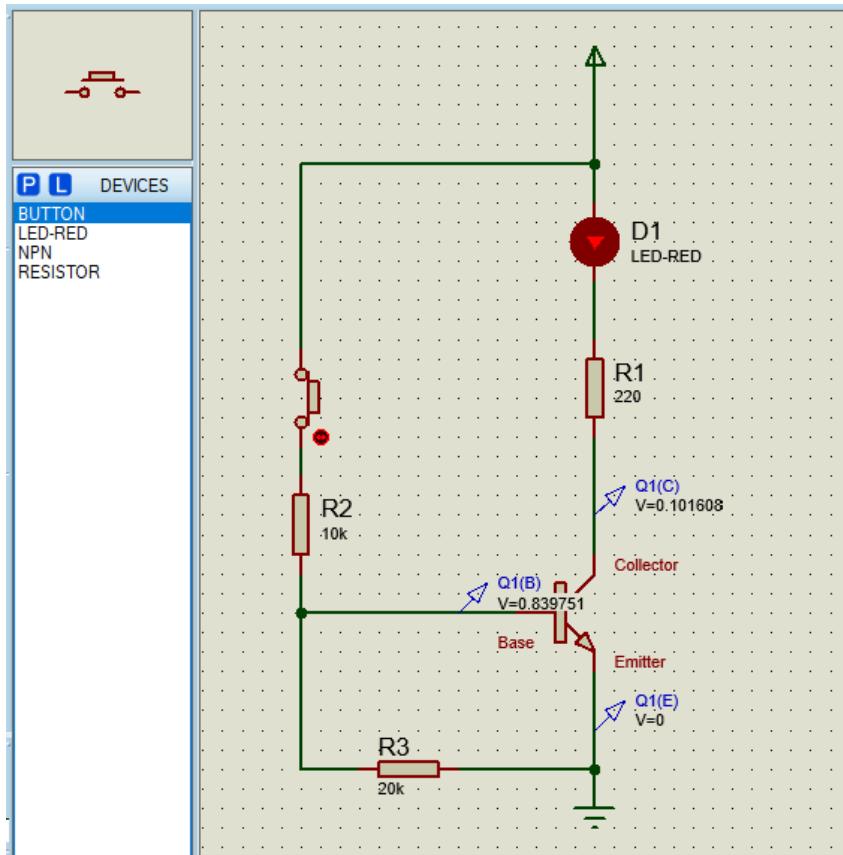
همچنین جریانی که برای فعال کردن رله نیاز هست نیز معمولاً از بیشینه جریانی که هر پین ورودی/خروجی ATMega ۲۵۶۰ فراهم می کند بیشتر است. (mA40.)



نتیجه: از این رو بین های ورودی/خروجی این میکروکنترلر به تنها بی توانایی فعال کردن رله یا آرمیچر را ندارد و به کاربردن آن ها برای این منظور می تواند سبب آسیب رسیدن به میکروکنترلر و سوختن آن شود، در نتیجه از ترانزیستور برای فعال کردن رله بهره می ببریم.

گام دو:

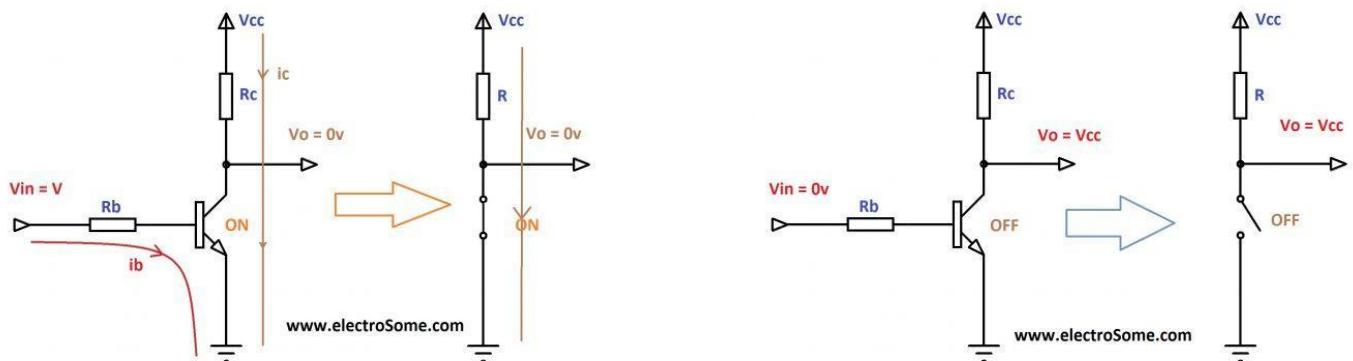
برای آشنایی با شیوه کارکرد ترانزیستورهای NPN مداری مانند مدار زیر را در Proteus رسم کرده و آن را اجرا کنید.



همان گونه که دیده می شود به طور خلاصه می توان گفت به ازای افزایش جریان از پایه Emitter (I_B)، بیشینه جریانی که از Emitter به Collector می تواند بگذرد به نسبت بیشتری افزایش می یابد. به گونه ای که در حالت اشباع ترانزیستور می توان از مقاومت در مدار Emitter به Collector از پوشی از ترانزیستور از چشم پوشی کرد.

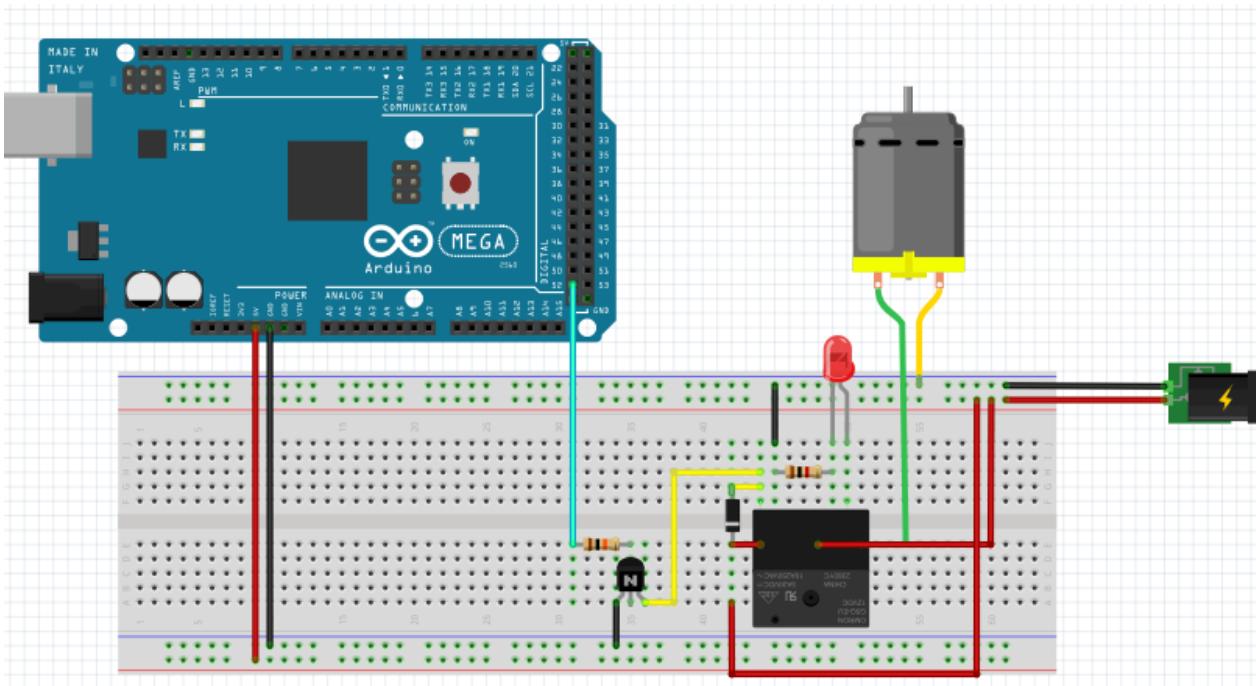
حالت کلید بسته:

حالت کلید باز:

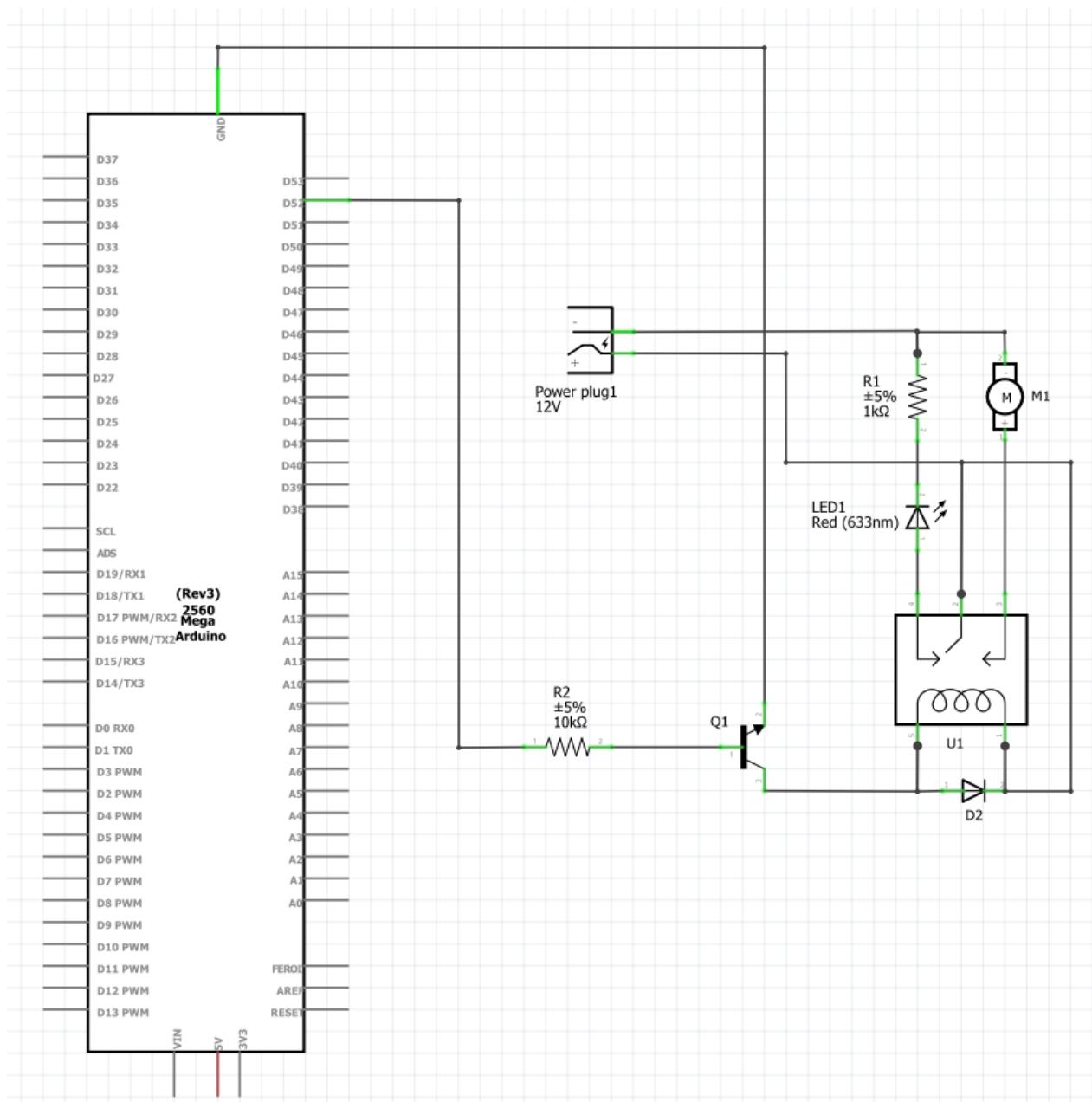


گام سوم:

از این ویژگی ترازنیستور برای راه اندازی رله استفاده کنید و مدار نهایی خواسته شده برای این آزمایش را که مدار شماتیک آن در شکل ۲ آورده شده است، رسم کنید.



نمودار ۱ (مدار آزمایش بر روی برد برد)



نمودار ۲ (شماتیک لازم برای راه اندازی موتور به کمک رله و بورد

آزمایش ۶: نیم-پروژه

هدف آزمایش:

مروری بر مطالبی که در جلسه‌های گذشته گفته شد و پیاده‌سازی یک کاربرد ساده از این مطالب با بهره‌گیری از خلاقیت و ایده پردازی خودتان هدف این آزمایش است. به عنوان مثال، تعدادی ایده پیشنهادی در زیر آورده شده است.

آنچه باید در پیش‌گزارش نوشته شود:

- کدهای مورد نیاز برای برنامه‌ریزی برد

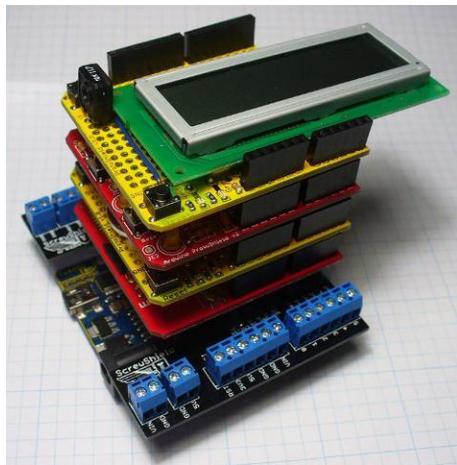
آردوینوهای همکار

هدف پروژه

پیاده‌سازی دو عدد آردوینو که با فرستادن و دریافت اطلاعات از راه ارتباطات سریال، می‌توانند یکسری مسائل بازگشتی (مانند بررسی زوج-فرد بودن بازگشتی) را حل کنند.

