



دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

دستور کار آزمایشگاه ریزپردازنده

(بهار ۱۴۰۰)

مؤلفان:

بهداد منصوری

علیرضا صالحی

با تشکر از:

امیرحسین شیخ‌الاسلامی - امیرحسین صفاتی

نرگس سدیفی - ارشیا رحیمی

فهرست مطالب

1	قوانين آزمایشگاه ریزپردازنده
7	آزمایش 1 : آشنایی با برد Arduino MEGA2560 و برنامه های مورد نیاز
10	آزمایش 2 : کیبورد ورودی و ارتباطات سریال
16	پیش نیاز اسمنلی
	آزمایش 3 : آزمایش یک اسمنلی
17	آزمایش 4 : راه اندازی سروو موتور و ورودی آنالوگ
21	آزمایش 5 : راه اندازی رله با Arduino MEGA2560
27	آزمایش 6 : نیم پروژه
31	آزمایش 7 : ماشین لباس شویی ساده
38	آزمایش 8 : اتاق تحت کنترل
42	آزمایش 9 : موسیقی و header
49	آزمایش 10 : پروژه نهایی
53	پیش نیاز پروژه های اسمنلی
54	پروژه های اسمنلی

قوانين آزمایشگاه ریزپرداز نده

به منظور افزایش کارایی درس آزمایشگاه ریزپرداز نده، 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:

1. مدرسین و دانشجویان موظفند راس ساعت در کلاس آنلاین حاضر شوند.
2. در تمام مدت زمان کلاس دانشجویان موظفند آنلاین بوده و بنا به صلاحیت مدرسین وبکم و میکروفون خود را فعال کرده و همچنین صفحه سکتابخانه خود را به اشتراک بگذارند.
3. غیبت در هیچ یک از جلسات کلاس‌های آنلاین به هیچ عنوان مجاز نیست.
4. اگر دانشجویانی نتوانند در طی یک جلسه آزمایش موردنظر را تمام کنند، در طول هفته موظفند تا آن آزمایش را تکمیل کرده و پیش از آغاز جلسه بعد نتیجه را به مدرس خود تحویل دهند. حداکثر تعداد جلساتی که به درازا کشیده می‌شود ۱ جلسه در طول نیمسال است. دانشجو برای تکمیل آزمایش نمی‌تواند به گروههای دیگر ملحق شود.
5. آزمایش‌ها در گروههای تک نفره انجام می‌شوند و میزان فعالیت افراد تعیین کننده نمره نهایی آن‌ها خواهد بود.
6. هر آزمایش شامل یک پیش‌گزارش باید به صورت دستنویس و پیش از شروع آزمایش‌ها از طریقی که مدرس آزمایشگاه اعلام کرده است به ایشان تحویل داده شود. پیش‌گزارش مناسب برای هر آزمایش در دستور کار آمده است.
7. ارائه گزارش کار آزمایش‌ها الزامی نیست و تحویل خروجی صحیح به مدرس کفایت می‌کند.

نکات امنیتی:

1. 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؛ چرا که در هنگام انجام آزمایش، وقت کافی برای توضیح و یادگیری قسمت تئوری وجود ندارد.
2. 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 تحویل می‌دهند (چه به صورت شفاهی و چه به صورت کتبی)، برای خود نیز یادداشت بردارند تا بعد از اتمام کلاس‌ها دچار مشکل نشونند.
3. 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 100 تا 330 اهمی سری نموده، و سپس وصل کنید.

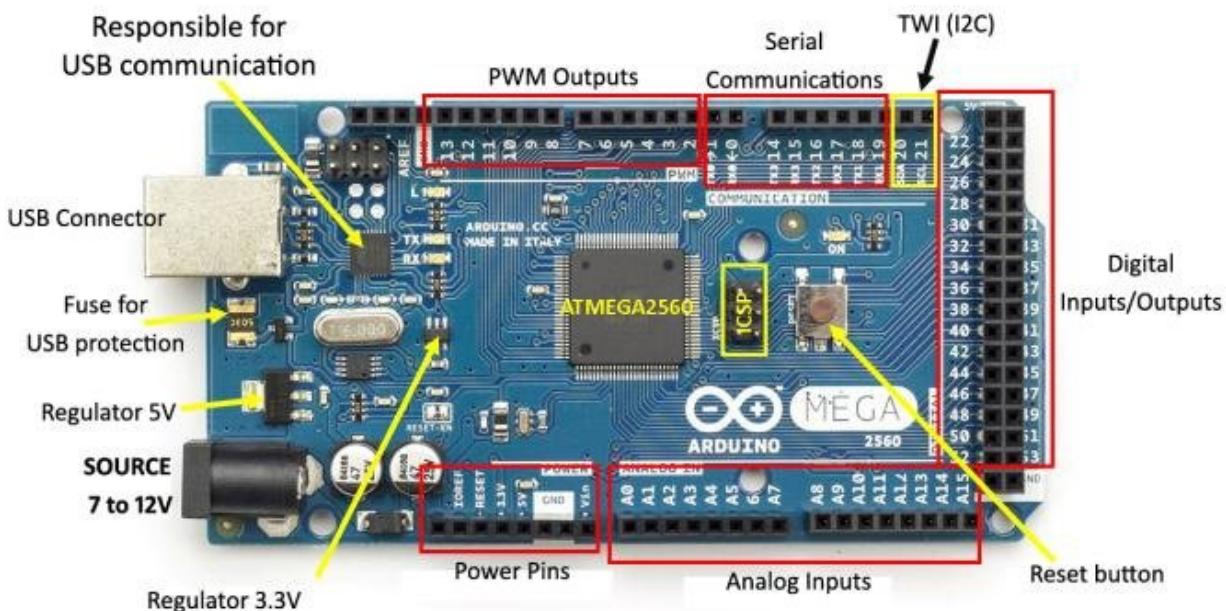
آشنایی با برد Arduino MEGA 2560

برد Arduino MEGA 2560 یک میکروکنترلر بر پایه ATmega2560 از شرکت Atmel است. این برد دارای 54 پین دیجیتال ورودی/خروجی (که 15 پین می‌تواند به عنوان خروجی PWM استفاده شود)، 16 ورودی آنالوگ، 4 پورتUARTs (پورت‌های سریال SXT-Aفزاری)، یک کلاک 16 مگاهرتزی، یک کابل ارتباطی USB، یک پاور جک، یک ICSP header و یک دکمه ریست است. این برد عملاً همه‌ی 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 2560 می‌تواند با اکثر شیلد‌های طراحی شده برای Uno و همچنین بردهای قدیمی Duemilanove or Diecimila برنامه‌ریزی می‌شود که نحوه نصب و اتصال برد به آن در پیوست آورده شده است.

مشخصات:

ATmega2560
5 ولت
7 تا 12 ولت
20-60 ولت
15 تای آن خروجی PWM تولید می‌کنند.
16 عدد
20 میلی‌آمپر
50 میلی‌آمپر
256 کیلوبایت که 8KB برای bootloader است.
8 کیلوبایت
16 مگاهرتز

- میکروکنترلر:
- ولتاژ عملیاتی:
- ولتاژ ورودی (پیشنهادی):
- دامنه مجاز ولتاژ ورودی:
- پین‌های دیجیتال ورودی/خروجی:
- پین‌های ورودی آنالوگ:
- جریان DC هر پین 3.3 ورودی/خروجی:
- جریان DC هر پین 5 ولت:
- حافظه فلاش:
- SRAM
- سرعت ساعت:



برد آردوینو Mega2560

نیشان‌گرها: LED

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

LED‌های TX و RX: بر روی برد دو بخش به نام‌های TX (ارسال) و RX (دریافت) وجود دارد. یکی در بخش پایه‌های 0، 1 و 14 تا 19 که برای ارتباط سریال هستند و دومی LED‌های TX و RX. چراغ مربوط به TX هنگام ارسال داده سریال مناسب با میزان سرعت چشمک می‌زند و چراغ مربوط به RX نیز هنگام دریافت داده چشمک می‌زند. میزان سرعت چشمک زدن به baud rate برد بستگی دارد.

تغذیه:

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

منبع تغذیه خارجی غیر از USB می‌تواند آداپتور AC به DC یا باتری باشد. آداپتور (با سوکت‌های center-positive به قطر 2.1 میلی‌متر) می‌تواند به پاورجک موجود بر روی برد متصل شود و سیم‌های باتری می‌توانند مستقیماً وارد پین‌های GND و Vin شوند.

برد می‌تواند با منبع تغذیه خارجی 6 تا 20 ولت کار کند. اگر ولتاژ منبع تغذیه پایین‌تر از 7 ولت باشد روی ولتاژ پین‌ها اثر خواهد گذاشت و ممکن است ولتاژ خروجی آن‌ها کمتر از 5 ولت شود و حتی نوساناتی را به وجود آورد. ولتاژ بیش از 12 ولت نیز می‌تواند موجب افزایش دمای رگولاتور و در نتیجه آسیب به برد گردد. ولتاژ پیشنهادی مناسب، بین 7 تا 12 ولت است.

پین‌های 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 پا اتصال USB با 5 ولت) از آن استفاده می‌شود و چنان‌چه برد از طریق پاورجک به منبع تغذیه وصل شده باشد، می‌توانید از طریق این پین (به عنوان خروجی) به ولتاژ منبع تغذیه دسترسی داشته باشید.

- 5V: این پین یک ولتاژ تنظیم شده 5 ولت را از طریق رگولاتور موجود بر روی برد فراهم می‌کند. برد می‌تواند از طریق پاورجک (12-7 ولت) DC، پورت USB (به اندازه 5 ولت) و یا پین V_{IN} برد (12-7 ولت)، تغذیه گردد. ولتاژ پین‌های 5 ولت و 3.3 ولت از رگولاتور عبور می‌نماید و استفاده از ولتاژ این پین‌ها ممکن است باعث صدمه دیدن برد شود، از همین رو، استفاده از این پین‌ها توصیه نمی‌شود.

- 3.3V: یک ولتاژ 3.3 ولتی، به وسیله‌ی رگولاتور روی برد فراهم می‌گردد که حداقل جریان آن 50 میلی‌آمپر است.
- GND: پین‌هایی که با اتصال به زمین، ولتاژ صفر را فراهم می‌کنند.
- IOREF: این پین میزان ولتاژ مرجعی که میکروکنترلر با آن کار می‌کند را مشخص می‌نماید. این پین اجازه می‌دهد 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، می‌تواند مقدار ولتاژ را از پین IOREF خوانده، منبع تغذیه مناسب خود را انتخاب نماید و یا این که مبدل‌های ولتاژ را برای کار کردن با ولتاژ‌های 5 یا 3.3 ولت، بر روی خروجی‌ها فعال نماید. این قابلیت، به شیلد‌ها امکان می‌دهد تا با برد 3.3 ولتی همچون DUE و برد‌های AVR-based که با ولتاژ 5 ولت کار می‌کنند، 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 ARM و... فراهم میکند. این فریم ورک API یکسانی را به ازای میکروکنترلرهای متقاول 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 میکروکنترلرهای مختلف با سخت های افزار های متقاول اجرا شود.

در پروژه آردوینو می توان زبانهای C، C++ و Assembly را به کاربرد. برای نمونه یک روال را به زبان اسمبلی نوشت 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 میکروکنترلرها به دلیل منابع سخت افزاری محدود، همه قابلیت های زبان C++ پیاده سازی نشده است.

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

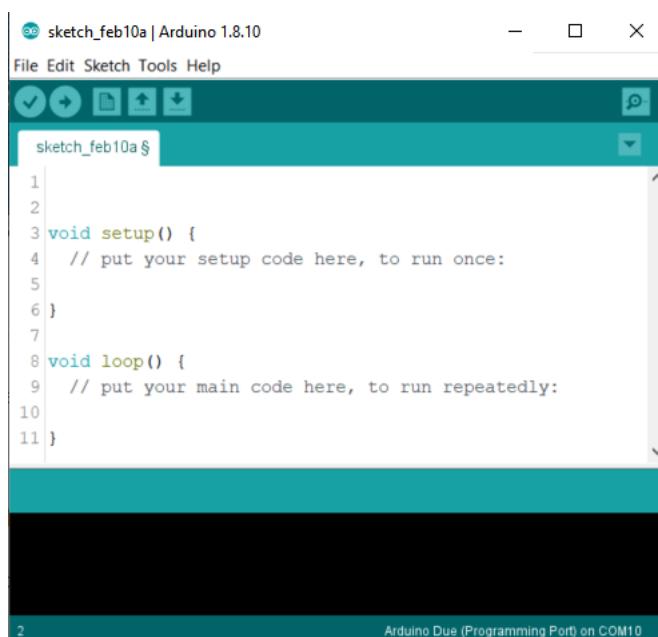
<https://github.com/MicroprocessorAUT/Lab/tree/main/Samples>

- پروژه های نمونه ای برای نشان داده شیوه استفاده از کلاس های C++ و ساختاربندی مناسب کد در آدرس بالا آورده شده است. از آنجا که رعایت این ساختاربندی در تحويل 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.