قطعات مورد نیاز:

- دو عدد بورد Arduino Mega
- کیبورد
- LCD کاراکتری



کامپیوتر ماشین کنترلی

هدف پروژه

این ماشین باید از ارتباطات سریال دستور بگیرد (که فرضاً به کنترلر رادیویی متصل شده)، چراغهای ماشین (LED) را روشن کند، به چرخهای جلو زاویه بدهد (سرورو موتور)، چرخهای عقب را به حرکت بیندازد (رله و آرمیجر) و بوق بزند (بازر). لازم است شما از سه حسگر مادون قرمز، لایت و فوتولسل استفاده کنید تا مشخص کند ماشین در خط مستقیم کند، به راست برود یا به چپ بپیچد و در صورت پایان خط باید ماشین بایستد. این ماشین دو mode دارد که مود مورد نظر توسط کیپ دریافت و روی LCD نمایش داده می‌شود. مود‌ها عبارتند از stepper mode و servo mode. در حالت سرورو تغییر جهتی که در ابتدا گفته شد انجام می‌شود و در حالت استپر حرکت ماشین همانند ویلچر است stepper یک موتور همانند سرورو است و یک کتابخانه دارد. برای اطلاعات بیشتر درباره آن به [اینجا](#) مراجعه کنید. برای تغییر سرعت ماشین نیز یک پتانسیومتر روی آن قرار دهید.



قطعات مورد نیاز:

- بورد Arduino Mega
- دیود نورافی (LED)

- سروو موتور
- آرمیچر
- رله
- بازر
- نمایشگر (LCD)
- کیبورد

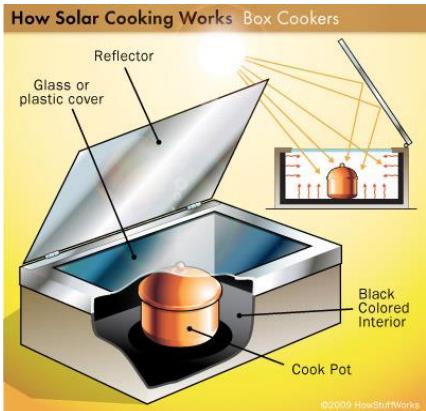
اجاق خورشیدی

هدف پروژه

اجاق خورشیدی دستگاه نور خورشید را از شیشه می گذراند و توسط دیوارهای سیاه درون، گرما را به دام می اندازد. میکروکنترلر شما باید شیشه را باز و بسته کند (سروو موتور) یک دما و یک زمان بگیرد (کیبورد) که هنگامی که حرارت درونی به دمای گرفته شده رسید (حرارت سنج) یک تایмер را شروع کند و پس از گذشت زمان به همان اندازه ای که گرفته شده است، با بازر (buzzer) کاربر را از پخت غذا آگاه کند. لازم است زمان گرفته شده روی LCD نمایش داده شود و به صورت معکوس تا تمام شدن پخت از مقدار آن کم شود. اجاق شما باید یک اپشن reset داشته باشد تا در صورت نیاز کاربر بتواند فرایند پخت را متوقف کند.

قطعات مورد نیاز:

- بورد Arduino Mega
- سروو موتور
- حرارت سنج
- کیبورد
- بازر
- LCD



یعقوب برق رایگان!



هدف پروژه

سیر کردن دانشجوها. یک گذر واژه هفت رقمی (کیبورد) در را باز می کند (یک سرو و موتور) همه بیچش ها را (تعداد زیادی سرو و موتور) به عقب می چرخاند تا غذا کار گذاشته شود. همه می چراغها (LED) در این مرحله خاموش است. دکمه دوباره در را می بندد و چراغها روشن می شوند. سپس دانشجوها با زدن شماره های دور قمی غذا ها، آن را تحويل میگیرند. همه چیز روی LCD نشان داده شود و دکمه پاک و تایید داشته باشد.

قطعات مورد نیاز:

- بورد Arduino Mega
- دیود نورافی (LED) به مقدار لازم
- تعداد زیادی سرو و موتور
- کیبورد
- LCD کاراکتری

پنکه



هدف پروژه

خنک کردن دانشجوها. یک دکمه (button) به کمک رله پنکه را به راه می اندازد (آرمیجر و رله)، هنگام فشرده شدن دکمه، بازر بوق می زند (buzzer) و یک دکمه دیگر بین سه سرعت مختلف آرمیجر سوییج می کند که هر سه توسط یک نور (LED) به کاربر نشان داده می شوند. یک دکمه دیگر هم به کمک سرو و موتور چپ و راست شدن پنکه را فعال و غیرفعال می نماید. هم چنین پنکه باید توانایی برنامه پذیری برنامه داشته باشد، به این شیوه که با یک صفحه کلید و برنامه دوره زمانی رفت و برگشت پنکه و هم چنین سرعت موتور مشخص می شود. که این برنامه بر روی LCD کاراکتری نمایش داده خواهد شد.

برای این که برنامه خودکار و کارکرد دستی نیز با یکدیگر تداخل نداشته باشند یک دکمه را نیز برای انتخاب روش کاری (خودکار یا دستی) به کار ببرید.

قطعات مورد نیاز:

- بورد Arduino Mega
- سه عدد LED
- (push button) چهار دکمه
- آرمیجر و رله



- بازه
- صفحه کلید
- LCD کاراکتری

گاؤصندوق

هدف پروژه

کاربر گذرواژه را وارد می‌کند (صفحه کلید) و سپس در گاؤصندوق باز (سرво موتور) و چراغ نمایانگر باز بودن در روشن خواهد شد. و باز ر به هنگام باز و بسته شدن در هشدار می‌دهد (بوق). کاربر می‌تواند با فشردن دکمه در را بیندد، همچنین زمان شماری راه اندازی می‌شود که پس از گذشت زمان مشخصی اگر در هم چنان باز باشد آن را خواهد بست. و چراغ نمایانگر بسته بودن روشن خواهد شد.

بر روی LCD درستی یا نادرستی گذرواژه، وضعیت در و زمان سنج بسته شدن در نمایش داده شود. همچنین باید بازه زمانی گفته شده و گذرواژه تازه را بتوان از صفحه کلید دریافت کرد و باز تنظیم نمود. لازم است گاؤصندوق یک دزدگیر داشته باشد به این صورت که در صورتی که رمز ۴ بار غلط وارد شد آذیر خطر را فعال کند و در LCD هشدار ورودی غیر مجاز چاپ شود همچنین پس از آن امکان وارد کردن رمز گاؤصندوق به مدت یک دقیقه غیر فعال می‌شود. اگر پس از ۴ بار اشتباه و همچنین گذشت یک دقیقه رمز درست وارد شد در گاؤصندوق باز می‌شود در غیر این صورت باید به مدت دو دقیقه غیر فعال شود و به همین ترتیب.

قطعات مورد نیاز:

- بورد Arduino Mega
- LED به مقدار لازم
- کیبورد
- LCD کاراکتری
- سرو موتور
- Buzzer

تردمیل



هدف پروژه

تردمیل با رله، موتور خود را به برق متصل می‌کند. دو نوع تنظیم دارد، یکی سرعت و دیگری شبی (که با یک سرو موتور قوی کنترل می‌شود). کاربر قبل از شروع به کار تردミل میتواند با چهار دکمه، سرعت و شبی را بالا و پایین بیاورد. افزایش سرعت و درجه ی شبی باز روی LCD نمایش داده شود. همچنین می‌توان سرعت و شبی مورد نظر را به یکباره توسط کیپد وارد کرد و لازم نباشد یکی یکی آن را افزایش داد.

قطعات مورد نیاز:

- بورد Arduino Mega
- سرو موتور
- رله
- موتور

دکمه و صفحه نمایش •

LCD

آزمایش ۷: ماشین لباسشویی ساده

هدف آزمایش:

آشنایی با پروتکل ارتباط سریال TWI

کار با حافظه EEPROM بیرونی و نوشتن داده‌های مانند تنظیمات دستگاه بر روی آن و یا خواندن از آن

قطعات مورد نیاز:

- Arduino Mega
- AT24C02
- LED
- صفحه کلید (Keypad)
- LCD کاراکتری

آنچه باید در پیش‌گزاری نوشته شود:

- به پرسش‌های درون مقدمه پاسخ داده شود.

مقدمه

در این آزمایش می‌خواهیم با پروتکل ارتباط سریال TWI بیشتر آشنا شویم. برای این هدف، از حافظه‌ی EEPROM با نام AT24C20 .
که با پروتکل ارتباط سریال کار می‌کند، استفاده می‌کنیم.

حافظه EEPROM:

فرض کنید می‌خواهیم یک ماشین لباسشویی طراحی کنیم، یکی از کارکردهای مورد نیاز ما این است که کاربر بتواند مدهای کاری دلخواه خود برای شستشو را تنظیم کند، از سوی دیگر کاربر مایل نیست به ازای هر بار کار با ماشین لباسشویی، دوباره این تنظیمات را انجام دهد. برای این کار نیاز است که مدهای دلخواه کاربر را در یک حافظه‌ی دائمی نگه داری کنیم که با خاموش کردن ماشین لباسشویی همچنان وجود داشته باشد، همچنین این داده‌ها احتمالاً به بیشتر از چند متغیر چند بایتی فضای برای ذخیره‌سازی نیاز ندارند و ممکن است برای نگهداری ویرایش یک مد کاری که کاربر انجام می‌دهد، نیاز باشد برخی از متغیرها را باز مقداردهی کنیم (بایت به بایت). برای پیاده‌سازی چنین کثیردی به EEPROM نیاز داریم.

پرسش: در چه کاربردهای EEPROM به کار برده می‌شود؟ چرا در اینجا حافظه Flash یا RAM را به کار نمی‌بریم؟ تفاوت حافظه EEPROM با RAM در چیست؟

پرسش: اگر بخواهیم برای نگهداری مدهای کاری حافظه **Flash** را به کار ببریم، فرآیند نوشتن باید چگونه انجام شود که داده‌های دیگری که بر روی همان بلاک هستند از دست نروند؟

در یک دسته‌بندی می‌توان EEPROM را به دو نوع درونی (Internal) و بیرونی (External) دسته‌بندی کرد. حافظه‌های درونی در کنار پردازنده بر روی یک IC قرار می‌گیرند. در نتیجه، نیازی به کارکرد پایه‌های آدرس (A0-A) نیست و می‌توان سریار استفاده از پروتکل ارتباط سریال برای نوشت و خواندن را از میان برداشت. بنابراین حافظه‌های درونی تندتر هستند، ولی اندازه‌شان محدود است.

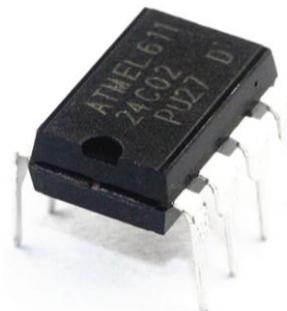
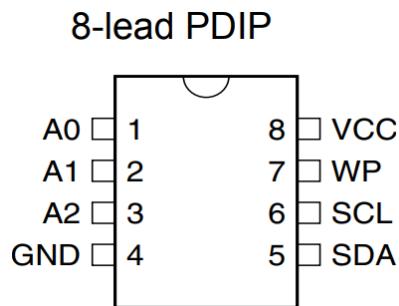
میکروکنترلر ATMEGA ۲۵۶۰ نیز دارای یک EEPROM درونی است. که البته در این آزمایش مورد توجه ما نیست.

گونه‌ای دیگر از EEPROM ها بیرونی هستند که بر روی یک IC جداگانه قرار می‌گیرند. برای ارتباط با این حافظه‌ها اکثراً از پروتکل‌های ارتباط سریال استفاده می‌شود. حافظه ۲AT24C ۲ نیز که در این آزمایش به کار برده می‌شود، پروتکل سریال TWI را به کار می‌گیرد. به این صورت که در این پروتکل حافظه‌ها همان برده‌ها (Slaves) هستند و برد کنترلر نیز سرپرست (Master) است. در این دسته گاهی برای این که بتوان چند حافظه را بر روی یک باس کنترل کرد (چند برده داشت) از پایه‌های آدرس استفاده می‌شود. این آدرس به صورت سخت‌افزاری پیکربندی می‌شود (Hard-Wired).

نمای بیرونی و پایه‌های ۲AT24C

در حافظه‌ای که در این آزمایش با آن کار می‌کنیم، سه پایه برای آدرس قطعه وجود دارد که در نتیجه ۳ قطعه از آن را می‌توان بر روی یک باس مشترک قرار داد.

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply



جدول کارکرد پایه‌ها

نمایی از پایه‌های ۲AT24C

تصویری از ۲AT24C

پرسش: اگر یک حافظه‌ی EEPROM بیرونی دارای ۴ KB حافظه و ۲ پایه آدرس باشد، در این صورت می‌توان حداقل چند KB حافظه بیرونی بر روی یک باس مشترک داشت؟

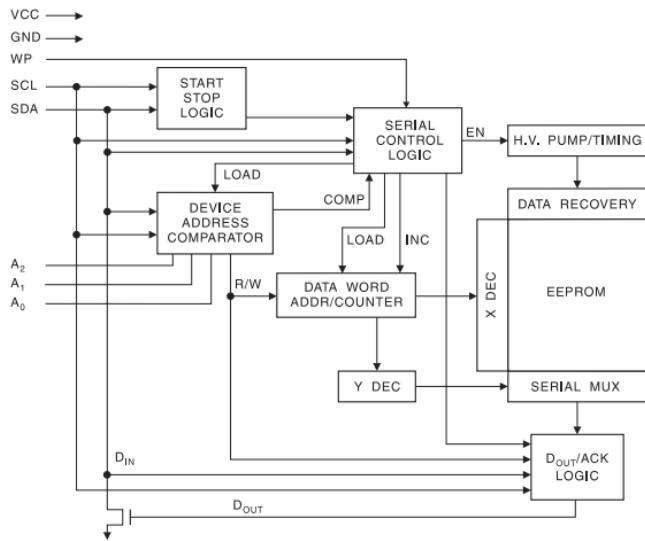
۲AT24C ۰ از پروتکل TWI بهره می‌برد؛ با این تفاوت که به جای ۷ بیت، تنها ۳ بیت برای آدرس دهی اختصاص یافته (A3-A) و ۴ بیت دیگر (پر ارزش‌تر) مقداری ثابت و برابر با ۱۰۰b دارند.

کارکرد دیگری که معمولاً در EEPROM های بیرونی وجود دارد، حفاظت نوشت (Write Protection) یا (Write Enable Protection) است. این ویژگی به این صورت است که هنگامی که سطح منطقی پایه WP برابر با ۰ باشد، خواندن و نوشت نهادی انجام شدنی است ولی هنگامی که مقدار ولتاژ منطقی آن برابر با ۱ باشد، نوشت غیرفعال می‌گردد.

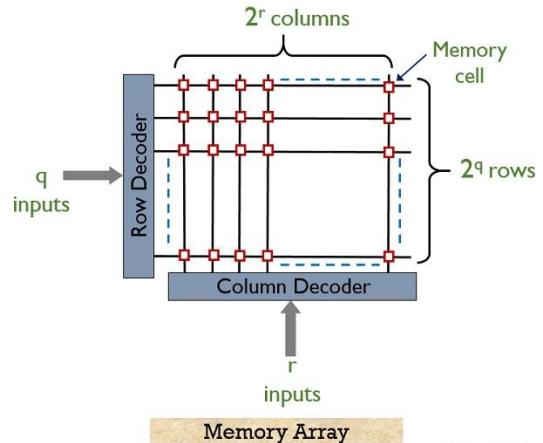
پرسش: نمودار شماتیک برای این که دو ۲AT24C را به یک باس مشترک وصل کنیم و حفاظت نوشت غیرفعال باشد را رسم کنید. (آدرس دهی سخت‌افزاری دلخواه - باس را هم به پایه‌های میکروکنترلر متصل کنید)

ساختار درونی ۲AT24C

بلوک دیاگرام حافظه ۲AT24C



ساختار عمومی ماتریس حافظه



Electronics Desk

شکل بالا ساختار درونی حافظه EEPROM را نشان می‌دهد، برای هر بار انجام عملیات نوشتمن در آغاز باید آدرس خانه‌ی مورد نظر در شمارنده آدرس (Data Word Addr/Counter) قرار بگیرد. همان‌گونه که دیده می‌شود مقدار این شمارنده و رویدی دیکتر آدرس (X) است و به این طریق ردیف مورد نظر در ماتریس حافظه انتخاب می‌شود. پس از انجام عملیات نوشتمن، مقدار شمارنده یک واحد افزایش می‌یابد. مقدار این شمارنده تا پیش از قطع تغذیه هم‌چنان ماندگار است.

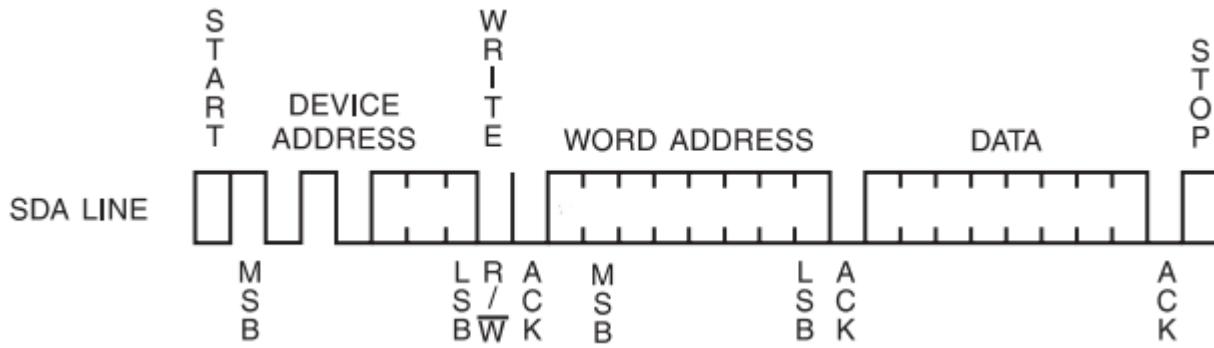
از این رو، برای این‌که بتوان از یک خانه حافظه خواندن باید نخست با انجام یک عملیات خواندن نیمه کاره (Dummy Write)، این شمارنده را دوباره مقدار دهی کرد تا خانه مورد نظر در ماتریس حافظه انتخاب گردد.

عملیات نوشتمن در یک خانه حافظه در بیشترین حالت ۵ میلی ثانیه طول خواهد کشید. (اگر در این مدت و پیش از به پایان رسیدن عملیات درخواست تازه ای برسد، حافظه در پاسخ NACK (Negative Acknowledgement) بر می‌گرداند و در نتیجه عملیات تازه انجام نخواهد شد).

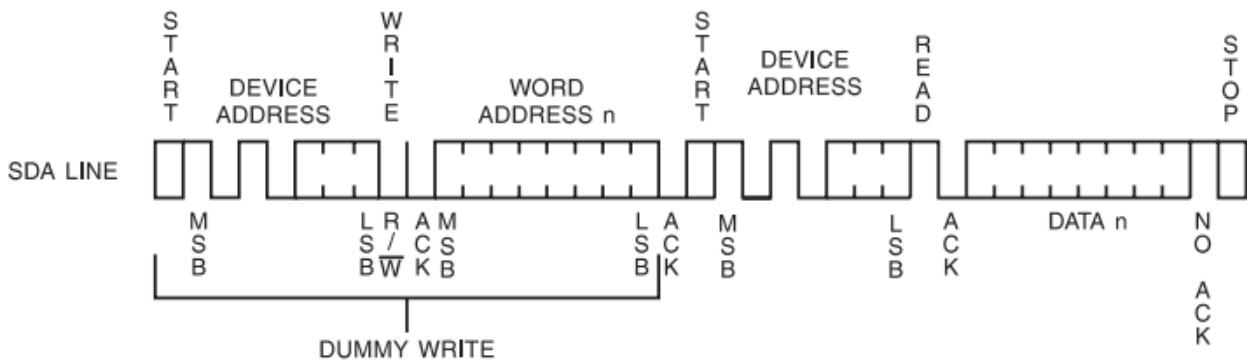
پیام‌های عملیات خواندن و نوشتمن

در زیر دنباله فریم‌های پیام برای عملیات خواندن و نوشتمن بایتی نشان داده شده است:

نوشتمن بایتی



خواندن بایت تصادفی



همانگونه که مشخص است، پیش از درخواست خواندن، یک عملیات نوشت نیمه‌کاره (Dummy Write) انجام می‌شود تا شمارنده آدرس باز مقداردهی گردد.

پرسش: هم خوانی این دنباله فریم‌ها را با پروتکل **TWI** بررسی کنید. (فریم‌های آدرس و داده را مشخص کنید، دستور خواندن یا نوشت

چگونه مشخص می‌شوند؟)

مشخصات تغذیه و فرکانس **SCL**

این قطعه ولتاژ تغذیه ۲,۷ تا ۵ ولت را پشتیبانی می‌کند، به طور کلی بیشینه فرکانس **TWI** که قطعه می‌تواند در آن به درستی کار کند با افزایش ولتاژ تغذیه افزایش می‌یابد. به طور کلی فرکانس 100 KHz را می‌توان برای فرکانس کلک (SCL) در ارتباط **TWI** استفاده کرد.

پرسش: فرکانس کلک در کدام دستگاه پیکربندی می‌شود؟ کلک را کدام دستگاه فراهم می‌کند؟ با توجه به زمان مورد نیاز برای انجام عملیات نوشت، با فرض این که کلک را 10 KHz تنظیم کرده باشیم، در این صورت حداقل با چه نرخی می‌توان عملیات نوشت را انجام داد؟

کتابخانه **Arduino TWI**

کتابخانه **Wire** که یکی از **کتابخانه‌های استاندارد** آردوینو است (از این رو نیازی به نصب آن نیست)، توابعی کاربردی برای کار با واحد **TWI** را فراهم می‌آورد. تابع‌هایی که برای این آزمایش مورد نیاز هستند:

- **begin()**

- `setClock()`
- `beginTransmission()`
- `write()`
- `endTransmission()`
- `requestFrom()`
- `available()`
- `read()`

پرسش: هر یک از تابع‌های نوشته شده را از راه لینک کتابخانه `Wire`، در مستندات آردوینو بررسی کنید و کد لازم را برای تولید دنباله‌ی فریم‌ها برای عملیات نوشت و خواندن گفته شده (با این تابع‌ها) بنویسید.

شرح آزمایش

در این آزمایش می‌خواهیم یک طراحی ساده از یک ماشین لباس‌شویی انجام دهیم. در هر ماشین لباس‌شویی چهار گام کلی ۱- پیش‌شستن با شوینده ۲- شستن با آب ۳- خشک کردن برای شستن لباس‌ها انجام می‌شود. در این آزمایش این که دستگاه در کدام یک از این گام‌ها است را با روشن بودن یک LED نشان می‌دهیم. هر یک از این گام‌ها می‌توانند مدت زمان قابل تنظیم داشته باشند. همچنانی دقت کنید زمان هر یک از گام‌ها باید ذخیره شود که در صورت قطع شدن تغذیه و وصل شدن مجدد آن فرایند شستشو ادامه پیدا کند و از اول شروع نشود.

دستگاه باید مدار کاری پیش‌فرض را بر روی LCD کاراکتری نمایش دهد و هم‌چنین کاربر باید بتواند مدهای کاری دلخواه را با صفحه کلید وارد کند، به گونه‌ای که با قطع تغذیه نیز در دستگاه ذخیره شده بمانند. پس از پایان هر یک از گام‌ها باید فرایند شستشو به گام بعدی منتقل شود پس در صورت قطع تغذیه و وصل شدن آن باید فرایند شستشو را تا گام آخر ادامه دهد.

کاربر باید بتواند از میان مدهای دستگاه یک را انتخاب کرده و توانایی انجام دستورهای آغاز و وقفه را داشته باشد. زمان مانده تا پایان فرایند شستشو باید بر روی LCD کاراکتری نمایش داده شود.

کاربر باید بتواند با دکمه‌ای از صفحه کلید وضعیت شستشو را در وضعیت `hold` قرار دهد و با دوباره فشردن آن فرایند شستشو را ادامه دهد.

پایان فرایند نیز با روشن شدن همه LED‌ها و نمایش عبارت مناسب بر روی نمایش‌گر نشان داده شود و دستگاه هنگامی که کاربر دکمه‌ی مناسب را وارد کرد، به حالت اولیه باز می‌گردد.

همان‌گونه که می‌دانید، شمار عملیات‌های نوشتن بر روی یک EEPROM محدود است و پس از آن، EEPROM کاربری خود را از دست می‌دهد. از این رو همانند حافظه RAM به آن نگاه نکنید و تنها وقتی لازم بود بر روی آن داده بنویسید.



هشدار

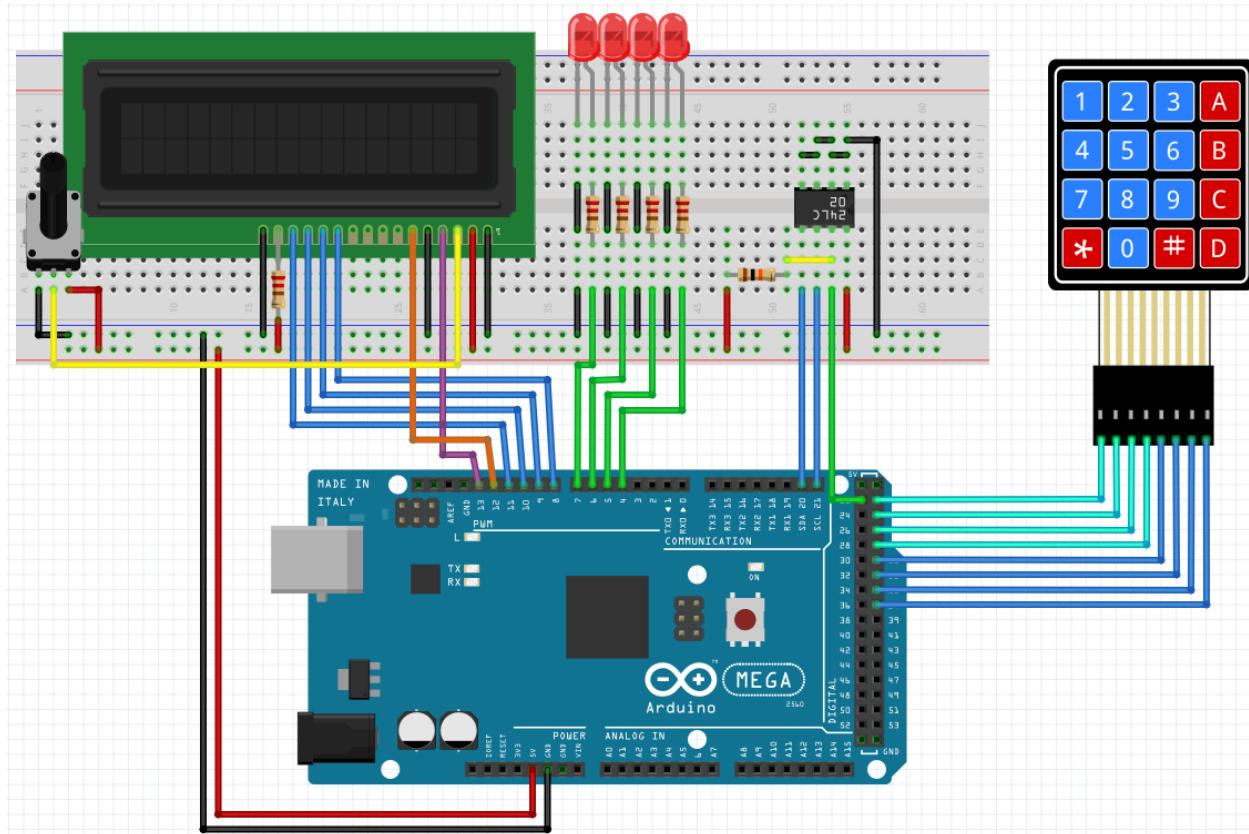
تمرین بیشتر: (تمرین‌های زیر اجباری نیستند، نباید جزوی از آزمایش در نظر گرفته شوند)