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. (پروگرم کردن میکروکنترلر)
- New: کلید میانبر برای ایجاد یک پروژه جدید.
- Open: کلید میانبر 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 خواهد کرد. چند ثانیه صبر نمایید، در زمان آپلود دو عدد **Led** با نامهای **RX** و **TX** روی برد چشمک خواهد زد. در صورتی که آپلود برنامه با موفقیت انجام شود، در نوار وضعیت، پیغام "Done Uploading" را خواهد دید.

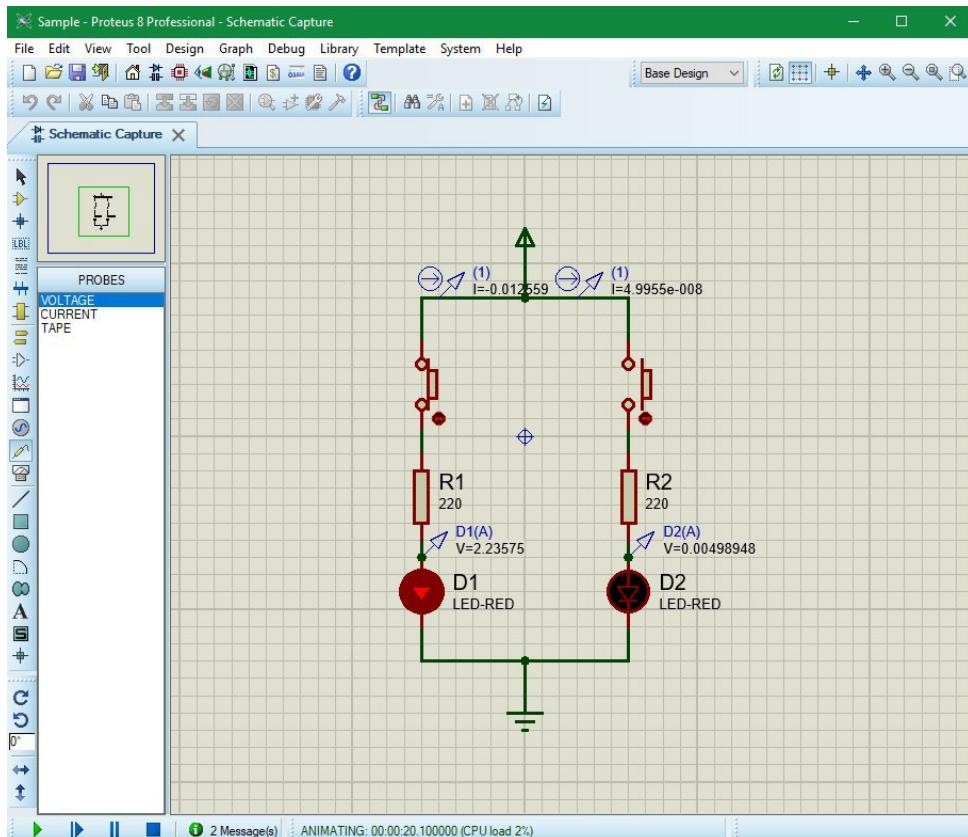
برنامه‌نویسی در آردوینو 3 بخش کلی دارد:

1. پیش‌پردازش‌ها: در بخش پیش‌پردازش، کتابخانه‌ها و یا متغیر‌هایی که محلی نیستند تعریف و فراخوانی می‌شوند.
2. حلقه **setup**: در این بخش کارهایی که باید یکبار انجام شوند را تعریف می‌کنیم. به طور مثال، تنها کافی است یکبار تعریف کنیم که یک پین دیجیتال ورودی باشد یا خروجی و یا این‌که به برد اطلاع دهیم که می‌خواهیم از پورت سریال استفاده کنیم.
3. 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 نمایشگر نشان دهد، عمل خواندن دما از سنسور و نوشتن 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 Binary** از تب **Sketch** می‌توانید فایل باینزی کامپایل شده برای پرینت‌رول 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، آزمایش‌ها در بستر شبیه‌سازی Proteus انجام خواهد شد. این نرم افزار مانند نرم افزار ORCAD که در آزمایشگاه مدارهای الکتریکی با آن آشنا شدید، امکان شبیه‌سازی مدارهای الکتریکی را ارائه می‌کند. افزون بر آن، شبیه‌سازی کارکرد خانواده‌هایی از میکروکنترلرهای در مدار را فراهم می‌کند. از این رو، می‌توان از این نرم افزار برای شبیه‌سازی آزمایش‌های درس ریزپردازندۀ بهره بردن.



محیط برنامه Proteus

چگونگی انجام شبیه سازی آزمایش‌ها در محیط Proteus

نخست مدار آزمایش را در نرم افزار Proteus طراحی کنید، سپس در محیط توسعه آردوینو در تابع `(void loop() {` پیکربندی ورودی/خروجی پایه‌های را انجام دهید و در تابع `)` منطق کنترلی برنامه را پیاده‌سازی کنید.

پس از آن، برنامه را برای ATmega2560 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 Address فایل هگز کامپایل شده را قرار دهید و شبیه‌سازی را اجرا کنید.

همچنان راه دیگری نیز برای دریافت فایل باینری در محیط Arduino IDE وجود دارد، پس از کامپایل شدن برنامه، گزینه Export Sketch در منوی Sketch را انتخاب کنید. این کار، فایل HEX برنامه را در پوشه‌ی Sketch Compiled Binary می‌ریزد و می‌توان با انتخاب 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 Arduino MEGA 2560
- دیود نورانی LED
- کلید
- مقاومت 220Ω
- مقاومت $10K\Omega$

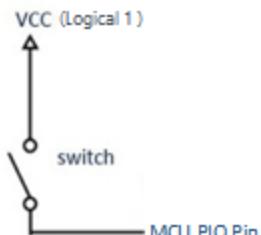
آنچه باید در پیش گزارش نوشته شود:

- 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 (1 یا 0) بر روی پایه میکرو (MCU PIO Pin) را پیوسته بررسی میکنیم و زمانی که برابر با 1 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.



بدیهی است تقاضت دو مدار فوق در سطح ولتاژ پایه میکروکنترلر در دو حالت فشرده شده یا آزاد میباشد. دو مدار فوق را بررسی کرده و از آنها در طراحی مدار خود بهره ببرید.

پرسش: در مدار های بالا چرا نیاز به مقاومت (Pull-Up/Pull-Down Resistor) داریم؟

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

وقفه:

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

واحد های گوناگونی می توانند تنظیم شوند تاریخ دادن 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 رو می 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 سرویس 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 سرویس وقفه متناظر با آن می شود؟

همه یا برخی از پایه های یک واحد GPIO برای ثبت رخداد های ورودی در نظر گرفته شده است. شمار این پایه ها بسته به مدل میکروکنترلر 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: انواع اتفاق های ورودی را که واحد GPIO در برد آردوینو ATmega2560 می تواند رخدان آن ها را بفهمد و اعلام کند بنویسید.

شرح آزمایش:

این آزمایش در بردارنده چندین LED (دلخواه اما بیشتر از 5 عدد) و سه دکمه می باشد. که مدار آن بر روی برد در شکل 1-1 و مدار شماتیک آن در شکل 1-2 نمایش داده شده است. در ابتدا LED ها خاموش می باشند. با هر بار فشردن دکمه یک، LED ها از سمت چپ یکی یکی 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 (فراخوانی `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

1. 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.

2. به پرسشهای زیر پاسخ دهید:

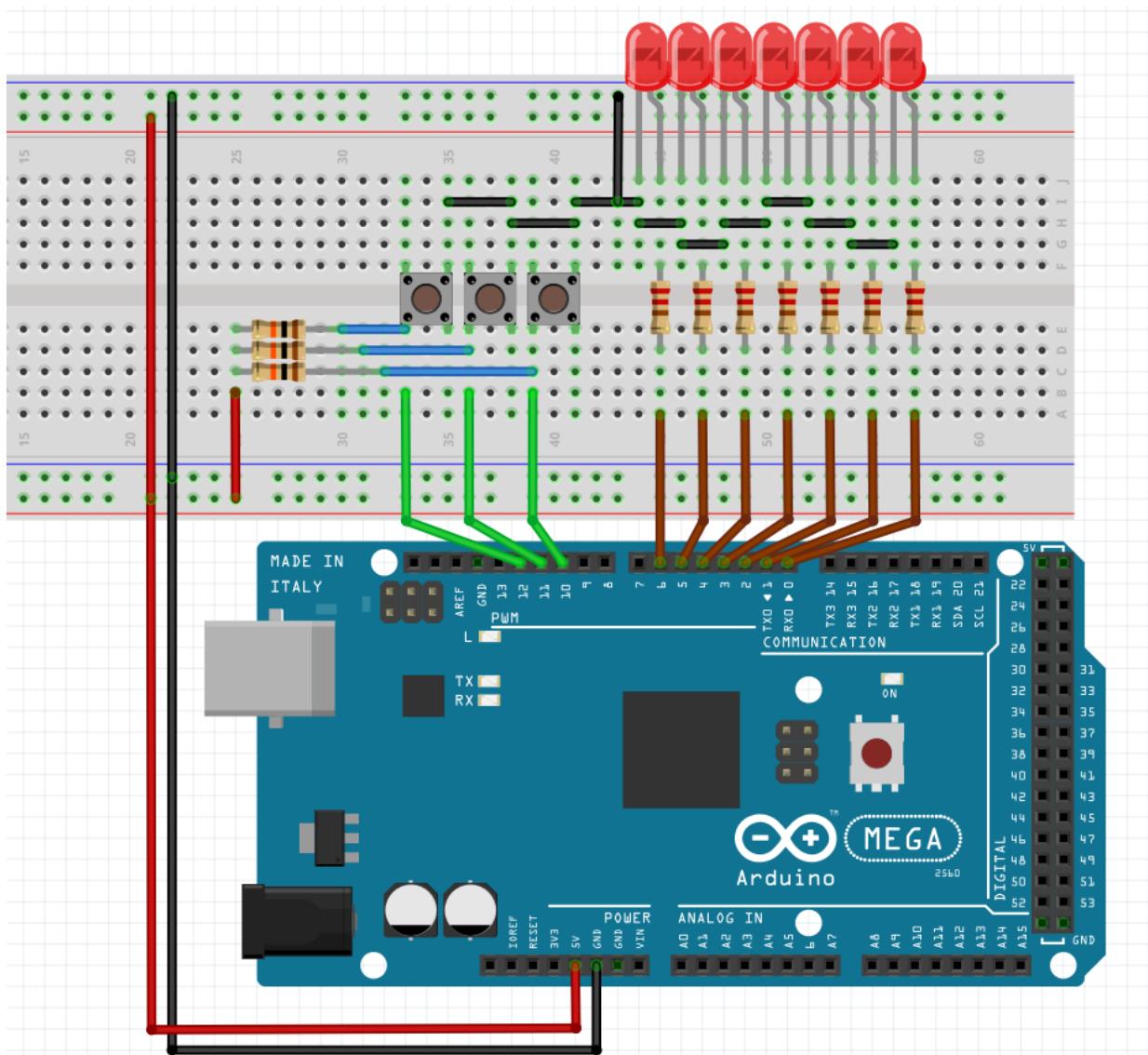
- اگر دکمه را در حالت فشرده برای زمان طولانی نگه داریم چه اتفاقی خواهد افتاد؟ آیا با منطق کارکرد خواسته شده سازگار است؟ چه راه حلی برای این مشکل (در صورت وجود) می توان پیشنهاد کرد؟

- فرض کنید می خواهیم برد موردنظر 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 5 ثانیه یک بار تغییر دهد. روشنی برای افزودن این کارکرد تازه به برنامه پیشنهاد دهید.

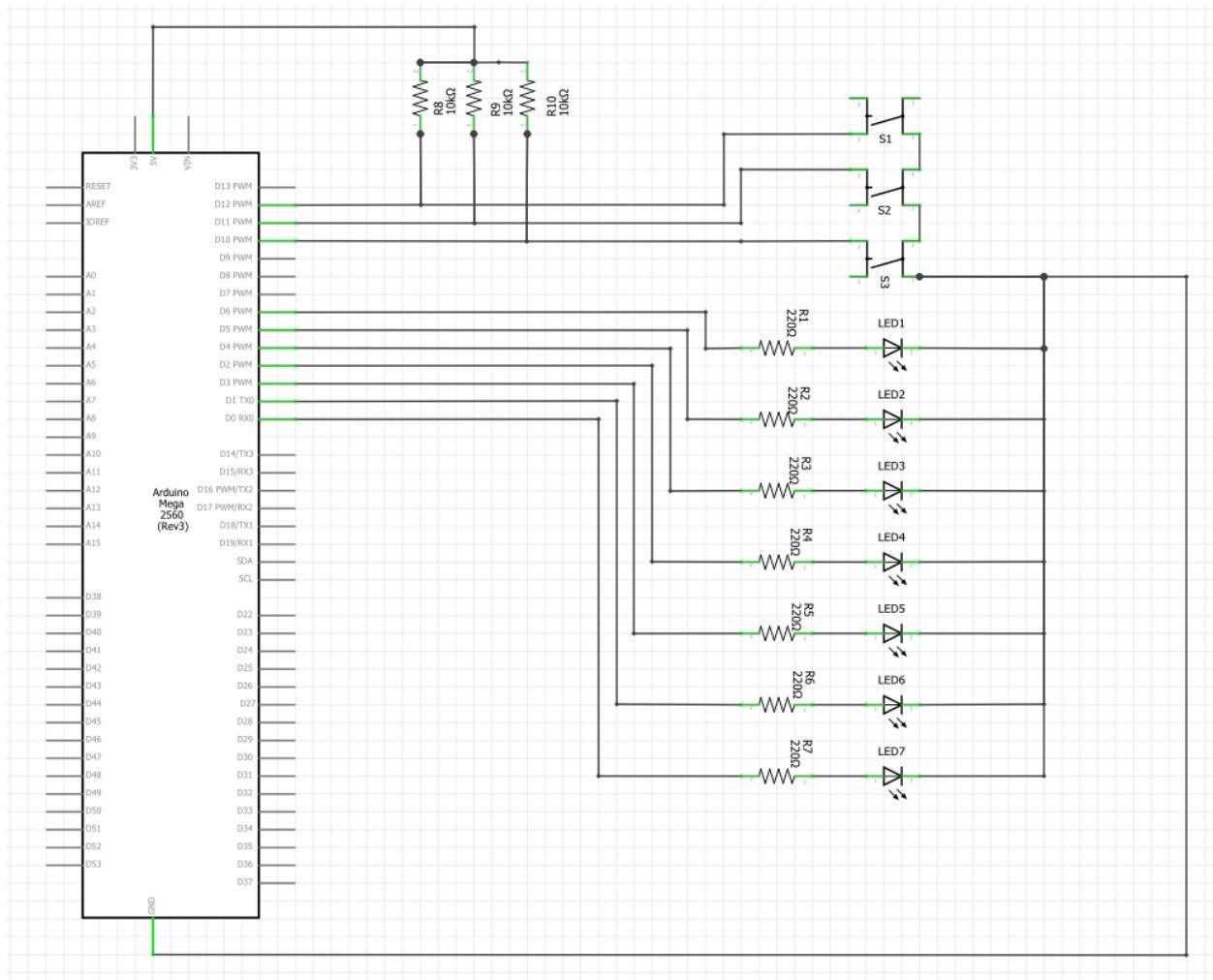
- فرض کنید می خواهیم کارکرد دیگری را به دستگاه اضافه کنیم به این صورت که در صورت یک شدن یک پایه عملیات مشخصی را به عنوان پاسخ انجام دهد. (حدودیت زمانی برای پاسخ دادن وجود دارد) هیچ یک از اتفاق های یک شدن پایه نباید از دست برود (بی پاسخ بماند). و یک شدن پایه نیز در هر زمانی ممکن است رخدهد. آیا برنامه شما -که به روش سرکشی واحد های جانبی 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 ببرود. 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؟

3. با پاسخ به پرسش های بالا می توان دریافت که روش سرکشی برای کنترل واحد های جانبی 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 1-1



شكل 2-1

آزمایش 2: کیبورد ورودی و ارتباطات سریال

هدف آزمایش:

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

راه اندازی یک ترمینال مجازی با Mega 2560 و آشنایی با ارتباطات Serial

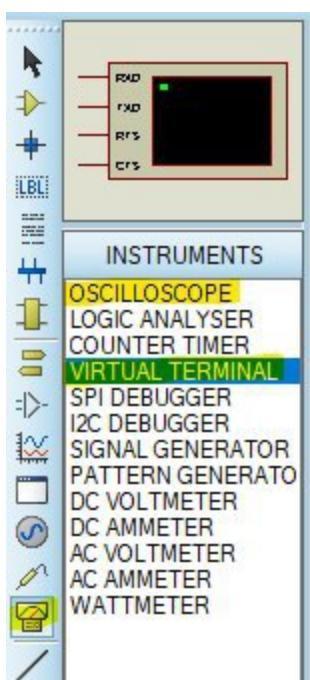
قطعات مورد نیاز:

- برد Arduino Mega2560
- صفحه کلید (Keypad) ماتریسی
- LED
- ترمینال مجازی برای سریال مانیتور
- اسیلوسکوپ

آنچه باید در پیشگزارش نوشته شود:

- کدهای مورد نیاز برای برنامه‌ریزی برد
- 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\(\)](#)



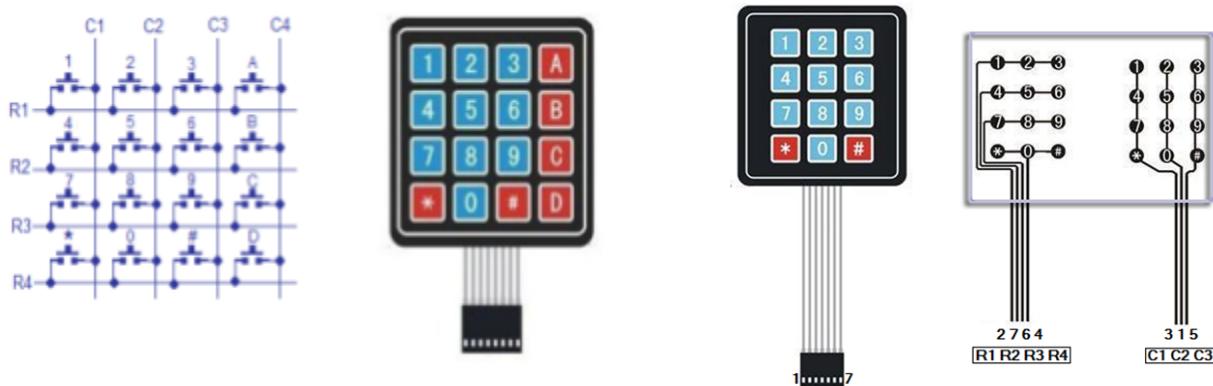
- نحوه و کاربردهای ارتباط سریال در آردوینو
- تعریف مختصر و نحوه کار با توابع ارتباط سریال مانند:

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

مقدمه:

صفحه کلید مجموعه‌ای از دکمه‌ها است که به صورت بلوك 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 برد را اشغال کنیم.

در یک صفحه کلید استاندارد، کلیدهای اصلی به صورت ۴×۳ ماتریس قرار گرفته‌اند که در حالت معمول به صورت رديفی و ستونی به یکدیگر متصل هستند. اگر یک صفحه کلید، ۱۲ کلید داشته باشد این صفحه کلید به صورت ۳ ستون ۴ رديفی به یکدیگر متصل می‌شوند. برد آردوینو می‌تواند از طریق پین‌های دیجیتال به این سطرها 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 سطرها و ستون‌ها وجود ندارد. شکل زیر ساختمان داخلی یک صفحه کلید ۴×۳ را نشان می‌دهد.



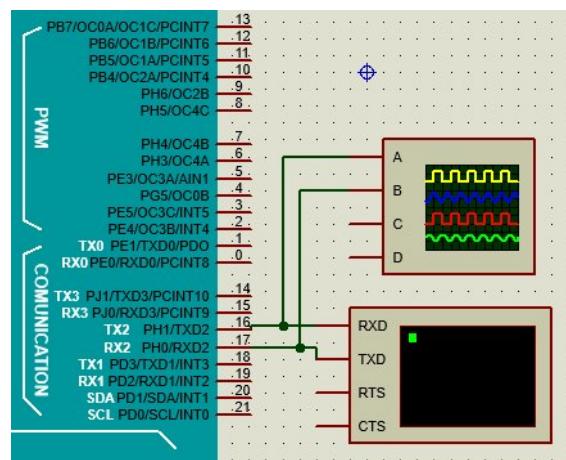
نمایی از ساختار درونی و بیرونی دو کلید

برای برنامه‌ریزی برد می‌توان از کتابخانه Keypad.h استفاده کرد. در صورتی که کتابخانه Keypad.h در نرمافزار IDE موجود نباشد، باید کتابخانه 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 پوشه \arduino\libraries\ ذخیره کنید.

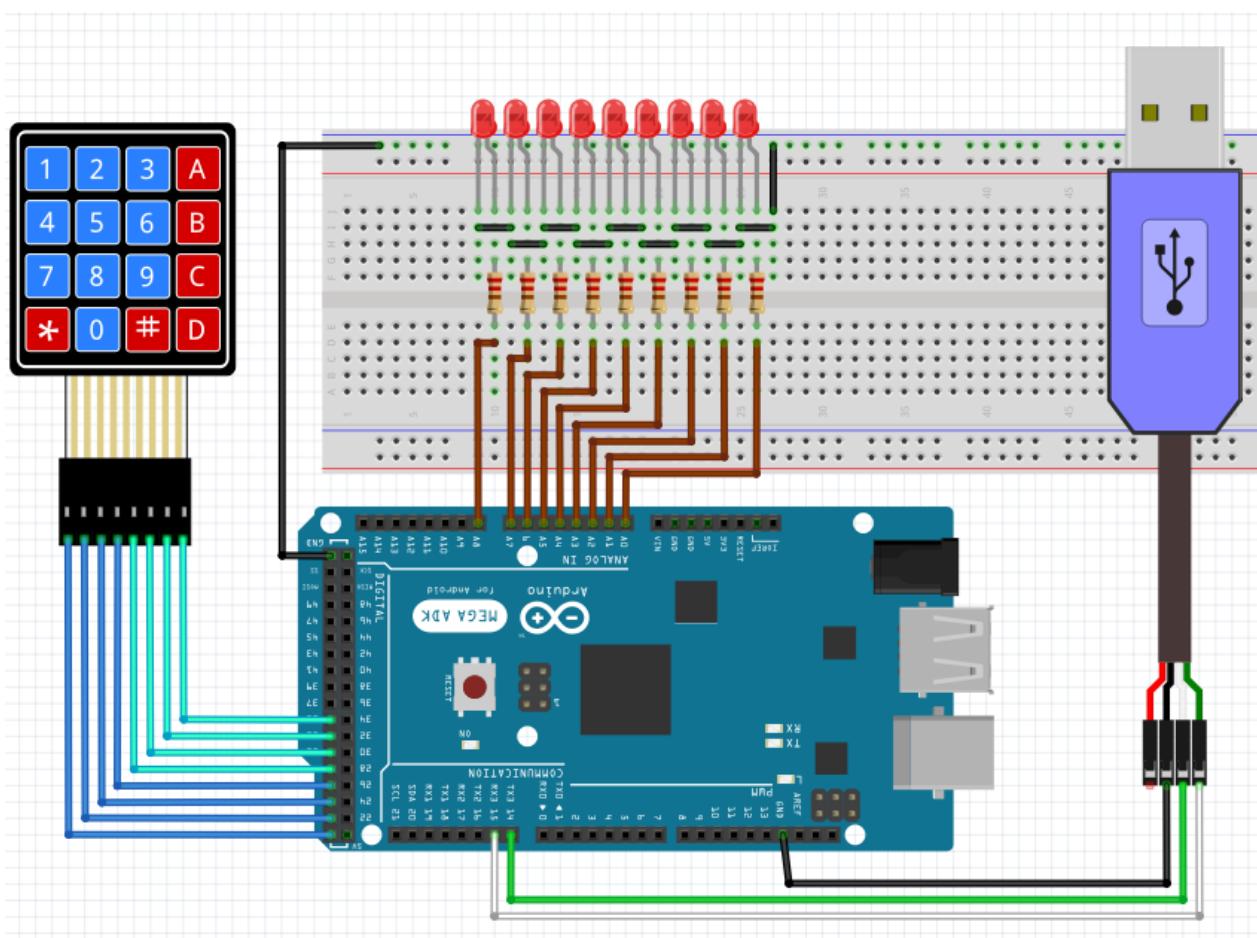
<https://playground.arduino.cc/uploads/Code/keypad/index.zip>