- با اسیلوسکوپ موج TWI فرستاده شده به EEPROM را تحلیل کرده و فریم‌های داده و آدرس را مشخص و هم‌خوانی آن‌ها با تنظیمات انجام شده در برنامه را بررسی کنید. در برنامه آدرس سخت‌افزاری EEPROM را تغییر دهید و نتیجه را بررسی کنید.
- طراحی را به گونه‌ای تغییر دهید که دو یا چند EEPROM دیگر نیز به باس متصل شده، داده‌ها دسته‌بندی شوند و بر روی ذخیره خود با متناظر EEPROM.
- در صورت موافق سریست آزمایشگاه، دو گروه بردهای خود را بکدیگر به اشتراک گذاشته و دو میکروکنترلر را با پروتکل TWI به هم مرتبط سازند و انجام یک الگوریتم مانند مرتب‌سازی یک آرایه را میان این دو پخش کرده، سپس خروجی‌ها را با

پک دیگر

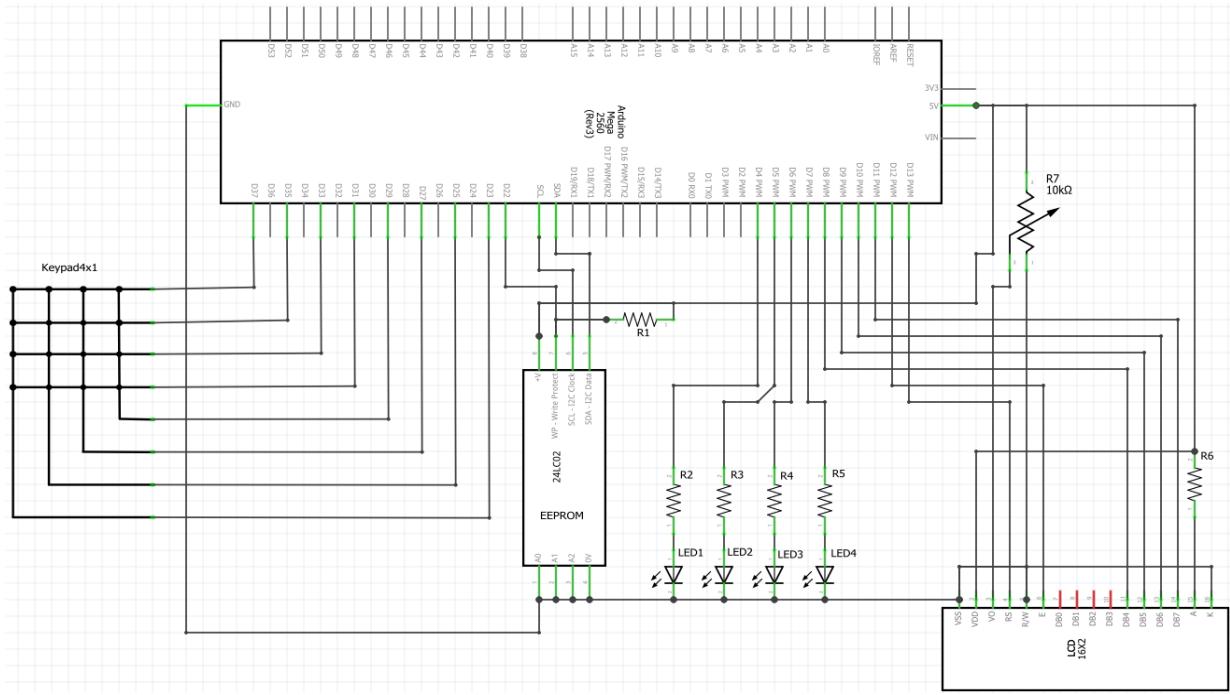
یک پارچه

در صورت موافقت سرپرست آزمایشگاه، دو گروه برد های خود را با یکدیگر به اشتراک گذاشته و دو میکروکنترلر را با پروتکل TWI با یکدیگر مرتبط سازند. کنترل واحد TWI را به روش سرکشی (یا Polling) و همچنین روش وقفه محور (با-Interrupt) بررسی کنید. (راهنمایی: تابع های `onRequest` و `onReceive`) بخشی از رابط کاربری کتابخانه Wire است که به روش وقفه محور واحد TWI را کنترل می کند.)



نمودار مدار آزمایش بر روی برد

کنند.



نمودار شماتیک آزمایش

آزمایش ۸ : اتاق تحت کنترل

هدف آزمایش:

آشنایی با پروتکل SPI

تحلیل موج خروجی آردوبینوی مرکزی (master)

راه اندازی حسگر نور و دما

قطعات مورد نیاز:

- Arduino Mega ۳ عدد برد
- مقاومت متغیر فتوسل (ldr)
- سنسور دمای ۳۵Lm

آنچه باید در پیش گزارش نوشته شود:

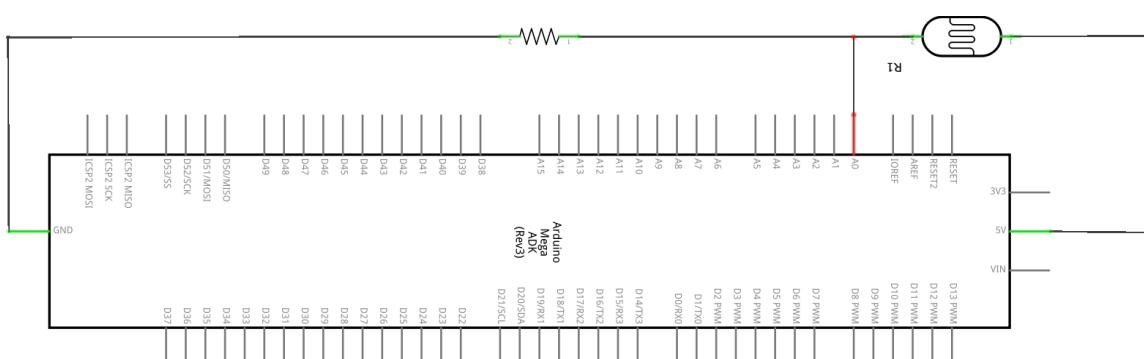
- به پرسش‌های درون مقدمه پاسخ داده شود.

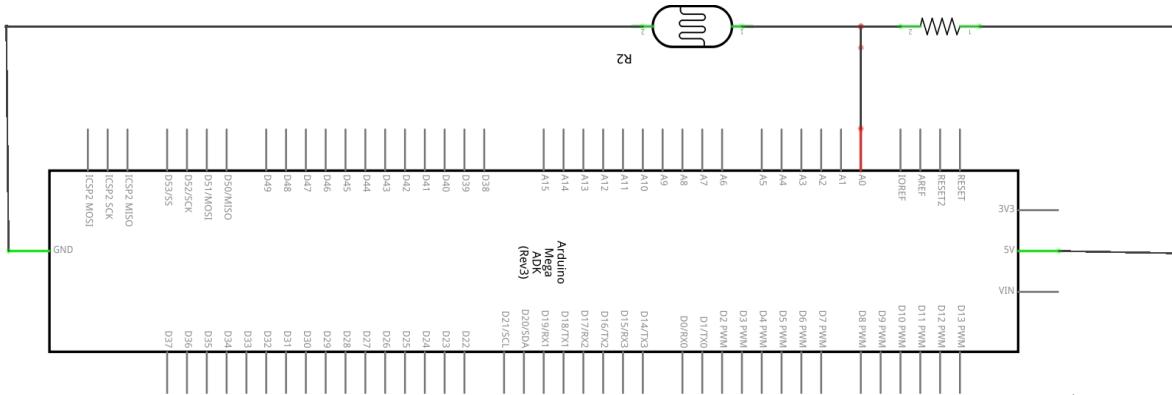
مقدمه

در این آزمایش می‌خواهیم با پروتکل ارتباط SPI بیش‌تر آشنا شویم و از آن برای ارسال میزان نور اتاق و دمای آن استفاده کنیم. این کار در حالت عادی به نظر منطقی نمی‌رسد، زیرا نمایش اطلاعات در همان آردوبینوی مرکزی هم امکان پذیر است. اما این کار زمانی مهم می‌شود که بخواهیم به صورت توزیع شده عملیات پردازشی بر روی این داده‌ها انجام دهیم که با توجه به زمان محدود این آزمایش، پردازشی بر روی این داده‌ها صورت نمی‌گیرد. در ابتدا به نحوه استفاده از مقاومت فتوسل و سنسور ۳۵lm می‌پردازیم و در ادامه پروتکل SPI را بررسی می‌کنیم.

راه اندازی سنسور میزان روش‌نای :

همان‌گونه که در ابتدا گفته شد، می‌توان برای راه اندازی این سنسور از مقاومت متغیر فتوسل بهره برد. این مقاومت با تغییرات میزان نور تغییر می‌کند. این تغییرات با میزان نور رابطه عکس دارد. برای تبدیل تغییرات مقاومت به تغییرات ولتاژ می‌توان از مدار تقسیم ولتاژ بهره برد.





پرسش: در مورد تفاوت دو مدار فوق تحقیق کنید. میزان ولتاژ خروجی هر کدام با تغییرات نور چگونه تغییر می کند.



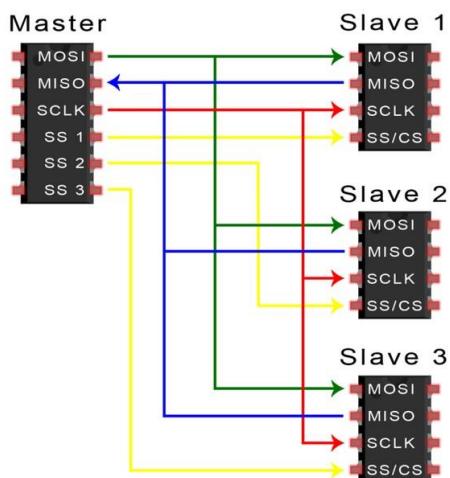
راه اندازی سنسور دما : ۳۵lm

سنسور دما، میزان دمای محیط را بر حسب درجه سانتی گراد به ولتاژ آنالوگ تبدیل می کند.

پرسش: در مورد پایه های آن و همینطور نحوه تبدیل ولتاژ خروجی به میزان دما تحقیق کنید.

راه اندازی ارتباط SPI در حالت چند برده ای :

همان طور که می دانید این نوع ارتباط از نوع ارتباطات (master/slave) است. در این نوع ارتباطات یک دستگاه می تواند با چند دستگاه دیگر ارتباط برقرار کند. در ارتباط SPI بورد مرکزی (master)، بدی که می خواهد با آن ارتباط برقرار کند را انتخاب کرده و برای آن پیام ارسال می کند و در صورت نیاز، از آن درخواست پاسخ می کند.



در این ارتباط می بایست سه پایه MISO (که برای ارسال داده از سوی بورد به بورد مرکزی در نظر گرفته شده) به همراه MOSI (که برای ارسال داده از سوی برد مرکزی به Slave انتخاب شده است) و SCLK (که کلک مرکزی تمام دستگاهها است)، در تمام دستگاه ها یکی شده باشد. از آنجا که تنها یک دستگاه مرکزی (به نام master) وجود خواهد داشت، بنابراین SS که در واقع برای تعیین Slave است، در تمام Slave ها یک پایه خاص است. ولی در دستگاه مرکزی (master) می توان آن را به تعداد Slave تعیین کرد.

پرسش: آیا در پروتکل SPI امکان حضور چند master وجود دارد؟ چه پروتکلی این امکان را به ما می دهد؟

پرسش: در رابطه با ارتباط full-duplex تحقیق کنید. آیا پروتکل SPI از این امکان بهره مند است؟

پرسش: در مورد پایه های SCLK، MISO، MOSI در آردوینو Mega تحقیق کنید. پایه ی پیشفرض برای SS کدام پایه است؟ برای مشاهده آن می توانید به محل نصب آردوینو رفته، مسیر زیر را دنبال نمایید و در انتهای فایل داخل پوشش را باز نمایید:

-> hardware -> arduino -> avr -> variants -> mega

پرسش : در مورد نحوه انتخاب برد **Slave** توسط **SS** تحقیق نموده و نحوه پیاده‌سازی برنامه را برای این که برد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از بردهای **Slave** داده ارسال کند، شرح دهدید. (برای این کار بهتر است نمونه کد‌هایی که برای ارتباط بین دو آردوینو از طریق پروتکل **SPI** در اینترنت موجود است را بررسی نمایید.)

پرسش : مقدار کلاک توسط **Master** تعیین می‌شود یا **Slave** ؟

کتابخانه SPI در Arduino

کتابخانه **SPI** که یکی از **کتابخانه‌های استاندارد آردوینو** است (نیازی به نصب آن نیست)، رابط کاربردی برای کار با واحد **SPI** را فراهم می‌آورد. تابع‌هایی که برای این آزمایش مورد نیاز هستند:

- [begin\(\)](#)
- [setClockDivider\(\)](#)
- [transfer \(\)](#)
- [attachInterrupt\(\)](#)

پرسش: هر یک از تابع‌های نوشته شده را از راه لینک کتابخانه **Wire**، در مستندات آردوینو بررسی کنید.

پرسش: دستور مورد نیاز تا آردوینو در حالت **Slave** قرار گیرد را نوشته و در مورد کارایی آن تحقیق نمایید.

پرسش: تابع **ISR** در کد **Slave** به چه منظور استفاده می‌شود؟ رجیستر مربوط به بایت دریافتی چیست ؟

شرح آزمایش

در این آزمایش قصد داریم با ارتباط **SPI** میزان نور محیط را بر حسب درصد به یکی از دو آردوینو (**Slave**) و مقدار دما را به آردوینو دیگر ارسال کنیم و در هر کدام این مقادیر بر روی نمایشگر سریال نمایش داده شود.