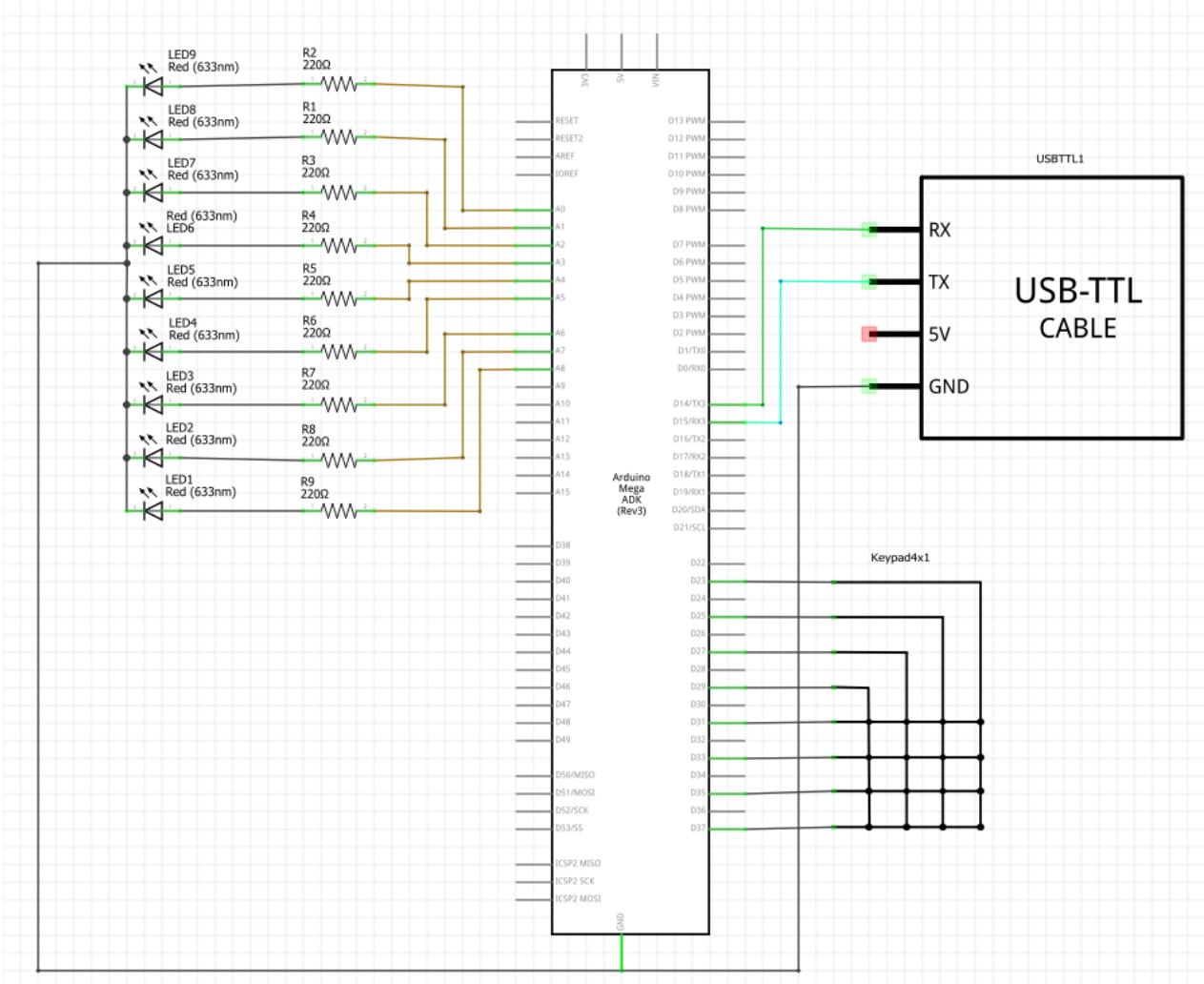


همچنین در این آزمایش قابلیت ارتباط سریال آردوینو را بررسی می‌کنیم. از نوار ابزار پروتوس (سمت چپ) و از تب INSTRUMENTS یک اسیلوسکوپ و یک ترمینال مجازی را انتخاب می‌کنیم. ترمینال را به یک TX و RX از پین‌های COMMUNICATION متصل می‌کنیم. این پین‌ها را برای مشاهده سیگنال‌های ارتباطی، به همان دو سر ترمینال مجازی متصل می‌کنیم. در آردوینو واحدهای UART و USART به کار برده می‌شوند و چنانچه از واحد USART یا USART به نام Serial استفاده می‌کنید (TXn، RXn) و همچنین اگر n = 0 باشد شماره Serial استفاده می‌کنید (TX0، RX0) در غیر این صورت Serials نام دارد.

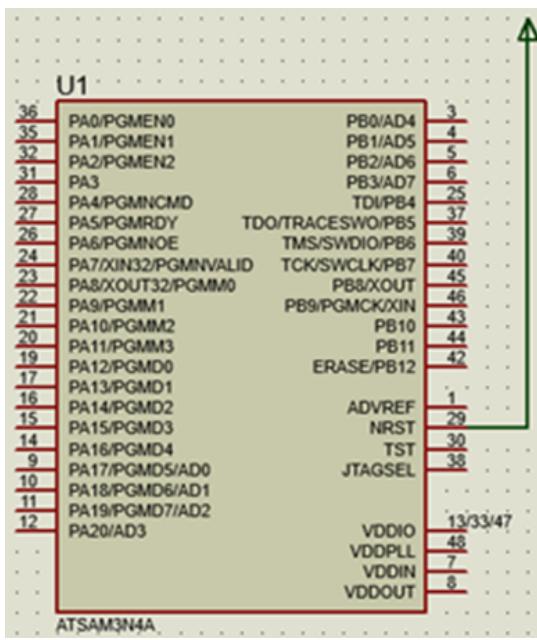
شرح آزمایش:

- نخست 9 عدد LED و یک صفحه کلید را از میان component‌های پرتوس اضافه و به برد Arduino Mega2560 وصل کنید. برنامه‌ای بنویسید که به تعداد عدد انتخاب شده در صفحه کلید از سمت چپ به راست LED هارا روشن کند.
 - سپس ترمینال مجازی را (که در شکل زیر به صورت یک تبدیل USB-TTL نشان داده شده است) به پین‌های ارتباطی برد وصل کنید. برنامه‌ای بنویسید که کاراکتر 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؟
 - برنامه‌ای بنویسید که ترمینال مجازی، یک عدد بین ۱ تا ۹ را به عنوان ورودی بگیرد و به تعداد آن LED هارا 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.





مقدمه



نکاتی درباره Proteus

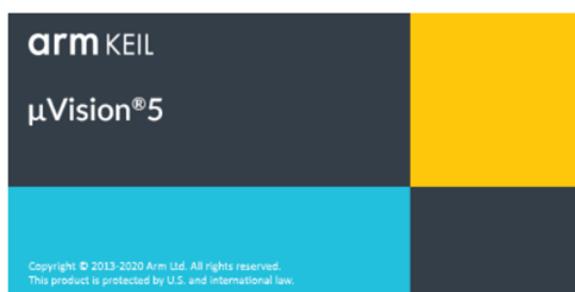
- از آنجا که میکروکنترلری که درس بررسی می شود ATSAM3X8E می باشد از این رو باید پروژه ها نیز بر روی همین میکرو انجام شود. اما از این میکروکنترلر پشتیبانی نمی کند. (Proteus)
- ولی خوبی‌خانه خویشاوند های این میکرو را پشتیبانی می کند: (Cortex-M3) یا خانواده ATSAM3N می باشد. و از دید معماری پردازنده (Cortex-M3) یا آدرس و کارکرد های پریفراں ها، میکرو های این خانواده نزدیکی زیادی با میکروکنترلر درس دارند.
- از این رو می توان از این میکروکنترلر ها 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.
- برای اینکه این میکروکنترلر ها را در شماتیک خود بیفزایید همانند گشته در پنجره Pick Device این بار به جای Arduino Mega 1280 نام ATSAM3N4A را جستجو کنید. و همانند گشته فایل هگر را که این بار از کامپایل پروژه در μ Vision به دست آورده اید، بر روی میکرو در پروتئوس بارگذاری کنید.
- از آن جا که پروتئوس تنها یک ابزار شبیه سازی می باشد، نیاز به افزودن VCC و GND به میکرو نیست و مدار زیر بسنده می کند.

ابزار توسعه keil

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

برای آشنایی بهتر با ابزار کیل میتوانید به [این ویدیو](#) مراجعه کنید.

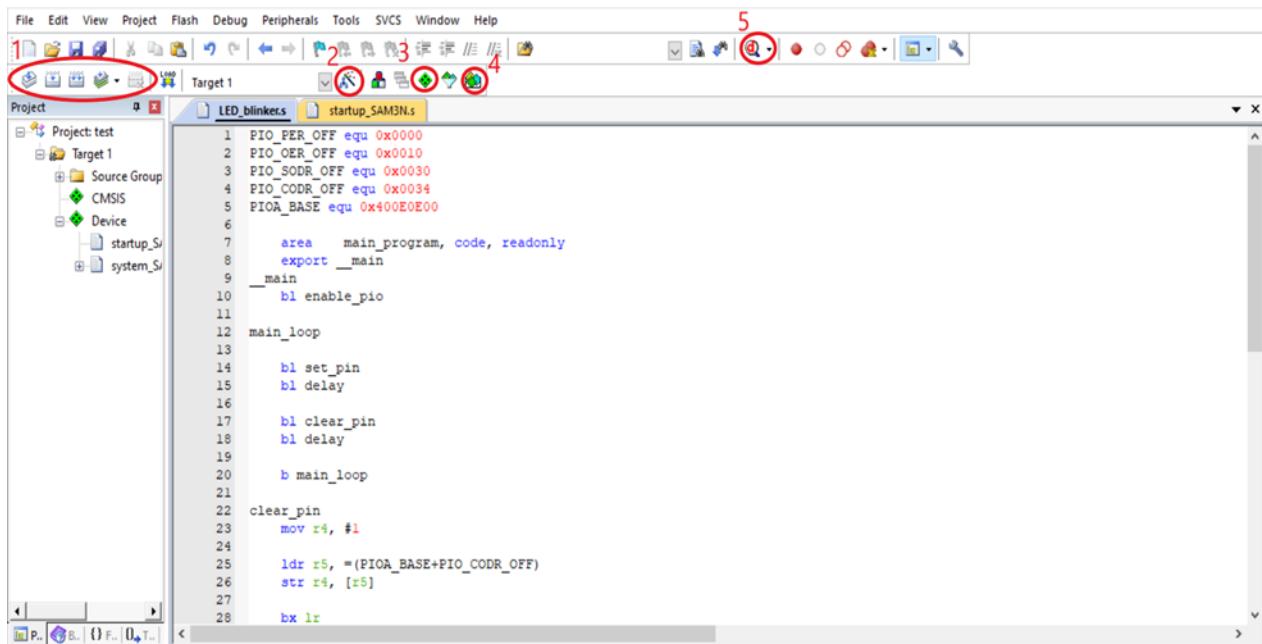
در این ابزار باید ابتدا پکیج های مربوط به خانواده میکروکنترلری را که در نظر داریم (ATSAM3N)، با ابزار Package Installer در μ Vision دانلود و نصب کنیم. سپس می توان با انتخاب میکروکنترلر دلخواه خود پروژه هایی را به زبان اسمنلی یا سی برای آن ساخت که در این صورت μ Vision کد ها و فایل های لازم برای کامپایل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



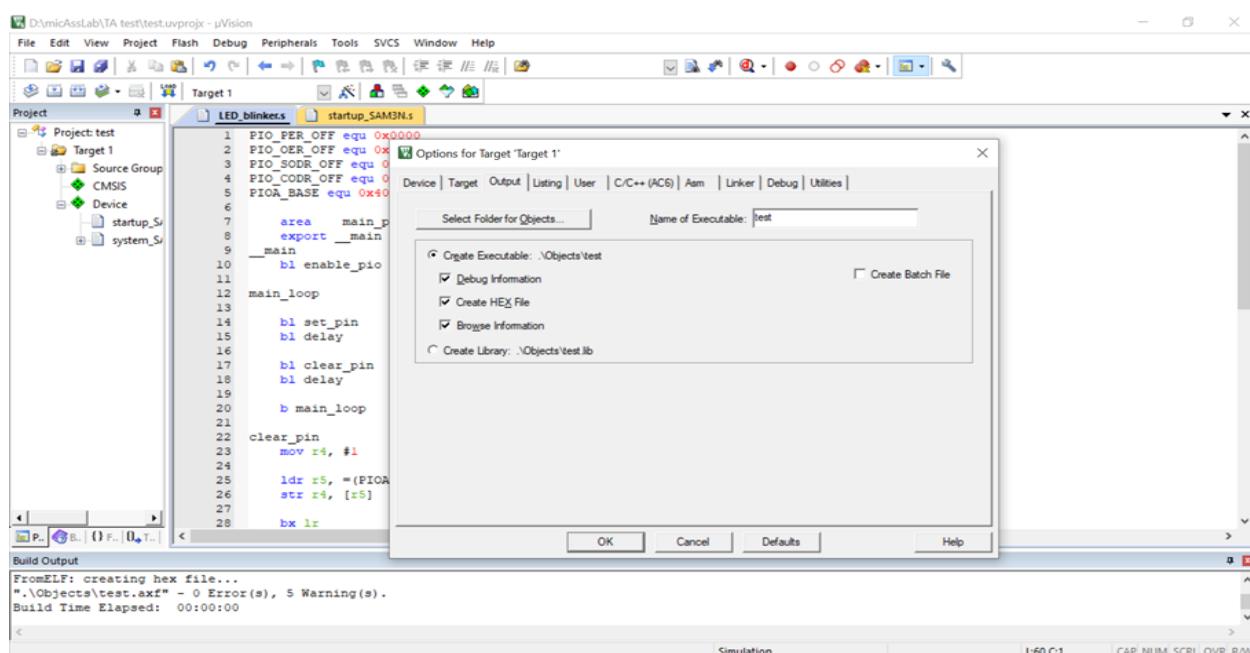
در پیوست فایل های کیل و پروتئوس برای پیاده سازی یک پروژه نمونه برای چراغ چشمک زن با اسمنلی بر روی میکروکنترلر بالا آورده شده است.