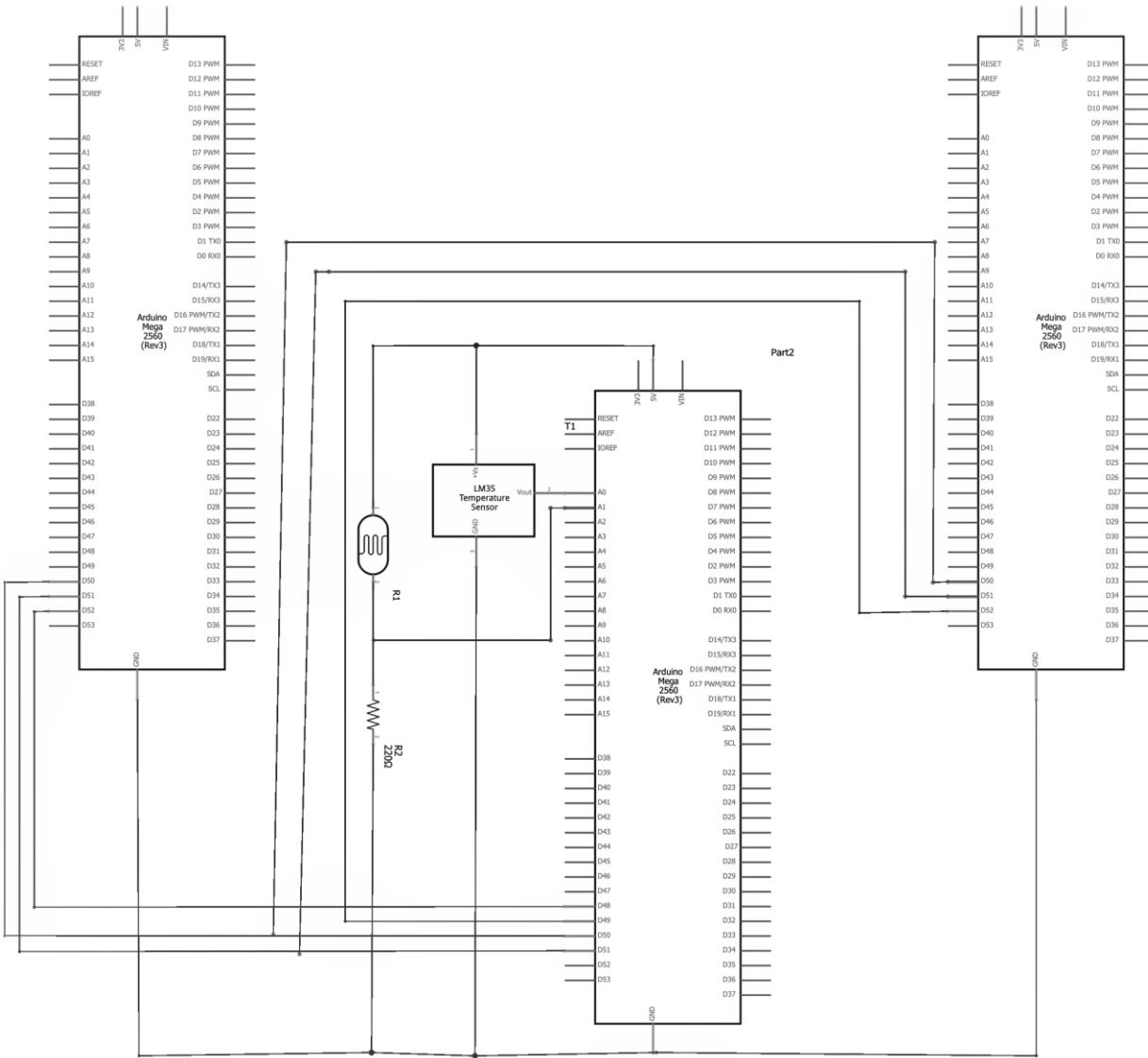
۱. ارتباط میان دو دستگاه آردوینو از طریق **SPI** برقرار نمایید. بدین منظور دو برنامه یکی برای دریافت اطلاعات توسط بورد **Slave** و دیگری برای ارسال اطلاعات از طریق بورد **master** بنویسید. لازم به توضیح است **master** هر ثانیه کلمه اسم و شماره دانشجویی شما را برای بود **Slave** ارسال می‌کند. حتما پایه **SS** در آردوینو **master** را پایه‌ای به جز پایه پیش‌فرض آردوینو قرار دهید.

۲. موج خروجی سه پایه **SS**، **SCLK**، **MOSI** را برای سه مقدار **Clock** توسط اسیلوسکوپ مشاهده کنید.

۳. کد قسمت **Master** و اتصالات آن را به گونه‌ای تغییر دهید که بعد از اضافه کردن یک **Slave** و قرار دادن کد مربوط به برد **Slave**، به طور متناوب و هر ثانیه به آردوینو دوم اسم شما ارسال شود و در ثانیه‌ی بعدی آردوینو اول کلمه "Hello "your name" را دریافت کند.

۴. داده‌های **Hello world** و **Hi** را با اطلاعات مربوط به دو سنسور دما و نور جای‌گزین نمایید. برای خواندن ولتاژ خروجی هر یک از سنسورها کافیست از دستور **analogRead** استفاده نمایید. سپس عدد به دست آمده را به بازه مناسب **map** نمایید.

۵. موج خروجی را برای چهار پایه **SS1**، **SCLK**، **MOSI**، **SS2** توسط اسیلوسکوپ مشاهده نمایید.



آزمایش ۹: موسیقی و header

هدف آزمایش:

آشنایی با عملکرد و نحوه کار با اسپیکرها piezo

استفاده از فایلهای header در برنامه‌نویسی آردوینو

پیاده‌سازی هرگونه موسیقی بر روی میکروکنترلر

قطعات مورد نیاز:

- بورد Arduino Mega
- اسپیکر piezo (در پروتئوس به نام sounder)
- پتانسیومتر
- کلید

آنچه باید در پیش‌گزارش نوشته شود:

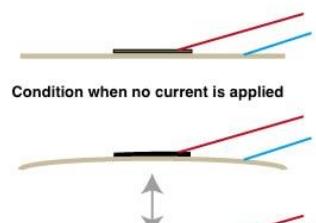
- به پرسش‌های درون مقدمه پاسخ داده شود.

مقدمه:

اسپیکر پیزوالکتریک، از پدیده‌ی پیزوالکتریک (رابطه‌ی بین نیروی الکتریکی و نیروی مکانیکی در بعضی اجسام جامد مثل کریستال‌ها و سرامیک‌ها) استفاده می‌کند تا بدین وسیله صوت تولید کند. این وسیله امروزه در ساعت‌های دیجیتال کوارتز و دستگاه‌های الکترونیک دیگر استفاده می‌شود و برای اسپیکر سیستم‌های ارزان مانند رادیوهای پرتاپل و لوازم خانگی کاربرد دارد. این طراحی نسبت به بقیه طراحی‌های بلندگو بسیار ساده تر کار می‌کند.

پرسش: چند مورد از کاربردهای پیزوالکتریک در دنیای واقعی را نام ببرید.

پرسش: اسپیکر پیزوالکتریک ما چطور کار می‌کند؟ فکر می‌کنید چرا این روش کار انتخاب شده است؟



Condition when no current is applied
When a current is applied, the piezoelectric elements shrink and cause vibrations in the air

دستور **tone** در **Arduino**

`tone(pin_number, frequency_in_hertz, duration_in_milliseconds);`

آرگومان سوم دلخواه است، یعنی می‌توانید `duration_in_milliseconds` را تعیین نکنید که در آن صورت، صدا تا وقتی که `noTone` صدا زده شود ادامه پیدا می‌کند. البته اگر `tone` در حال اجرا بر یک پین دیگر باشد، صدا زدنش هیچ تاثیری ندارد و اگر در حال اجرا بر همان پین باشد، صدا زدنش فرکانس را آپدیت می‌کند. اگر دو اسپیکر `piezo` را به دو پین برد وصل کنید، نمی‌توانند هم‌زمان اجرا شوند؛ باید یکی کارش تمام شود و سپس دیگری شروع به کار کند.

مهمترین نکته در مورد دستور `tone` این است که زمان را با تایمِر داخلی برد می‌سنجد، پس اگر نت‌های مشخص می‌خواهید، بعد از دستور، به همان اندازه `duration_in_milliseconds` یک `delay` قرار دهید.

بررسی: تایمِری که دستور `tone` استفاده می‌کند با خیلی از پین‌های برد مشترک است. بررسی کنید که به چه روش‌هایی می‌توانید آن تایمِر را به هم بزنید که دستور `tone` خراب شود و صدای مطلوب را اجرا نکند.

تئوری موسیقی:

اسپیکر‌های `piezo` برای اجرای فرکانس‌های کمتر از ۳۱ هرتز نیستند. این قضیه مشکل زیادی ندارد، چون گوش انسان از ۲۰ هرتز شروع به شنیدن می‌کند. پایین ترین اکتاو موسیقی (هر بار که تمامی نت‌های موسیقی را از پایین به بالا می‌شمارند یک اکتاو نامیده می‌شود) با فرکانس ۳۲ هرتز شروع می‌شود که دقیقاً بالای حداقل `piezo` است. در زیر، فرکانس‌های نت‌های مختلف است.

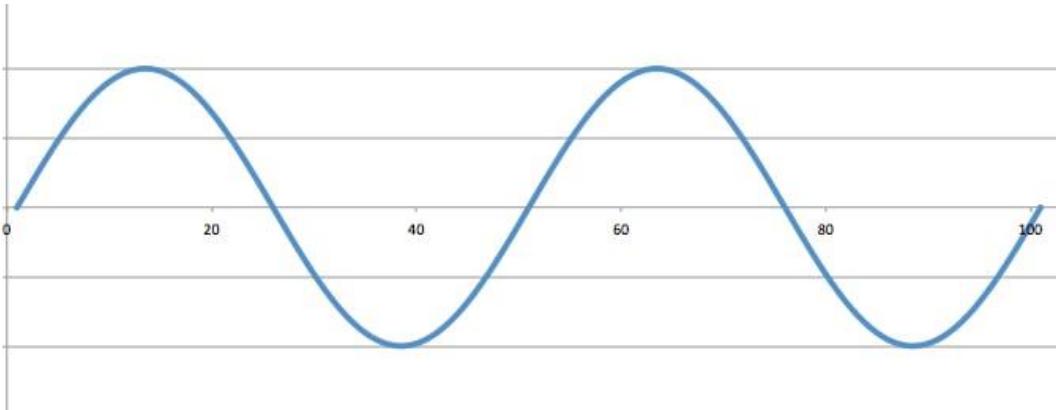
در یک آورده جدول شده

Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz	Note	Hz
C1	32.7	C2	65.4	C3	130.8	C4	261.6	C5	523.3	C6	1046.5	C7	2093.0
C#1	34.6	C#2	69.3	C#3	138.6	C#4	277.2	C#5	554.4	C#6	1108.7	C#7	2217.5
D1	36.7	D2	73.4	D3	146.8	D4	293.7	D5	587.3	D6	1174.7	D7	2349.3
D#1	38.9	D#2	77.8	D#3	155.6	D#4	311.1	D#5	622.3	D#6	1244.5	D#7	2489.0
E1	41.2	E2	82.4	E3	164.8	E4	329.6	E5	659.3	E6	1318.5	E7	2637.0
F1	43.7	F2	87.3	F3	174.6	F4	349.2	F5	698.5	F6	1396.9	F7	2793.8
F#1	46.2	F#2	92.5	F#3	185.0	F#4	370.0	F#5	740.0	F#6	1480.0	F#7	2960.0
G1	49.0	G2	98.0	G3	196.0	G4	392.0	G5	784.0	G6	1568.0	G7	3136.0
G#1	51.9	G#2	103.8	G#3	207.7	G#4	415.3	G#5	830.6	G#6	1661.2	G#7	3322.4
A1	55.0	A2	110.0	A3	220.0	A4	440.0	A5	880.0	A6	1760.0	A7	3520.0
A#1	58.3	A#2	116.5	A#3	233.1	A#4	466.2	A#5	932.3	A#6	1864.7	A#7	3729.3
B1	61.7	B2	123.5	B3	246.9	B4	493.9	B5	987.8	B6	1975.5	B7	3951.1

هر ستون یک اکتاو را نشان می‌دهد که نت‌ها (و بعضًا #‌های نت‌ها) با فرکانس مربوطه شان نشان داده شده‌اند. دقت کنید که نت‌های B و C و نت‌های E و F بین‌شان # ندارند.

برای مطالعه بیشتر درباره اکتاو موسیقی، می‌توانید لینک زیر را مطالعه فرمایید:

<https://www.masterclass.com/articles/music-101-what-is-an-octave#how-is-an-octave-divided>



برای مثال، این فرکانس مربوط به نت A ۴ است. به این معنا که ۴۴۰ هرتز / ۴۴۰ نوسان در ثانیه در هوا و اسپیکر piezo ساخته می شود.

یک ملودی، یک سری نت است که در یک ترتیب خاص و به مدت معلوم نوخته می شود. مثلاً ملودی زیر را در نظر بگیرید:

Jingle Bells

TRADITIONAL
arr. A.L.Christopherson

Piano

این ملودی را می توانیم به عنوان یک سری از نت های نوخته شده بخوانیم. در این مثال به ترتیب EEEEEE GCDE. ولی هر نت ارزش زمانی خاصی دارد که به آن اندازه کشیده می شود. می توانیم نت های توپر (سیاه) را به عنوان استاندارد در نظر بگیریم. عده های نوشته شده اول نت ۴/۴ به ما می گوید در هر جعبه، ما اندازه چهار نت سیاه زمان سپری می کنیم. پس از جعبه اول هم چنین می توان فهمید که هر نت سفید دو برابر یک سیاه کشیده می شود، پس دو سیاه و یک سفید یعنی چهار سیاه. یک جعبه.

ITEM	NOTE	VALUE
Whole note	○	1
Half note	♩	2
Quarter note	♪	4
Eighth note	♫	8
Sixteenth note	♪♪	16

این جدول برای دیدن ارزش‌های زمانی پر کاربرد است. البته این جا نت دایره معيار قرار داده شده است. سفید، یک دوم دایره است. سیاه یک چهارم، چنگ یک هشتم دایره است. همچنین دولاچنگ یک شانزدهم دایره است.