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

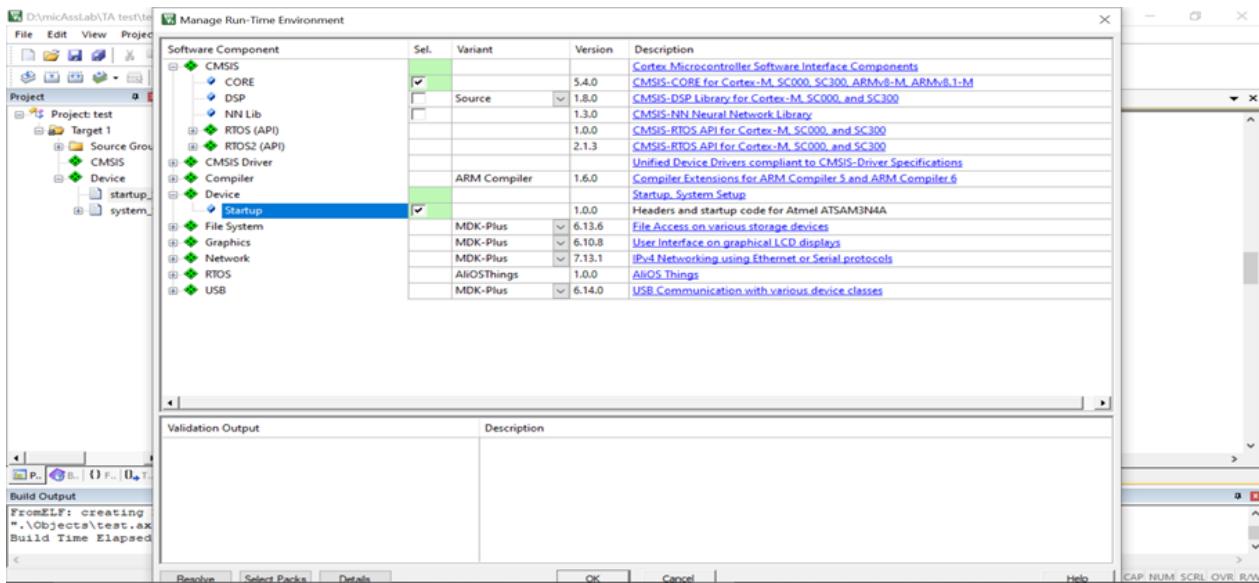
بخش یک: بخش هایی که به هنگام ساختن کد مورد استفاده قرار می گیرند هستند پس از تغییر کد از دستور rebuild استفاده می کنیم. با مطالعه در رابطه با بقیه دستور های گفته شده توضیحات مختصاتی ارائه دهید.



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 تیک create HEX file را فعال کنید.



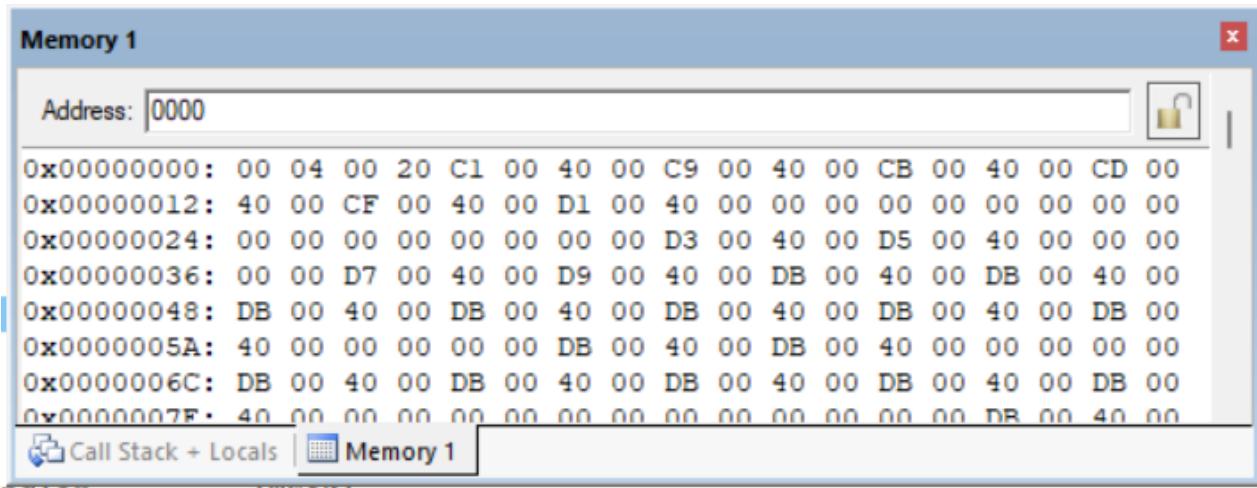
بخش سه : manage run-time environment است . باید دو بخش مشخص شده در شکل زیر را فعال کنید.



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 دیگر 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.

بخش شش: با استفاده از این بخش در حین debug می تواند محتوای هر خانه از حافظه خود را ببینید.



ما نیاز به کد اسمبلی از پیش آمده `startup` داریم این کد به منظور تعریف کردن میکروکنترلر استفاده می شود و باید با طراحی سخت افزاری ما مطابقت داشته باشد. می توانید در رابطه با بخش های 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 رجیستر ها و مدیریت سطح پایین با زبان اسملی و کار با ابزار keil

قطعات آزمایش:

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

آنچه باید در پیش گزارش نوشته شود:

- توضیحات مختصری درباره ی دستورات MOV, LDR و STR دهید.
- ایده ای برای پیاده سازیتابع تالخیر در زبان اسملی ارائه دهید.
- به پرسش ها در بخش مقدمه ای در رابطه با keil که بارنگ قرمز مشخص شده است پاسخ دهید.
- نحوه برقراری ارتباط میکروکنترلر و دیود ها کلید ها

مقدمه:

همانطور که در ویدیو آموزشی نیز به آن اشاره شد، برای اجرای برنامه‌ی نوشته شده با استفاده از زبان اسملی 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 در بردارنده سه عمل مهم زیر می باشد :

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

:startup.s کد بررسی

```
27 Stack_Size      EQU      0x00000200
28
29             AREA      STACK, NOINIT, READWRITE, ALIGN=3
30 Stack_Mem       SPACE     Stack_Size
31 __initial_sp
```

تصویر-1 دستورهای پیکربندی پشتئ

#define EQU به اندازه 0X00000200 تعریف می شود. (معادل دستور EQU مانند در زبان C می باشد) 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 نوشتن نیز وجود دارد). در آخر نیز با استفاده از SPACE برای STACK حافظه را رزرو می شود.

```
38 Heap_Size       EQU      0x00000200
39
40             AREA      HEAP, NOINIT, READWRITE, ALIGN=3
41 __heap_base
42 Heap_Mem        SPACE     Heap_Size
43 __heap_limit
```

تصویر-2 دستورهای پیکربندی هیپ

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

```

52      AREA    RESET, DATA, READONLY
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

```

تصویر-3 دستورهای پیکربندی بردار وقفه

در این قسمت interrupt vector table را مشاهده می کنیم. در اولین خط آن initial_sp را داریم که حاوی بالاترین آدرس stack می باشد. هنگامی که میکرو شروع به کار کردن می کند رجیستر sp بوسیله ی initial_sp مقدار دهی می شود.

دومین خط این قطعه کد آدرس وقفه ی reset_handler می باشد که به تابع reset_handler اشاره می کند که در ادامه به آن می پردازیم.

برای مثال برای تغییر روند وقفه در هر یک از این قسمت ها باید کد PROC مرتبط با قسمت مورد نیاز را پیدا کرده و آن را تغییر داده یا از ابتدا تغییر دهیم.

```

111  Reset_Handler  PROC
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

```

تصویر-4 پیاده سازی روال Reset Handler

وظیفه ای اصلی تابع `reset_handler`، تنظیم کلک سیستم و در نهایت پریدن به تابع `main` برنامه ای ما می باشد. که برای قسمت هایی میخواهیم برنامه را دیباگ کنیم و یا تمرین های درس که قسمت عملی ندارد دو خط 115 و 116 باید کامنت شود.

شرح آزمایش:

این آزمایش شامل 3 عدد LED و 2 دکمه می باشد. در اینتا LED ها خاموش می باشد. هنگامی که دکمه ای اول فشرده می شود. هر 3 شروع به چشمک زدن می کنند با این تفاوت که سرعت چشمک زدن LED ها از چپ به راست افزایش پیدا می کند. اولین LED کند ترین و آخرین تند ترین چرا غ چشمک زن ما است و در اصطلاح کمترین مقدار تأخیر را بین هر بار خاموشی دارد. با فشردن دکمه ای دوم نیز همه ای LED ها خاموش می شوند.

آزمایش ۳: مانیتور خروجی

هدف آزمایش:

اتصال صفحه نمایش کاراکتری به میکروکنترلر برای نمایش اطلاعات

قطعات مورد نیاز:

- برد Arduino Mega2560
- صفحه نمایش کاراکتری 16×2
- مقاومت 220Ω یا پتانسیومتر $10k\Omega$
- ترمینال مجازی برای سریال مانیتور
- کیبورد 4×4

آنچه باید در پیشگزارش نوشته شود:

- کدهای مورد نیاز برای برنامه ریزی برد
- مشخصات فنی مازول نمایشگر ال سی دی کاراکتری 2×16 و دلیل استفاده از پتانسیومتر در مدار

- تعريف 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.



LCD کاراکتری 2×16 یکی از پایه‌ای ترین نمایشگرهای الکترونیکی است. مژول نمایشگر دارای یک صفحه نمایش LCD با قابلیت نمایش 2 سطر و 16 ستون کاراکتر است. این مژول در شکل (1-3) نشان داده شده است. پایه‌های LCD کاراکتری از چپ به راست به صورت زیر است:

- | | |
|--|----|
| GND = VSS | .1 |
| VCC = تغذیه ۵ ولت | .2 |
| VO (Display Contrast Pin) = تنظیم شدت نور صفحه | .3 |
| RS (Register Select) = انتخاب رجیستر | .4 |
| RW (Read/Write) = پایه Read و Write | .5 |
| Enable = پایه E | .6 |
| D0 – D7 = پایه‌های دیتا (داده 8 بیتی) | .7 |
| Anode = پایه A | .8 |
| Cathode = پایه K | .9 |

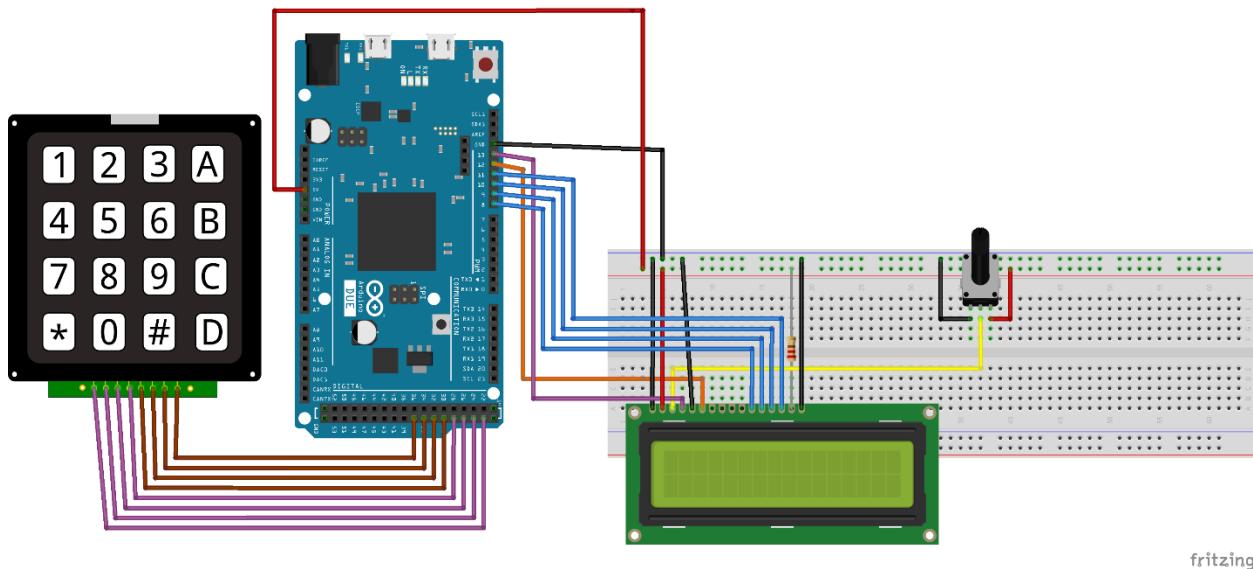
برای برنامه‌ریزی LCD موجود می‌توانید از کتابخانه LiquidCrystal استفاده کنید که به صورت پیش‌فرض در نرم افزار IDE آردوینو وجود دارد. لازم به ذکر است که در محیط شبیه سازی پروتوس، مقاومت/پتانسیومتر تاثیری ندارد و صفحه نمایش همواره اعداد را نشان میدهد، در حالی که در عمل میزان ولتاژ آن VO، میزان نمایان بودن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

شرح آزمایش:

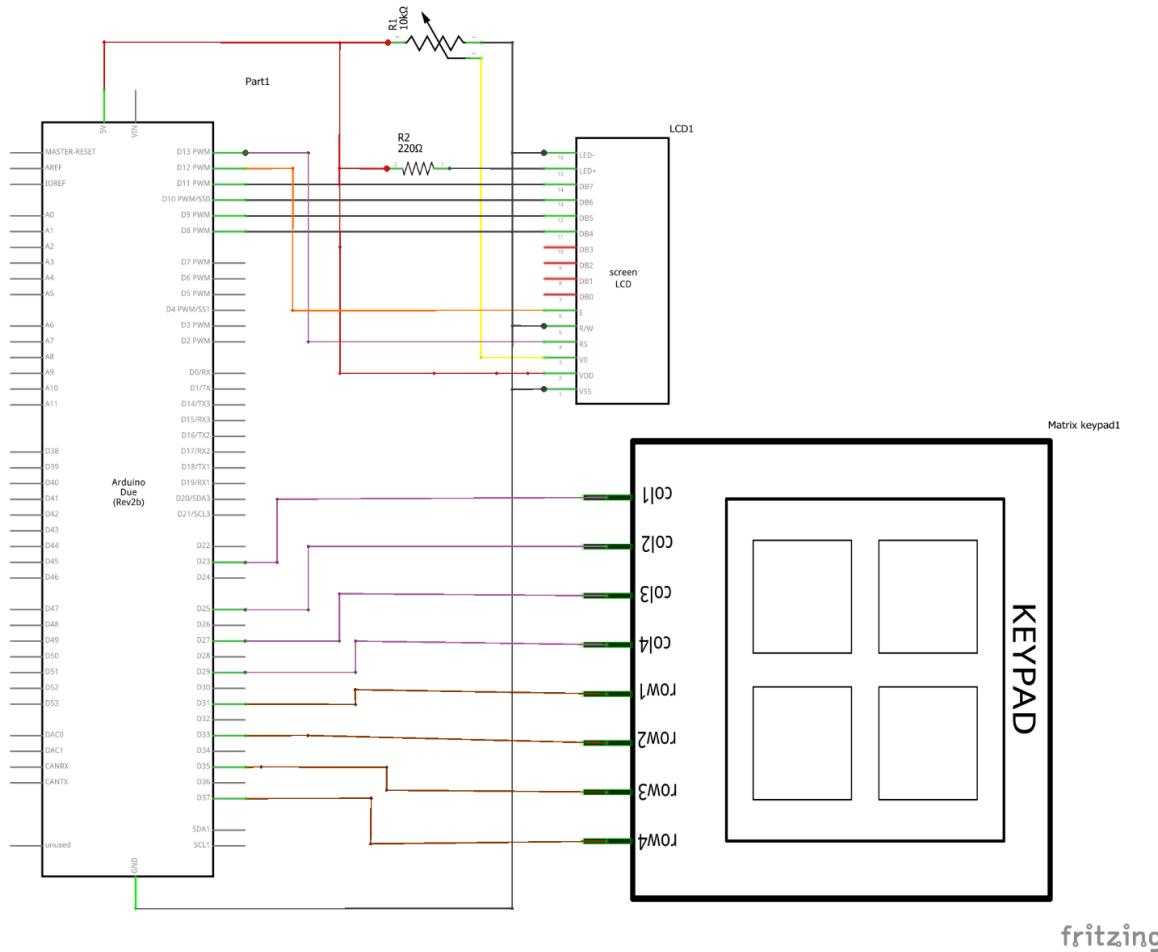
- همانطور که در عکس زیر می‌بینید، ابتدا یک صفحه نمایش را از بین component‌های پروتوس اضافه کرده و به برد Arduino Mega 2560 وصل کنید. با استفاده از توابع کتابخانه‌ای برنامه‌ای بنویسید که اسم خودتان در ابتدای سطر اول LCD کاراکتری نوشته شده و هر ثانیه یک خانه به جلو حرکت کند و وقتی به انتهای خط اول رسید به خط دوم رفته و باز به سمت جلو حرکت نماید. سپس به طور پیوسته این کار را تکرار نماید.
- یک صفحه کلید از پیش‌فرض های پروتوس 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) بررسی کنید. در صورت درستی رمز عبور پیغام "correct password" در خط دوم نمایش داده شود و در غیر اینصورت پیغام "wrong password" نمایش داده شود.
- .3. یک کد ماشین حساب بنویسید که بر اساس نوشه های کلید (که در LCD مانند 59+23 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.
- .4. با استفاده از توابع کتابخانه ای برنامه ای بنویسید که یک کاراکتر را در ایندای سطر اول LCD بنویسد و هر ثانیه در حالی که یک خانه به جلو حرکت می کند، به خط بعدی برود(اگر در خط اول بود به خط دوم برود و بر عکس). سپس بطور پیوسته این کار را تکرار نماید.



fritzing



آزمایش 4: راه اندازی سروو موتور و ورودی آنالوگ

هدف آزمایش

راه اندازی سروو موتور با Arduino Mega 2560 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 2560
- سروو موتور (MOTOR-PWMSERVO) در پروتئوس
- پتانسیومتر (POT در پروتئوس) با مقاومت 10K اهم
- کیبورد 4x4
- اسیلوسکوپ

آنچه باید در پیشگزارش نوشته شود:

- مفهوم PWM و استفاده های آن
- 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()`

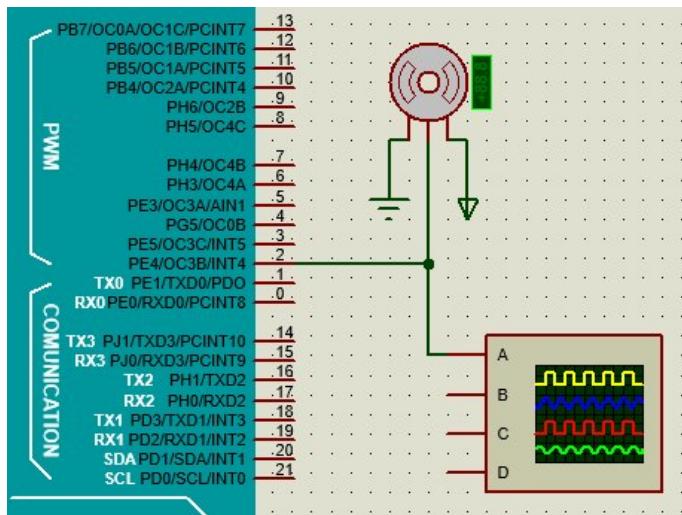
مقدمه:

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.

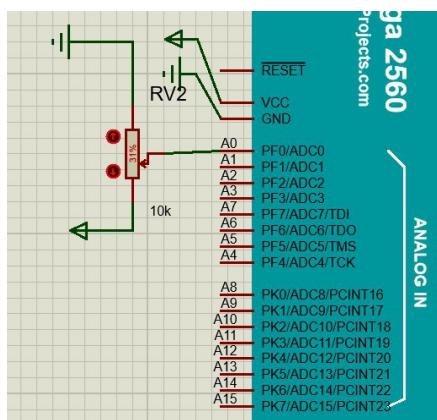
برای مدیریت سروو موتور، نیاز به موج مربعی با PWM است. به عبارت دیگر، سروو موتور یک موتور کوچک دارای یک محور یا شفت خروجی است. این محور خروجی قادر است در یک موقعیت و زاویه خاص با سیگنال دریافتی قرار گیرد.

سروو موتور دارای سه پایه به ترتیب Signal، VCC، GND است. سیم تغذیه به $+5$ ولت، سیم زمین به زمین مدار، و در آخر سیم سیگنال به یک پین دیجیتال آردوینو که قابلیت PWM داشته باشد متصل می شود.

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



سپس برای اتصال پتانسیومتر، پایه‌های کناری را به تغذیه +5 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.



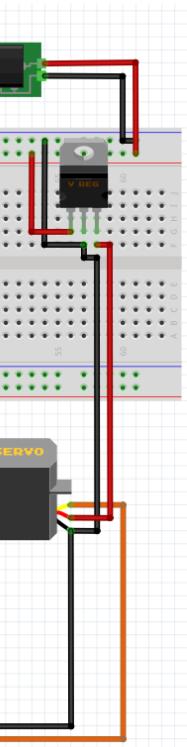
پیشنهاد: تابع map آردوینو در کار با سروو موتور و خواندن داده آنالوگ پتانسیومتر میتواند به شما کمک شایانی نماید.



سروو موتور SG90

شرح آزمایش:

1. برنامه‌ای بنویسید که به صورت خودکار سروو 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.
2. برنامه‌ای بنویسید که کاربر با کیبورد یک عدد بین ۰ تا ۳۶۰ انتخاب کرده و سروو موتور آن را بین -۱۸۰ درجه و +۱۸۰ درجه نشان دهد.
3. برنامه‌ای بنویسید که با استفاده از سریال مانیتور، مقدار زاویه مورد نظر را وارد کنیم و سروو موتور به اندازه‌ی فرینه‌ی آن عدد تغییر زاویه دهد.
4. برنامه باید به گونه‌ای نوشته شود که با تغییر مقدار پتانسیومتر که به یک پایه آنالوگ برد متصل است، زاویه سروو موتور تغییر کند.
5. اسیلوسکوپ را به خط سروو موتور متصل کنید. چه چیزی متوجه می‌شود؟ آیا می‌توانید دوره پایه (Fundamental) و همچنین دوره کاری ([Duty Cycle](#)) موج را به ازای زاویه‌های گردش مختلف سرو موتور به دست آورید؟



1

SB-TTL
CABLE

آزمایش 5: راه اندازی رله با Arduino MEGA2560

هدف:

آشنایی با سوییچ 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 قطعاتی مانند لامپ که نیازمند جریان یا ولتاژ بالایی هستند و مستقیم با پایه های برد قابل فعال سازی نیستند.

قطعات مورد نیاز:

Arduino Mega	●
رله	●
موتور	●
LED	●
مقاومت $10k\Omega$ و $1k\Omega$	●
ترانزیستور NPN	●
دیود	●
باتری	●
دکمه	●
ولت سنج (DC Voltmeter)	●
آمپر سنج (DC Ammeter)	●

آنچه باید در پیشگیری از 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؟

مقدمه

کلیدها می توانند با فرمان های مختلفی تحریک شوند. فشار مکانیکی، نور، لرزش و صد البته جریان الکتریکی! وقتی فرمان یک کلید جریان الکتریکی است، نام سوییچ 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 می توانند جریانی قوی تر از جریان ورودی را هدایت کنند، به معنی وسیع تر می توان آن ها 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 آن ها، رله های SPDT هستند. نمونه ای از این نوع رله در شکل (1-6) 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 5 پایه هستند که دو پایه coil برای فرمان (قسمت فرمان) و سه پایه برای خروجی (مدار قدرت) دارند.