برای نواختن ملودی‌های این درس، یک تکنیک دیگر در نت‌خوانی را باید بلد باشیم. این‌که اگر بخواهیم یک نت را به اندازه یک واحد و نیم بکشیم، باید چه کنیم. یعنی نمی‌خواهیم دو نت به اندازه‌های سفید و نصف سفید بنوازیم، بلکه یک نت را به اندازه یک سفید و نیم نگه داریم. در این‌جا از نقطه استفاده می‌کنیم:

ITEM	NOTE	REST	VALUE (number of beats)
Dotted whole note/rest	○ .	— .	6
Dotted half note/rest	♩ .	— .	3
Dotted quarter note/rest	♪ .	♩ .	1 1/2
Dotted eighth note/rest	♫ .	♩ .	3/4
Dotted sixteenth note/rest	♪♪ .	♩ .	3/8

این جدول همه‌چیز را نسبت به سیاه مقایسه کرده. سیاه یک واحد زمانی دارد. پس سیاه نقطه دار یک و نیم واحد زمانی. سفید نقطه دار سه واحد زمانی و...

هم‌چنان این جدول علامت‌های سکوت را به ما نشان می‌دهد. مثلاً نماد سکوت نت سیاه به این معناست که اگر این علامت را دیدیم، به اندازه مدت زمانی یک سیاه هیچ‌چیزی نمی‌نوازیم. و اگر آن علامت با یک نقطه بود، به اندازه یک سیاه و نیم هیچ‌آهنگی نمی‌نوازیم.



```
#include "pitches.h"
#include "themes.h"
```

فایل‌های هدر:

تاکنون در آردوینو از کتابخانه‌ها استفاده کرده‌ایم. برای LCD و Servo،
و اخیراً هم TWI. به هر حال، ساختن یک فایل هدر در آردوینو خودش
بسیار ساده است. برای اینکه یک فایل جدید کنار فایل اصلی درست
کنید، از دکمه‌ی New Tab یا از میانبر Ctrl+Shift+N استفاده کرده
فایل جدید را با پسوند h ذخیره کنید. اکنون می‌توانید در فایل اصلی خود آن را include کنید.

برای مثال در این آزمایش، دو بخش از قطعات حجیم کد را به جای گذاشتن در فایل اصلی،
به عنوان هدر می‌گذاریم. مثلاً یک فایل pitches.h که در آن نت‌ها و فرکانس‌های مربوطه
را آن‌جا ثبت می‌کنیم، و یک فایل themes.h برای ملودی‌ها. فایل pitches.h در اختیار
شما قرار می‌گیرد (فقط شامل یک سری #define است که اسم نت‌ها را به فرکانس‌شان
مپ می‌کند، همان‌طور که خودتان از جدول فرکانس‌ها در بالا می‌توانید درست کنید)

```
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
```

<pre>int melody[] = { NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5, NOTE_D5, NOTE_G5 };</pre>	<pre>int noteDurations[] = { 4, 4, 2, 4, 4, 2, 4, 4, 3, 8, 1, 4, 4, 4, 4, 4, 4, 4, 8, 8, 4, 4, 4, 4, 2, 2 };</pre>
--	---

برای مثال، در پیاده‌سازی ملودی Jingle Bells که در بالا دیدید، می‌توان برای راحتی کار این دو آرایه را در ذخیره themes.h کنید. آرایه‌ی اول شامل اسم نت‌ها (که بخاطر #define های pitches.h با فرکانس برای tone استفاده می‌شوند) و آرایه دوم شامل ارزش زمانی است. ارزش زمانی بر مبنای نت دایره حساب شده است، و همه‌ی این duration‌ها کسری از دایره هستند. برای مثال ۲ یعنی یک دوم دایره، پس سفید. ۴ یعنی یک چهارم دایره، پس سیاه. ۶ یعنی یک سیاه و نصف، پس یعنی دایره نقطه دار.

نحوه آزمایش

۱. در پروتئوس یک دکمه و یک اسپیکر و یک صفحه کلید را به برد وصل می‌کنیم. برنامه‌ای بنویسید که به هنگام فشردن شدن هر کلید فایل ملودی متناظر با آن را پخش کند.

پرسش: یک اسیلوسکوپ به سیم اسپیکر متصل کنید. چه اتفاق دارد می‌افتد؟

۲. هنگام پخش موسیقی از اسپیکر، پروتئوس با کارت صدا آن را به کامپیوتر شما می‌دهد. به احتمال زیاد هشدار اجرا نشدن کد در ریل تایم می‌گیرید، پس با کلیک راست روی برد، فرکانس کلاک آن را پایین بیاورید تا به جای برسید که موسیقی به نرمی اجرا شود. متوجه می‌شوید که فرکانس نت‌ها هم عوض می‌شود، پس با ایجاد تغییری کوچک در کد، آن را هم رفع کنید.

۳. یک پتانسیومتر به برد وصل کنید. همان‌طور که قبله تابع map ورودی آنالوگ را استفاده کردید، اکنون برنامه ای بنویسید که هر دفعه که دکمه فشرده می‌شود، بنا به وضعیت پتانسیومتر یک نت زیرتر یا بمتر اجرا کند.

۴. با دانشی که دارید، برنامه ای بنویسید که ملوڈی Ode to Joy بتهوون را پخش کند.

Ode to Joy

from the "Choral" Symphony

LUDWIG VAN BEETHOVEN

(1770-1820)

Op.125

arr. A.L. Christopherson

۵. ملوڈی امتیازی: (۱ همان # است ولی بجای یک قدم جلو، یک قدم عقب. □ نت را عادی می‌کند. □ های اول خط تا آخر خط باقی می‌مانند، مگر اینکه با □ عادی سازی شوند)

The Imperial March

Music by
John Williams

آزمایش ۱۱ : پروژه نهایی

هدف آزمایش:

در طول این ترم، شما با مفاهیم اساسی کار با میکرورکنترلر، و ماژول های جانبی پایه ای آن آشنا شدید. برای اختتام این آزمایشگاه و تثبیت دانسته هایتان در یک پروژه نهایی که می تواند تجلای دروس سخت افزار دانشگاه در رزومه تان باشد، همه ی دانش خود را استفاده کنید تا یک سیستم کاربردی تولید کنید.

پروژه ی مورد نظر خود را با مستوی آزمایشگاه خود هماهنگ کنید، و در رابطه با پیچیده بودن آن، ایشان را قانع کنید. و گزنه دوتا از پروژه های پیشنهادی زیر را میتوانید پیاده سازی کنید. توجه داشته باشید که پروژه خلاقانه می تواند به اندازه صلاح حدید استاد، نمره اضافی هم دریغ گردید. استفاده از ماژول های جدید، حتی اتصال میکرو به یک نرم افزار GUI ، وب و یا موبایل، هم برای رزومه خودتان هم بابت نمره اضافی، مزیت محسوب می شود.

رای گیری الکترونیکی

هدف پروژه



چند برد آردوینو که هر کدام با مانیتور و کیبورد یک رای گیری را انجام می دهند و همگی به یک برد مرکزی اطلاعات جمع آوری شده را می فرستند. برد مرکزی هم در یک EEPROM نتایج رای گیری را ذخیره می کند. در دستگاه مرکزی کاربر باید بتواند بین چند مود (لایک و دیسلایک یا بله و خیر، رای دادن از پنچ ستاره، رای دادن تنها به یک نفر، امکان رای دادن به چند نفر) این تغییرات فقط در دستگاه مرکزی قابل اعمال است همچنین باید امکان توقف رای گیری در یک حالت نشان دادن نتایج و سنت کردن یک نوع جدید بدون خاموش کردن دستگاه وجود داشته باشد. دقت داشته باشید که لازم است یک توکن برای هر کاربر وجود داشته باشد تا نتواند دو بار در رای گیری شرکت کند و در صورت شرکت مجدد پیغام "رای شما قبل اثبات شده است" چاپ شود.

قطعات مورد نیاز:

- پنچ عدد بورد Arduino Mega
- کیبورد و LCD کاراکتری
- EEPROM
- بازر و ال ای دی به مقدار کافی

گیتار الکتریکی ۳ سیم

هدف پروژه



بعد از آنکه پیانوی الکتریکی را یاد گرفتید، اکنون نوبت این رسیده که یک گیتار الکتریکی پیاده سازی کنید! از طریق کیبورد و رودی بگرید. دست راست با سه دکمه M K O نواختن سیم M و سیم K را تعیین می کند و دست چپ با دکمه های چپ تر کیبورد، سیم را میگیرد. چالش سطحی، پیاده سازی نت های گیتار روی صفحه کلید است. چالش اصلی، نحوه اجرای چند نت همزمان است. قبل اگفتیم این ممکن نیست، چون فقط یک تایмер استفاده می شود، ولی راه هایی وجود دارد.

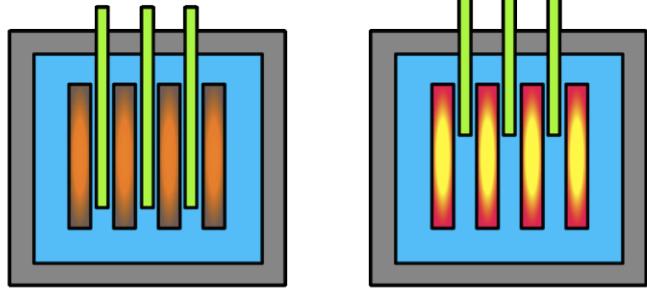
امتیازی: یکسری پتانسیومتر که نقش کوک کردن سیم ها را دارد. هرجه بیشتر چرخانده شود، نت های سیم بم تر و زیر تر می شود.

قطعات مورد نیاز:

- بورد Arduino Mega
- پتانسیومتر
- پیزوالکتریک اسپیکر
- ال ای دی به مقدار کافی

میله های کنترل

هدف پروژه



در این مخزن، یک واکنش حرارت زای تصاعدی اتفاق می افتد. تصاعد و واکنش و تصاعد دما در یک نقطه، باعث خرابی کل مخزن می شود و برای مقابله با آن، هر چقدر میله ها پایین تر بیایند، واکنش کنترل می شود. هدف ما این است که تا جای ممکن واکنش را بدون تخریب مخزن ادامه دهیم.

پس هنگامی که حرارت سنج دمای بالاتر از حدی مشخصی را ثبت کند، سرو و موتور میله های کنترل را پایین می آورد که واکنش از کنترل خارج نشود و هنگامی که دما کم باشد، میله ها را بالا می آورد تا واکنش ادامه پیدا کند. بالا و پایین رفتن میله ها باید نسبی و مناسب با دما باشد.

اگر مخزن در یک بازه ی حرارتی بالای خاص قرار داشته باشد، به ازای زمان سپری شده در آن بازه حرارتی، امتیاز دریافت می کند و این امتیاز همزمان در LCD ای که کنار مخزن قرار دارد آپدیت می شود.

هنگامی که دما بالاتر از یک میزان حساس برود، میله ها بصورت اضطراری تا انتهای رها می شوند، به برد های دیگر در سراسر ساختمان (یعقوب برق، پنکه، هرآنچه در میان ترم پیاده کردید) اطلاع داده می شود که همه فعالیت خود را متوقف کنند و ال ای دی خطر روشن شده و آژیر خطر نیز به صدا در می آید. واکنش که متوقف شد، به برد های دیگر خبر قطع آژیر فرستاده می شود و پس از بالا آمدن میله ها، واکنش دوباره از سرگرفته و آژیر وال ای دی خاموش می شود.

قطعات مورد نیاز:

- حداقل سه بورد Arduino Mega
- حرارت سنج (برای آزمایش، راحت تر و بهتر است که حرارت در نرم افزار شبیه سازی شود)

- تعداد زیادی سرو و موتور
- باز ر و ال ای دی به مقدار کافی
- LCD کاراکتری

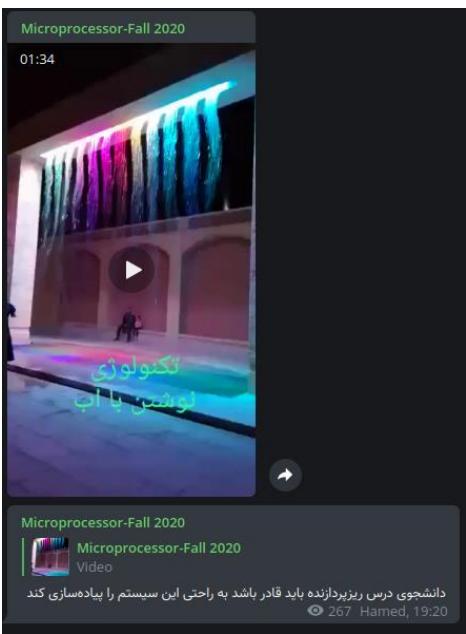
رقص آب (همان چیزی که استاد گفت!)

هدف پروژه

پیاده کردن همان چیزی که استاد گفت! البته به خاطر محدودیت های پروتئوس، به صورت کمی ساده تر.

یک دیوار از LED بسازید. LED های ردیف اول از برد دستور می گیرند و LED های ردیف های بعدی فقط اگر بینند LED بالای روشن بوده و خاموش شده، برای نیم ثانیه روشن می شوند. اینگونه سرشار شدن آب را شبیه سازی کنید. برای مستقل بودن از میکرو LED ها باید با دانش الکترونیک شما برنامه ریزی شوند.

کار اصلی شما این است که از طریق دستگاه ارتباطات سریال برنامه ای بنویسید که هر کاراکتر UNICODE که گرفته شود یا چند کاراکتری که جا شوند را، بصورت سرشار شونده روی LED ها پدیدار کند.



قطعات مورد نیاز:

- بورد Arduino Mega
- باز ر و ال ای دی به مقدار کافی

ربات سوپرمارکت (ربات کارتزین)



هدف پروژه

یک سیستم هایپر مارکت ریاتیک به همراه رابط کاربری مثلا صد عدد کالای مختلف در قفسه هایی که شامل ده ستون و ده ردیف هستند چیده شده اند.