دو نمونه از رله های 5 ولت SPDT

نمایی از ساختار درونی رله SPDT

1. پایه Common (COM): پایه مشترک میان دو پایه NO و NC می باشد که بسته به شرایط به یکی از این دو پایه متصل است. می توانیم این پایه را به یک سر موتور متصل کنیم.
 - 2،3. 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.
 4. پایه Normally closed (NC): پایه ای که در حالت عدم تحریک کویل به COM متصل است. می توانیم این پایه را به منفی تغذیه متصل کنیم.
 5. 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 می کنیم.

پرسش: درباره چگونگی کارکرد این مدار توضیح دهید.

شرح آزمایش:

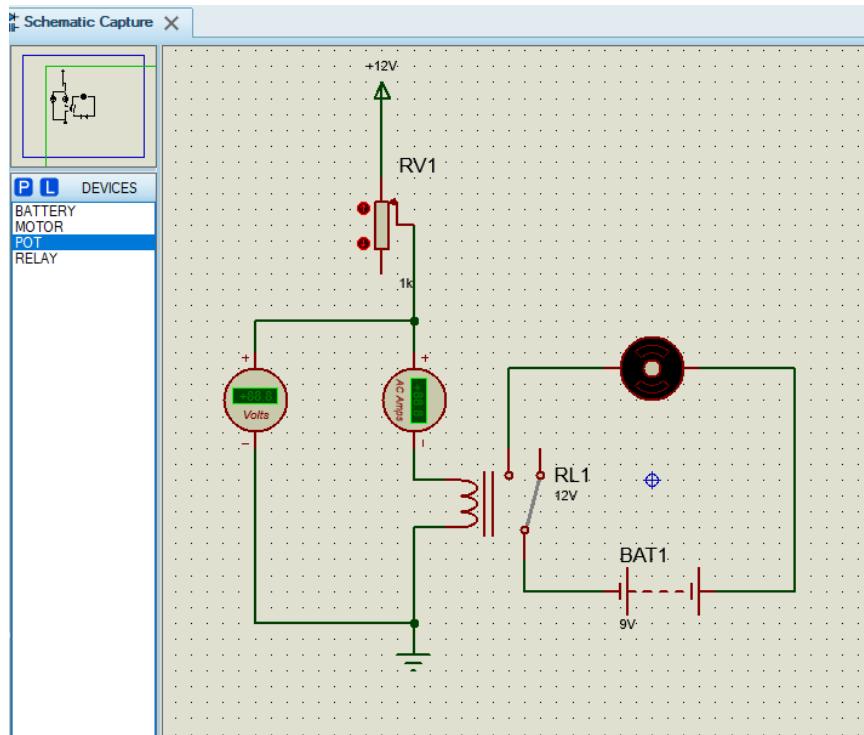
- در این آزمایش همانند مدار های شکل 1 و 2 که در زیر آورده شده است، می خواهیم با رله یک موتور را راه اندازی کنیم، هنگامی که دکمه را فشار میدهیم موتور روشن و LED خاموش می شود و هنگامی که دکمه را رها می کنیم موتور خاموش و LED روشن می شود.
- گام یک:

ابتدا مدار زیر را با پتانسیومتر (POT)، رله (Relay)، موتور (Motor)، باتری، آمپر متر 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 آمپر متر کمترین مقدار را نشان خواهند داد. همچنین رله غیر فعال است.

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

همان گونه که دیده می شود ولتاژ آستانه مورد نیاز برای فعال شدن رله به مرتب بیشتر از ولتاژ یک منطقی ATMega2560 است.
(5v)

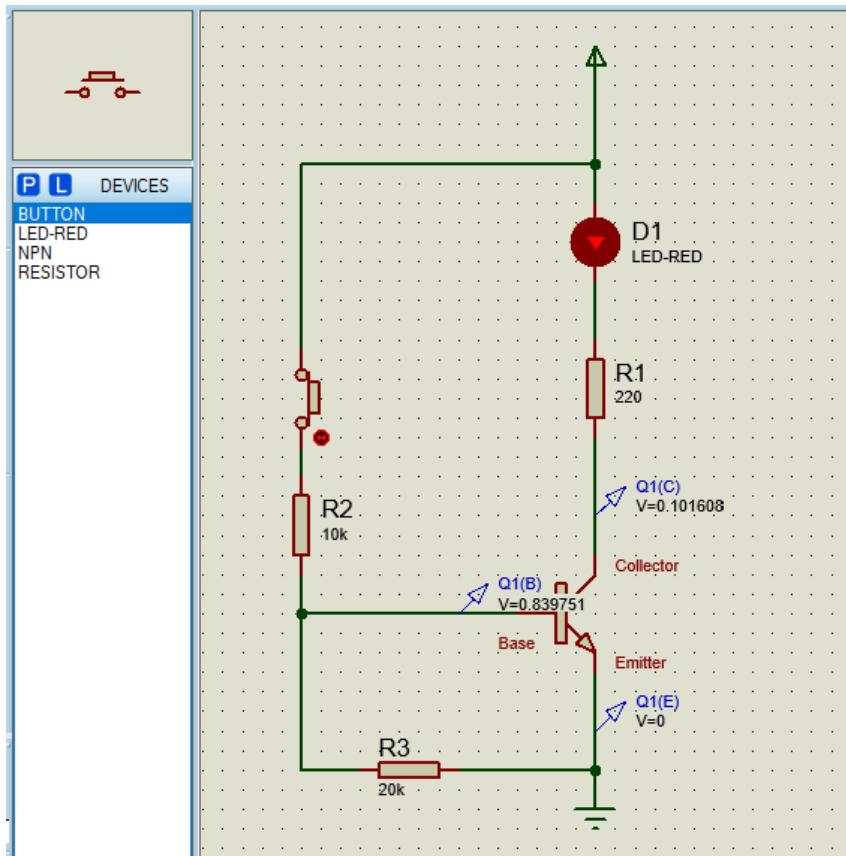
همچنین جریانی که برای فعال کردن رله نیاز هست نیز معمولا از بیشینه جریانی که هر پین ورودی/خروجی ATmega2560 فراهم می‌کند بیشتر است. (40mA)



نتیجه: از این رو پین‌های ورودی/خروجی این میکروکنترلر به تنهایی توانایی فعال کردن رله یا آرمیچر را ندارد و به کاربردن آن‌ها برای این منظور می‌تواند سبب **آسیب رسیدن به میکروکنترلر و سوختن آن** شود، در نتیجه از ترانزیستور برای فعال کردن رله بهره می‌بریم.

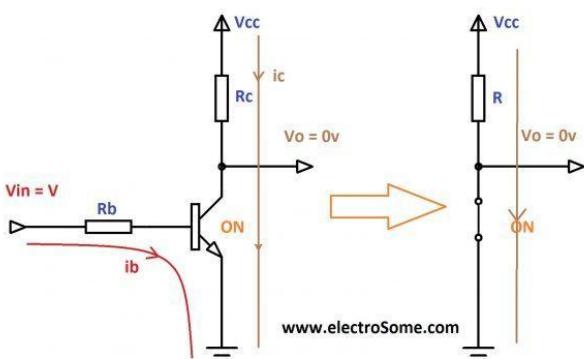
گام دو:

برای آشنایی با شیوه کار کرد تر انزیستور های NPN مداری مانند مدار زیر را در Proteus رسم کرده و آن را اجرا کنید.

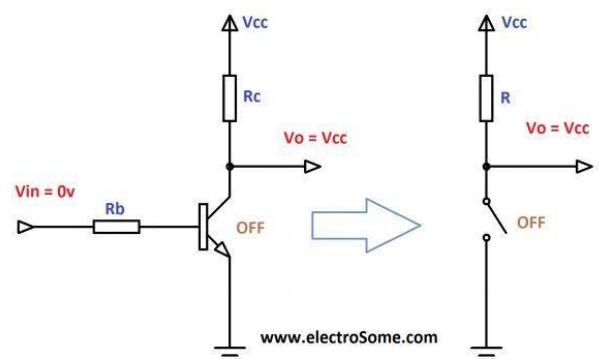


همان گونه که دیده می شود به طور خلاصه می توان گفت به ازای افزایش جریان از پایه Base به I_B ، بیشینه جریانی که از Emitter به Collector می تواند بگذرد به نسبت بیشتری افزایش می یابد. به گونه ای که در حالت اشباع ترانزیستور می توان از مقاومت ترانزیستور از Emitter به Collector در مدار چشم پوشی کرد.

حالت کلید بسته:

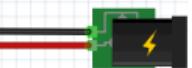


حالت کلید باز:



گام سوم:

از این ویژگی ترانزیستور برای راه اندازی رله استفاده کنید و مدار نهایی خواسته شده برای این آزمایش را که مدار شماتیک آن در شکل 2 آورده شده است، رسم کنید.



نمودار 1 (مدار آزمایش بر روی برد برد)



نمودار 2 (شماییک لازم برای راه اندازی موتور به کمک رله و بورد Arduino Mega

آزمایش 6: نیم-پروژه

هدف آزمایش:

مروری بر مطالبی که در جلسه‌های گفته شد و پیاده‌سازی یک کاربرد ساده از این مطالب با بهره‌مندی 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

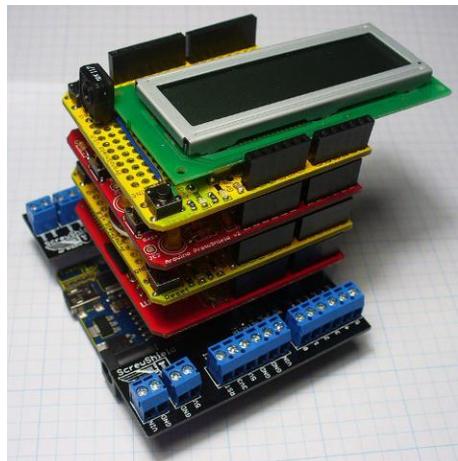
آردوینوهای همکار

هدف پروژه

پیاده‌سازی دو برد آردوینو که با فرستادن و دریافت اطلاعات از راه ارتباطات سریال، می‌تواند پکسری مسائل بازگشتی (مانند بررسی زوج-فرد بودن بازگشتی) را حل کند.

قطعات مورد نیاز:

- دو عدد بورد Arduino Mega2560
- کیبورد
- LCD کاراکتری



کامپیوتر ماشین کنترلی

هدف پروژه

این ماشین باید از ارتباطات سریال دستور بگیرد (که فرضا به کنترلر رادیویی متصل شده)، چراغهای ماشین (LED) را روشن کند، به چرخهای جلو زاویه بدهد (سرво موتور)، چرخهای عقب را به حرکت بیندازد (رله و آرمیچر) 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 موردنظر توسط کیبد دریافت و روی LCD نمایش داده می‌شود. مود ها عبارتند از stepper mode و servo mode. در حالت سروو تغییر جهتی که در ابتدا گفته شد انجام می‌شود و در حالت استپر حرکت ماشین همانند ویلچر است stepper یک موتور همانند سروو است و یک کتابخانه دارد. برای اطلاعات بیشتر درباره آن به [\[لينك\] مراجعه کنید](#). برای تغییر سرعت ماشین نیز یک پتانسیومتر روی آن قرار دهید.



قطعات مورد نیاز:

- بورد Arduino Mega2560
- دیود نورانی (LED) به مقدار لازم

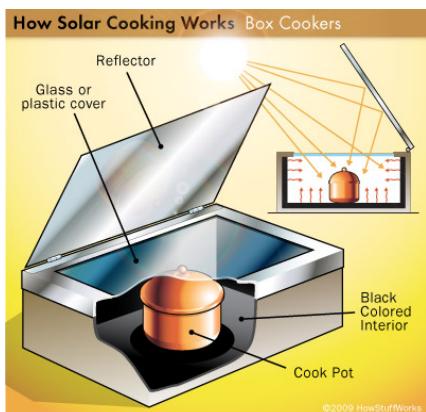
- سروو موتور
- آرمیچر
- رله
- بازر
- نمایشگر (LCD)
- کیبورد

●

اجاق خورشیدی

هدف پروژه

اجاق خورشیدی دستگاه نور خورشید را از شیشه می‌گیراند و توسط دیوارهای سیاه درونی، گرمای را به دام می‌اندازد. میکروکنترلر شما باید شیشه را باز و بسته کند (سروو موتور) یک دما و یک زمان بگیرد (کیبورد) که حرارت درونی به دمای گرفته شده رسید (حرارت سنج) یک تایмер را شروع کند و پس از گشتن 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 Mega2560
- سروو موتور
- حرارت سنج
- کیبورد
- بازر
- LCD



یعقوب برقی رایگان!