در قسمت رابط کاربری:

ابتدا یک فرم اپلیکیشن اندروید یا وب فرم یا ویندوز فرم طراحی کنید که شامل یک لیست باکس و یک باتن باشد و کاربر نام محصول مورد نظر را بتواند انتخاب کند و یک جعبه متغیر هم برای وارد کردن تعداد کالای درخواستی داشته باشد.

یک دکمه برای افزودن محصول جدید

یک دکمه برای حذف محصول انتخاب شده و یک دکمه هم برای محاسبه قیمت فاکتور در یک جعبه پیام لیست اقلام سفارش داده شده و تعداد و قیمت کل را نمایش می دهد. و با یک دکمه هم اقدام به پایان خرید می نماید.

لیست مربوط به نام اقلام باید به یک جدول پایگاه داده متصل باشد که حاوی فیلد های نام کالا، تعداد موجودی و قیمت کالا باشد که با هر بار خرید هر مشتری به روزرسانی شود.

اما در قسمت مکاترونیک:

کالاهای باید دارای کد منحصر به فرد باشند و هر کالا هم در یک خانه به خصوص از قفسه فروشگاهی قرار دارد.

با وارد کردن کد هر کالا یک سبد خرید توسط یک ربات کارتزین به نقطه مختصات مربوط به کالای درخواستی می‌رود و توسط یک سروو موتور به تعداد کالای مورد نظر سرو و حرکت می‌نماید و پس از پایان خرید به نقطه مختصات صفر و صفر باز می‌گردد و سبد را به مشتری تحویل می‌دهد.

ربات کارتزین دارای دو استپر موتور می‌باشد که یکی از آنها برای حرکت در محور افقی و دیگری برای حرکت در محور عمودی استفاده می‌شود.

قطعات مورد نیاز:

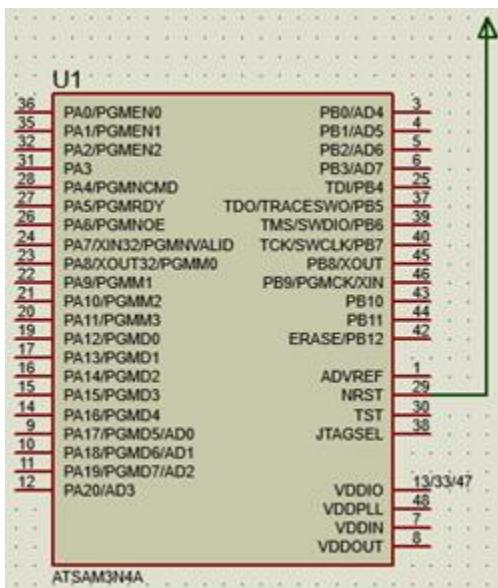
- ۲۵۶۰ Arduino Mega بورد
- استپر موتور
- تعداد زیادی سرو و موتور
- کیبورد و LCD کاراکتری
- بازر و ال ای دی به مقدار کافی

آزمایشگاه اسمنبلی

مقدمه

نکاتی درباره Proteus

- از آنجا که میکروکنترلری که درس بررسی می شود ATSAM3X8E می باشد از این رو باید پروژه ها نیز بر روی همین میکرو انجام شود. اما Proteus از این میکروکنترلر پشتیبانی نمی کند. :
- ولی خوشبختانه خویشاوند های این میکرو را پشتیبانی می کند: (که خانواده ATSAM3N می باشد. و از دید معماری پردازنده Cortex-M) یا آدرس و کارکرد های پریفرال ها، میکرو های این خانواده نزدیکی زیادی با میکروکنترلر درس دارند.
- از این روی توان از این میکروکنترلرها در پروتئوس برای انجام پروژه ها بهره مند شد. مانند ATSAM3N4A در نتیجه دقت کنید در μ Vision نیز پروژه را برای همان میکروکنترلری از خانواده ATSAM3N بسازید که در پروتئوس انتخاب کرده اید.



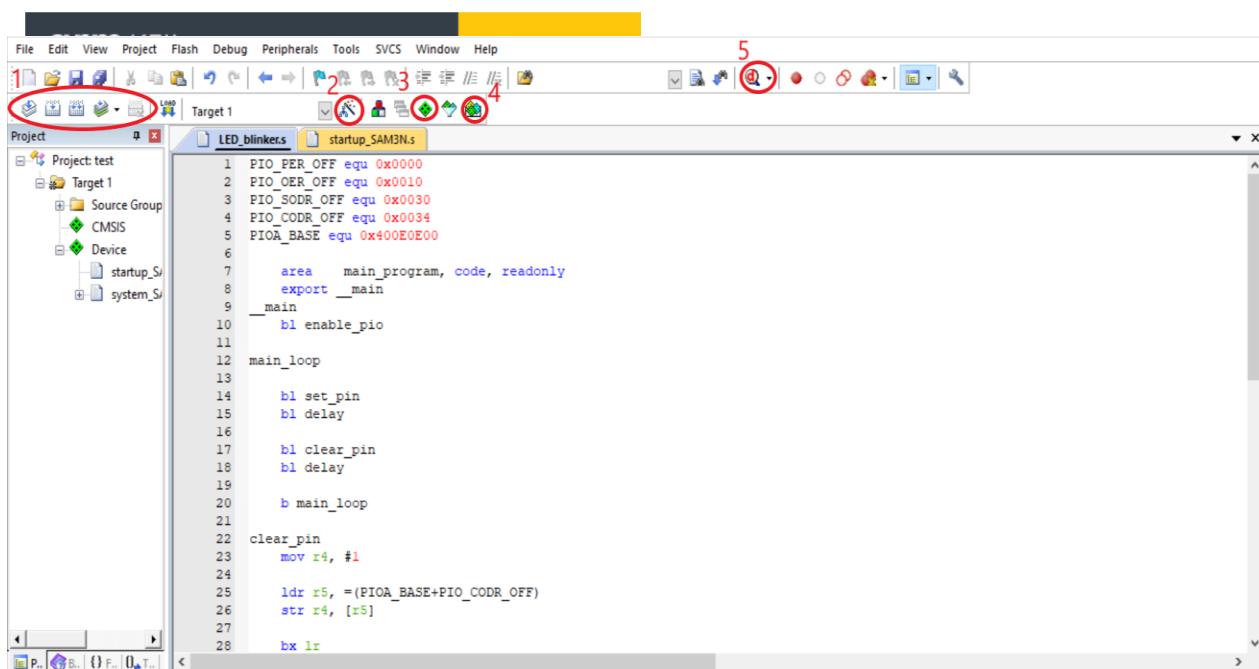
ابزار توسعه keil

پروژه هایی که در زیر آورده شده است باید با زبان اسمنبلی پیاده سازی شود. برای اینکار می توانید از ابزار توسعه Keil μ Vision بهره ببرید. این ابزار محیطی را برای توسعه میکروکنترلرهای سازندگان مختلف مانند: Microchip، LPC، STM32 فراهم میکند که معماری های پردازنده شرکت ARM را در میکروکنترلرهای خود پیاده سازی میکنند.

در این ابزار باید ابتدا پکیج های مربوط به خانواده میکروکنترلری را که در نظر داریم (ATSAM3N)، با ابزار Package Installer در μ Vision دانلود و نصب کنیم. سپس می توان با انتخاب میکروکنترلر دلخواه خود پروژه هایی را به زبان اسمنبلی یا سی برای آن ساخت

که در این صورت µVision کد ها و فایل های لازم برای کامپایل شدن و لینک درست برنامه را به صورت خودکار می سازد. برای دانلود نسخه رایگان این محیط توسعه به این [لينک](#) مراجعه کنید. (نیاز به ثبت نام دارد)

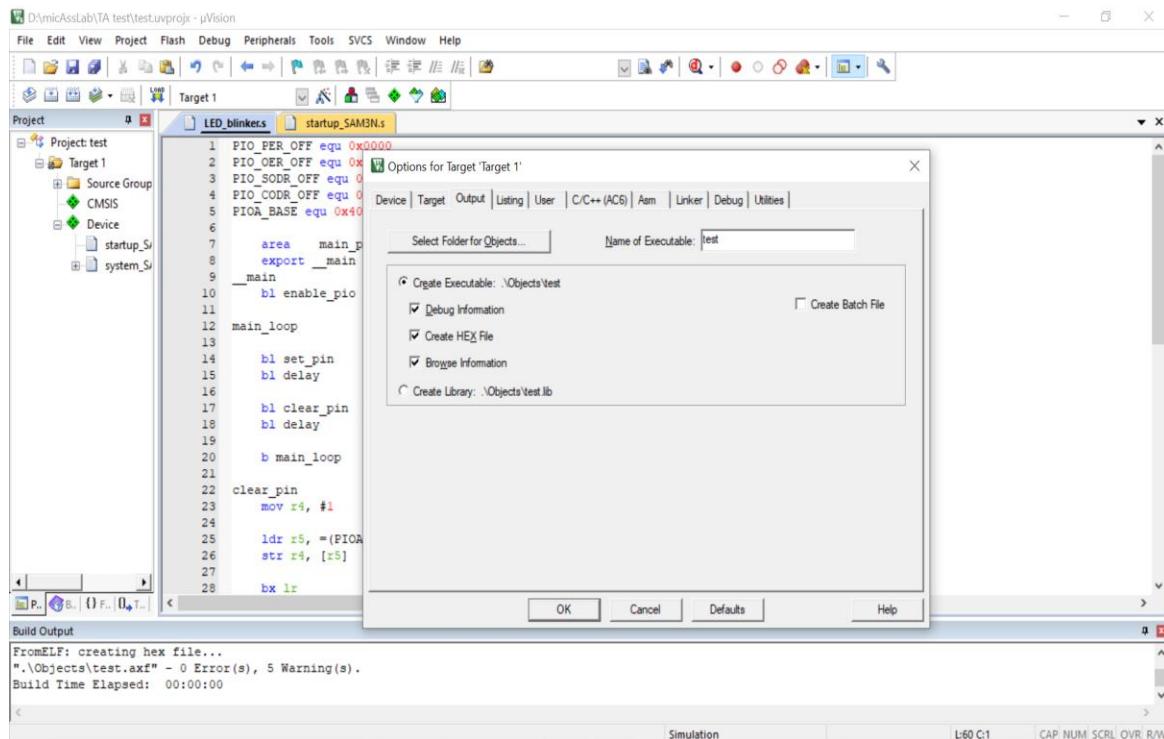
در پیوست فایل های کیل و پروتئوس برای پیاده سازی یک پروژه نمونه برای چراغ چشمک زن با اسمبلی بر روی میکروکنترلر بالا آورده شده است.



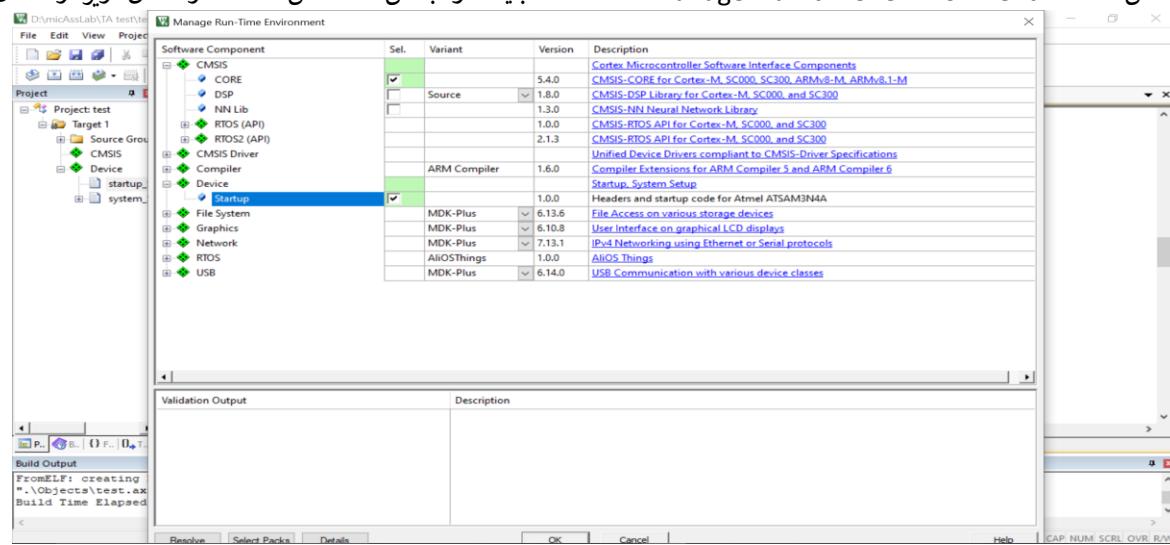
اکنون به توضیح مختصری درباره چند بخش پرکاربرد در ابزار keil میپردازیم:

بخش یک : بخش های translate , build , rebuild , batch build , stop build ابزار هایی که به هنگام ساختن کد مورد استفاده قرار می گیرند هستند پس از تغییر کد از دستور rebuild استفاده می کنیم. **با مطالعه در رابطه با بقیه دستور های گفته شده توضیحات مختصری ارائه دهید.**

بخش دو : option for target است پس از کلیک روی آن پنجره ای مانند شکل زیر مشاهده می کنید که شامل بخش های مختلفی است برای وصل کردن کد اسمبلی به پروتئوس لازم است که همانند شکل در بخش output create HEX file تیک



است . باید دو بخش مشخص شده در شکل زیر را فعال کنید.



بخش چهار: pack installer برای اضافه کردن پک های مختلف استفاده می شود.

بخش پنج : debug session است مثل سایر ابزار های برنامه نویسی نرم افزار keil هم بخش دیباگ را در اختیار ما قرار داده است تا بتوانیم به راحتی کد زده شده را عیب یابی کنیم. امکان اضافه کردن breakpoint و ... نیز وجود دارد.

ما نیاز به کد اسembلی از پیش آمده‌ی startup داریم این کد به منظور تعریف کردن میکروکنترلر استفاده می‌شود و باید با طراحی سخت افزاری ما مطابقت داشته باشد. می‌توانید در رابطه با بخش‌های مختلف این کد با مراجعه به [اینجا](#) اطلاعات بیشتری پیدا کنید. **راجع** به دو بخش **interrupt vector** و **reset-handler** توضیح مختصری ارائه دهید.

آزمایش ۱:

هدف آزمایش:

- آشنایی با شیوه پیکربندی رجیسترها و مدیریت سطح پایین با زبان اسembly

قطعات آزمایش:

- میکروکنترلر ATSAM3N4A
- دیود نورانی LED
- کلید
- مقاومت $\Omega 220$
- مقاومت $K\Omega 10$

آنچه باید در پیش گزارش نوشته شود:

- توضیحات مختصری درباره ی دستورات MOV, LDR و STR دهید.
- ایده ای برای پیاده سازی تابع تاخیر در زبان اسembly ارائه دهید.
- به پرسش ها در بخش مقدمه ای در رابطه با keil که با رنگ قرمز مشخص شده است پاسخ دهید.

مقدمه:

همانطور که در ویدیو آموزشی نیز به آن اشاره شد، برای اجرای برنامه‌ی نوشته شده با استفاده از زبان اسembly نیاز به وارد کردن یک فایل مهم به نام startup.S (import) داریم.

در این فایل همه پیکربندی های آغازی لازم در هنگام روشن کردن و آغاز به کار (Startup) میکروکنترلر پیاده سازی می شود.

این پیکربندی ها دربردارنده سه عمل مهم زیر می باشد :

- پیکربندی پشتۀ (Stack) و Heap
- ذخیره جدول بردار وقفه (Interrupt Vector Table)
- پیاده سازی روال Reset Handler

:startup.s کد بررسی

```
28
29          AREA      STACK, NOINIT, READWRITE, ALIGN=3
30  Stack_Mem     SPACE    Stack_Size
31  __initial_sp
```

در تصویر ابتدا سایز استک با استفاده از دستور EQU به اندازه $0x.....200$ تعریف می شود. (معادل دستور #define مانند) سپس ناحیه مربوط به STACK با استفاده از تگ AREA مشخص می شود(دقت کنید که این ناحیه از نوع READWRITE می باشد، یعنی علاوه بر خواندن، اجازه ای نوشتن نیز وجود دارد). در آخر نیز با استفاده از SPACE برای حافظه رزرو می شود.

```

39
40           AREA      HEAP, NOINIT, READWRITE, ALIGN=3
41   __heap_base
42   Heap_Mem      SPACE     Heap_Size
43   __heap_limit

```

خطوط بالا مربوط به ناحیه `heap` و اختصاص دادن حافظه به آن می باشد که بسیار مشابه کد `stack` می باشد با این تفاوت که با آدرس شروع و پایان ناحیه `heap` `__heap_limit` و `__heap_base` استفاده از `__heap_limit` می شود.

```

53           EXPORT  __Vectors
54
55   __Vectors    DCD    __initial_sp          ; 0: Top of Stack
56           DCD    Reset_Handler         ; 1: Reset Handler
57           DCD    NMI_Handler          ; 2: NMI Handler
58           DCD    HardFault_Handler    ; 3: Hard Fault Handler
59           DCD    MemManage_Handler    ; 4: MPU Fault Handler
60           DCD    BusFault_Handler    ; 5: Bus Fault Handler
61           DCD    UsageFault_Handler ; 6: Usage Fault Handler
62           DCD    0                 ; 7: Reserved
63           DCD    0                 ; 8: Reserved
64           DCD    0                 ; 9: Reserved
65           DCD    0                 ; 10: Reserved
66           DCD    SVC_Handler        ; 11: SVCall Handler
67           DCD    DebugMon_Handler   ; 12: Debug Monitor Handler
68           DCD    0                 ; 13: Reserved
69           DCD    PendSV_Handler    ; 14: PendSV Handler
70           DCD    SysTick_Handler   ; 15: SysTick Handler
71

```

در این قسمت interrupt vector table را مشاهده می کنیم. در اولین خط آن `initial_sp` که حاوی بالاترین آدرس `stack` می باشد. هنگامی که میکرو شروع به کار کردن می کند رجیستر `sp` بوسیله `initial_sp` مقدار دهی می شود. دومین خط این قطعه کد آدرس وقفه `reset` می باشد که به تابع `reset_handler` اشاره می کند که در ادامه به آن می پردازیم.

```

112           EXPORT  Reset_Handler
113           IMPORT  SystemInit
114           IMPORT  __main
115           LDR    R0, =SystemInit
116           BLX    R0
117           LDR    R0, =__main
118           BX    R0
119           ENDP

```

وظیفه ای اصلی تابع `reset_handler`، تنظیم کلاک سیستم و در نهایت پریدن به تابع `main` برنامه‌ی ما می‌باشد.

شرح آزمایش:

این آزمایش شامل ۳ عدد LED و ۲ دکمه می‌باشد. در ابتدا LED‌ها خاموش می‌باشد. هنگامی که دکمه‌ی اول فشرده می‌شود. هر ۳ LED شروع به چشمک زدن می‌کنند با این تفاوت که سرعت چشمک زدن LED‌ها از چپ به راست افزایش پیدا می‌کند. با فشردن دکمه‌ی دوم همه‌ی LED‌ها خاموش می‌شوند.

آزمایش ۲ (مخصوص ترم‌های حضوری)

هدف آزمایش:

فراخوانی اسمنبلی با استفاده از روش `inline`

قطعات مورد نیاز:

arduino due •

آنچه باید در پیش گزارش نوشته شود:

- درباره‌ی مزایای استفاده از inline assembly تحقیق کنید.

در آزمایش قبل با مدیریت سطح پایین در زبان اسمنلی آشنا شدیم. در این آزمایش قصد داریم که با نوع دیگری از کاربردهای زبان اسمنلی یعنی فراخوانی با استفاده از روش inline نیز آشنا شویم.

گاهی اوقات نیاز داریم که از دستورات اسمنلی در زبان‌های سطح بالا استفاده کنیم. یکی از روش‌ها برای انجام این کار این است که مانند آزمایش قبل درون یک فایل جدا توابع اسمنلی را بنویسیم و سپس با استفاده از دستور "C extern آن توابع را وارد کد C++ خود کنیم. یکی از معایبی که این روش دارد، ایجاد سریار زیاد است به همین دلیل سراغ روش دوم یعنی inline assembly می‌رویم.

گاهی اوقات کامپایلر بهینه سازی‌هایی را در هنگام اجرای کدا انجام می‌دهد که باعث پاک شدن بلاک‌های برنامه می‌شود. برای آنچه باشد که این روش در کامپایلر بهینه سازی را در هنگام اجرای کدا انجام می‌دهد، کافی است که دهده تا بتوانیم با استفاده از کلید واژه‌ی __asm از دستورات اسمنلی بطور در

زبان‌های C/C++ استفاده کنیم.

گاهی اوقات کامپایلر بهینه سازی‌هایی را در هنگام اجرای کدا انجام می‌دهد که باعث پاک شدن بلاک‌های برنامه می‌شود. برای آنچه باشد که این روش در کامپایلر بهینه سازی را در هنگام اجرای کدا انجام می‌دهد، کافی است که دهده تا بتوانیم با استفاده از کلید واژه‌ی __asm volatile آن را نوشت.

```
__asm volatile ("ADD R0, R1, R2");
```

تصویر- ۱ نحوه‌ی استفاده از __asm

در صورتی که بخواهیم درون بلاک __asm از چند دستور اسمنلی استفاده کنیم باید مانند تصویر ۲ در انتهای هر خط \n\t قرار داد.

```
__asm volatile (
    "ADD R0, R1, R2\n\t"
    "SUB R1, R2, R3\n\t");
```

تصویر - ۲ نحوه‌ی inline کردن بیش از ۱ دستور

گاهی اوقات می‌خواهیم بلاک اسمنلی ما با متغیر‌هایی که در C تعریف می‌کنیم تعامل داشته باشد، بطور مثال ورودی دریافت کند، پردازش بر روی آن داده‌ها انجام دهد و آن را در خروجی از پیش تعریف شده ذخیره کند. در این صورت بلاک اسمنلی شکل زیر را به خود می‌گیرد:

```
__asm [volatile] (
    Assembly instruction/s
```

:Output operand list لیستی از عملوند های خروجی و متغیر معادل آن در زبان C
 :Input operand list لیستی از عملوند های ورودی و متغیر معادل آن در زبان C
 :Clober list لیستی از رجیستر های که نمی خواهیم در هنگام اجرای برنامه به عنوان عملوند ورودی/خروجی استفاده شوند.
);

لیست های بالا اختیاری می باشند و در صورتی که به آن ها نیاز نداشته باشیم، می توانند خالی بمانند.

در علاوه بر نام رجیستر ها، از دو رشته‌ی خاص "memory" و "cc" نیز می توان استفاده کرد. در مورد این دو رشته تحقیق کنید و نتیجه‌ی آن را در پیش گزارش بنویسید.

```
int add(int i, int j)
{
    int res = 0;
    __asm (
        "ADD %[result], %[input_i], %[input_j]"
        : [result] "=r" (res)
        : [input_i] "r" (i), [input_j] "r" (j)
        : "r5", "r6", "cc", "memory"
    );
    return res;
}
```

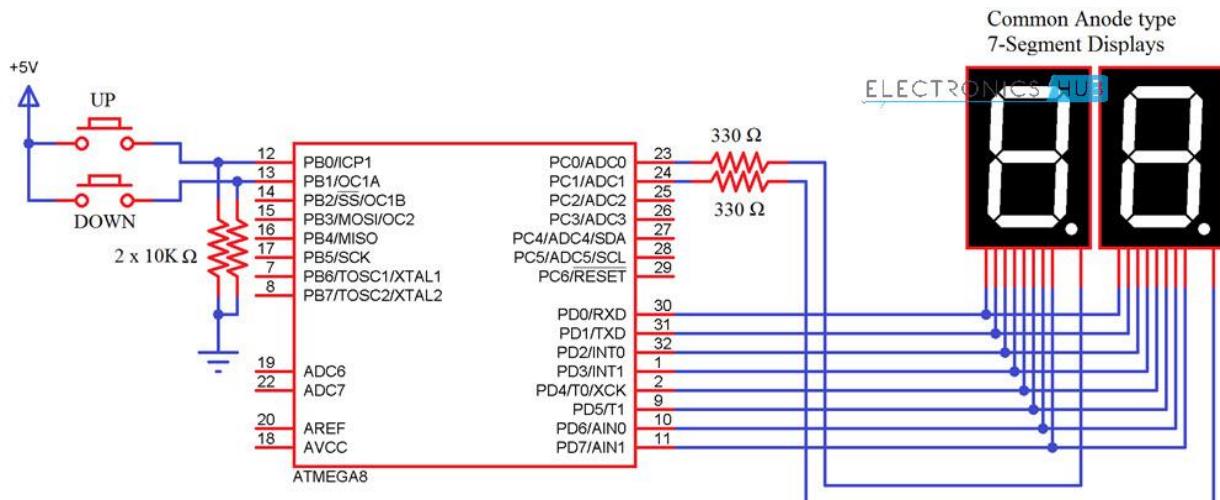
شرح آزمایش:
 با استفاده از روش inline تابع بازگشتی فاکتوریل را پیاده سازی کنید و نتیجه‌ی آن را بر روی صفحه نمایش کاراکتری نمایش دهید.

پروژه های اسمبلی

ثانیه شمار

در این پروژه شما باید با وقفه شمارنده Systick و دو سون سگمنت (Seven-segment) برای نمایش ثانیه های شمرده شده از ۰ تا ۵۹، ثانیه شماری را به زبان اسمبلی پیاده سازی کنید که هر زمان کلیدی فشرده شود، شمارش از ۰ آغاز گردد و هر زمان کلید رها شود شمارنده بایستد ولی مقدار سون سگمنت ها نباید پاک شود و تنها در زمان فشردن کلید مقدار نمایش داده شده با سون سگمنت ها بازنشانی خواهد شد.

بررسی سطح منطقی پایه PIO باید با وقفه انجام شود (Polling) و نه سرکشی (Interrupt-Driven).

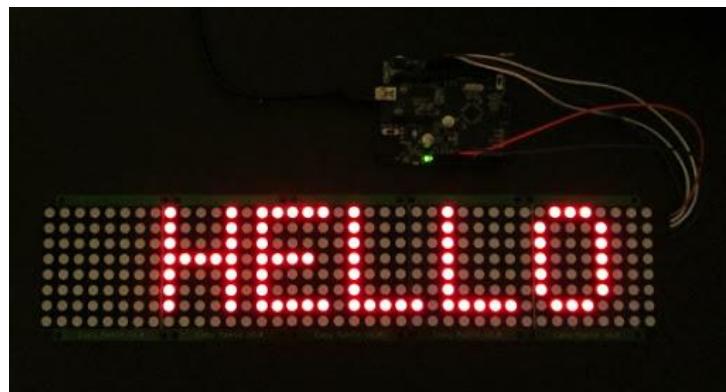


شماتیک از شیوه اتصال دو سون سگمنت به یک میکروکنترلر

تابلو روان

در این پروژه باید با بهره گیری از چند ماتریس LED (در [تابلو روان](#) را پیاده سازی کنید. بر روی تابلو روان HELLO نمایش داده می شود (در سمت چپ ترین ماتریس) که هر ثانیه چند خانه به سمت راست می رود. و رفته رفته از راست ترین سمت ماتریس ناپدید می شود و دوباره از چپ ترین سمت ماتریس نمایش داده می شود.

یکی از ماتریس های LED در پروتئوس MATRIX-8*8-GREEN می باشد. ممکن است به دلیل محدودیت در پایه های میکرو یا ماتریس LED، روشن کردن همزمان همه LED های نمایانگر HELLO ممکن نباشد برای حل این مسئله باید در هر لحظه تنها LED را روشن نگه دارید و سپس چند LED دیگر را. اگر این کار به اندازه کافی تند انجام شود. چشم انسان توانایی تشخیص پیوسته نبودن روشنایی ها را ندارد و نتیجه آن می شود که همه LED ها را روشن میبینیم. به این روش [Multiplexed Display](#) گفته می شود.



نمایی از چند ماتریس به هم متصل شده که بر روی آن ها واژه Hello نمایش داده شده است