هدف پروژه

سیر کردن دانشجوها. یک گذروازه هفت رقی (کیبورد) در را باز می‌کند (یک سروو موتور) همه پیچش ها را (تعداد زیادی سروو موتور) به عقب می‌چرخاند تا غذا کار گذاشته شود. همه

چراغها (LED) در این مرحله خاموش است. دکمه دوباره در را می‌بندد و چراغها روشن می‌شوند. سپس دانشجوها با زدن شمارهای دورقمی غذا، آن را تحویل می‌گیرند. همه چیز روی LCD نشان داده شود و دکمه پاک و تایید داشته باشد.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- دیود نورانی (LED) به مقدار لازم
- تعداد زیادی سروو موتور
- کیبورد
- LCD کاراکتری

پنکه

هدف پروژه



خنک کردن دانشجوها. یک دکمه (button) به کمک رله پنکه را به راه می‌اندازد (آرمیچر و رله)، هنگام فشرده شدن دکمه، بازر بوق می‌زند (buzzer) و یک دکمه دیگر بین سه سرعت مختلف آرمیچر سوییچ می‌کند که هر سه توسط یک نور (LED) به کاربر نشان داده می‌شوند. یک دکمه دیگر هم به کمک سروو موتور چپ و راست شدن پنکه را فعال و غیرفعال می‌نماید. همچنین پنکه باید توانایی برنامه پذیری داشته باشد، به این شیوه که با یک صفحه کلید و برنامه دوره زمانی رفت و برگشت پنکه و همچنین سرعت موتور مشخص می‌شود. که این برنامه بر روی LCD کاراکتری نمایش داده خواهد شد.

برای این‌که 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 Mega2560
- سه عدد LED
- چهار دکمه (push button)
- آرمیچر و رله
- بازر
- صفحه کلید
- LCD کاراکتری



گاو صندوق

هدف پروژه

کاربر گذرواژه را وارد می‌کند (صفحه کلید) و سپس در گاو صندوق باز (سروو موتور) و چراغ نمایانگر باز بودن در روشن خواهد شد. و بازر به هنگام باز و بسته شدن در هشدار می‌دهد (بوق). کاربر می‌تواند با فشردن دکمه در را

بینند، 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 گاوصندوق یک دزدگیر داشته باشد به این صورت که در صورتی که رمز 4 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 گاوصندوق به مدت یک دقیقه غیر فعال می‌شود. اگر پس از 4 بار اشتباه و همچنین گذشت یک دقیقه رمز درست وارد شد در گاوصندوق باز می‌شود در غیر این صورت باید به مدت دو دقیقه غیر فعال شود و به همین ترتیب.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- LED به مقدار لازم
- کیبورد
- LCD کاراکتری
- سروو موتور
- Buzzer

تردمیل



هدف پروژه

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

قطعات مورد نیاز:

- بورد Arduino Mega2560
- سروو موتور
- رله
- موتور
- دکمه و صفحه نمایش LCD

آزمایش 7: ماشین لباسشویی ساده

هدف آزمایش:

آشنایی با پروتکل ارتباط سریال TWI

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

قطعات مورد نیاز:

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

آنچه باید در پیش‌گزارش 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 AT24C02 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 کثیر‌بودی به EEPROM نیاز داریم.

| پرسش: در چه کاربردهای EEPROM به کار برد 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؟ 2

در یک دستگاهی می‌توان EEPROM را به دو نوع درونی (Internal) و بیرونی (External) دستگاهی کرد. حافظه‌های درونی در کنار پردازنده بر روی یک IC قرار می‌گیرند. در نتیجه، نیازی به کارکرد پایه‌های آدرس (A0-A2) نیست و می‌توان سریال استفاده از پروتکل ارتباط سریال برای نوشتن و خواندن را از میان برداشت. بنابراین حافظه‌های درونی تندتر هستند، ولی اندازه‌شان محدود است.

میکروکنترلر ATMEGA2560 نیز دارای یک EEPROM درونی است. که البته در این آزمایش مورد توجه ما نیست.

گونه‌های دیگر از EEPROM ها بیرونی هستند که بر روی یک IC جدگانه قرار می‌گیرند. برای ارتباط با این حافظه‌ها اکثراً از پروتکل‌های ارتباط سریال استفاده می‌شود. حافظه AT24C02 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 این‌که 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).

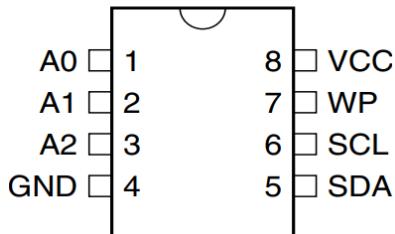
نمای بیرونی و پایه‌های AT24C02

در حافظه‌ای که در این آزمایش با آن کار می‌کنیم، سه پایه برای آدرس قطعه وجود دارد که در نتیجه ۲^۳ قطعه از آن را می‌توان بر روی یک باس مشترک قرار داد.

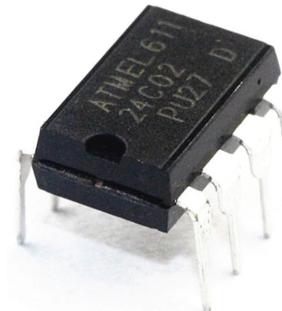
Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply

جدول کارکرد پایه‌ها

8-lead PDIP



نمایی از پایه‌های AT24C02



تصویری از AT24C02

پرسش: اگر یک حافظه EEPROM بیرونی دارای 4KB حافظه و 2 پایه آدرس باشد، در این صورت می‌توان حداقل چند KB حافظه EEPROM بیرونی بر روی یک باس مشترک داشت؟ 3

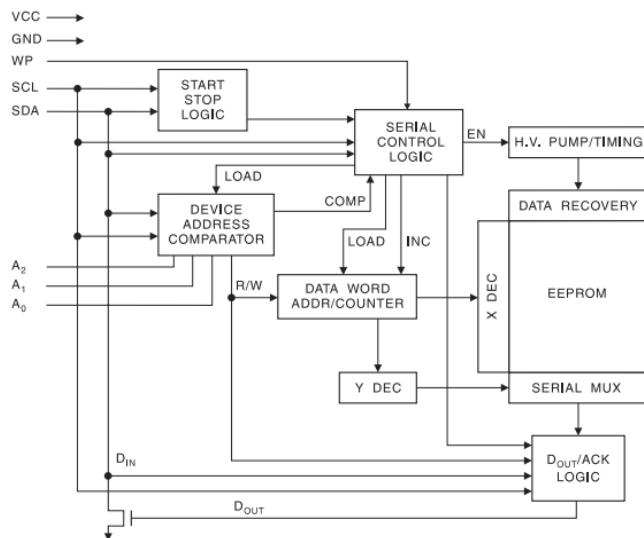
AT24C02 از پروتکل TWI بهره می‌برد؛ با این تفاوت که به جای 7 بیت، تنها 3 بیت برای آدرس دهی اختصاص یافته (A0-A3) و 4 بیت دیگر (پر ارزش‌تر) مقداری ثابت و برابر با 0b1010 دارند.

کارکرد دیگری که معمولاً در EEPROM‌های بیرونی وجود دارد، حفاظت نوشتن (Write Enable و Write Protection) است. این ویژگی به این صورت است که هنگامی که سطح منطقی پایه WP برابر با 0 باشد، خواندن و نوشتن به صورت عادی انجام شدنی است ولی هنگامی که مقدار ولتاژ منطقی آن برابر با 1 باشد، نوشتن غیر فعال می‌گردد.

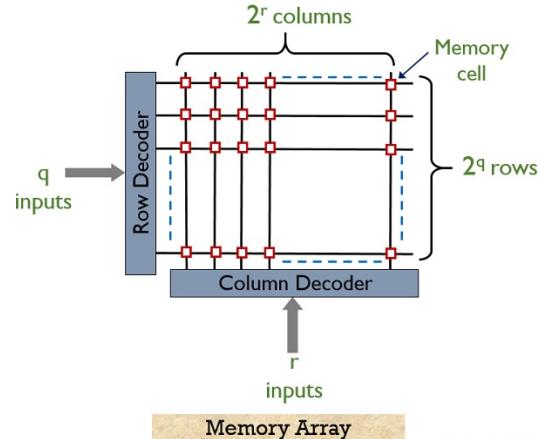
پرسش: نمودار شماتیک برای این‌که دو AT24C02 را به یک باس مشترک وصل کنیم و حفاظت نوشتن غیر فعال باشد را رسم کنید. (آدرس دهی سخت‌افزاری دلخواه - باس را هم به پایه‌های میکروکنترلر متصل کنید) 4

ساختار درونی AT24C02

بلوک دیاگرام حافظه AT24C02



ساختار عمومی ماتریس حافظه



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

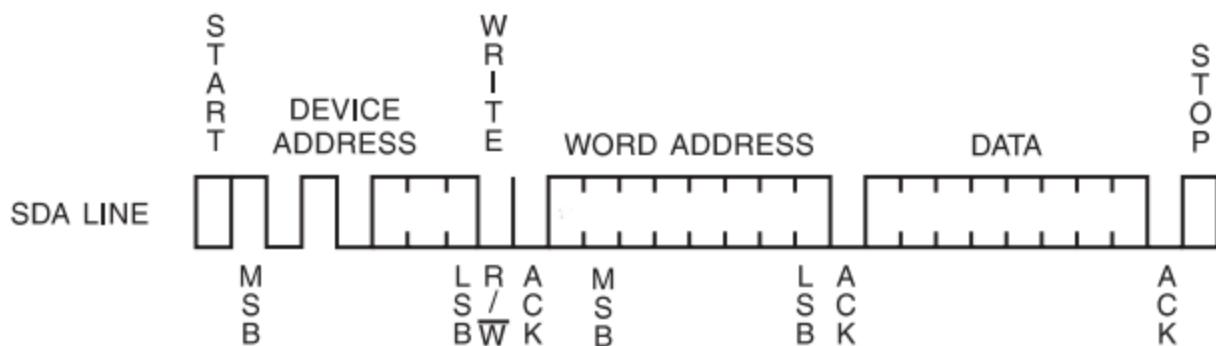
از این رو، برای اینکه بتوان از یک خانه حافظه خواند باید نخست با انجام یک عملیات خواندن نیمه کاره (Dummy Write)، این شمارنده را دوباره مقدار دهی کرد تا خانه مورد نظر در ماتریس حافظه انتخاب گردد.

عملیات نوشتن در یک خانه حافظه در بیشترین حالت 5 میلی ثانیه طول خواهد کشید. (اگر در این مدت و پیش از به پایان رسیدن عملیات درخواست تازه‌ای برسد، حافظه در پاسخ (NACK) بر می‌گرداند و در نتیجه عملیات تازه انجام نخواهد شد).

پیام‌های عملیات خواندن و نوشتن

در زیر دنباله فریم‌های پیام برای عملیات خواندن و نوشتن باقی نشان داده شده است:

نوشتن باقی



خواندن بایت تصادفی

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

پرسش: همچنانی این دنباله فریم‌ها را با پروتکل TWI بررسی کنید. (فریم‌های آدرس و داده را مشخص کنید، دستور خواندن یا نوشتن چگونه مشخص می‌شوند؟)

5

مشخصات تغذیه و فرکانس SCL

این قطعه ولتاژ تغذیه 2.7 تا 5 ولت را پشتیبانی می‌کند، به طور کلی بیشینه فرکانس TWI که قطعه می‌تواند در آن به درستی کار کند با افزایش ولتاژ تغذیه افزایش می‌باید. به طور کلی فرکانس 100KHz 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 نوشتن، با فرض این‌که کلک را 10KHz تنظیم کرده باشیم، در این صورت حداقل با چه نرخی می‌توان عملیات نوشتن را انجام داد؟

6

کتابخانه TWI در Arduino

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 نوشتن و خواندن گفته شده (با این تابع‌ها) بنویسید.

7

شرح آزمایش

در این آزمایش می‌خواهیم یک طراحی ساده از یک ماشین لباس‌شویی انجام دهیم. در هر ماشین لباس‌شویی چهار گام کلی 1- پیش‌شستن با شوینده 3- شستن با آب 4- خشک کردن برای شستن لباس‌ها انجام می‌شود. در این آزمایش این‌که دستگاه در کدام یک از این گام‌ها است را با روش بودن یک LED نشان می‌دهیم. هر یک از این گام‌ها می‌توانند مدت زمان قابل تنظیم داشته باشند.

دستگاه باید مد 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 تعذیب نیز در دستگاه ذخیره شده بماند.

کاربر باید بتواند از میان مدهای دستگاه یکی را انتخاب کرده و توانایی انجام دستور‌های آغاز و وقفه را داشته باشد. زمان مانده تا پایان فرآیند 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 فرآیند شستشو را ادامه دهد.

پایان فرآیند نیز با روش 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.



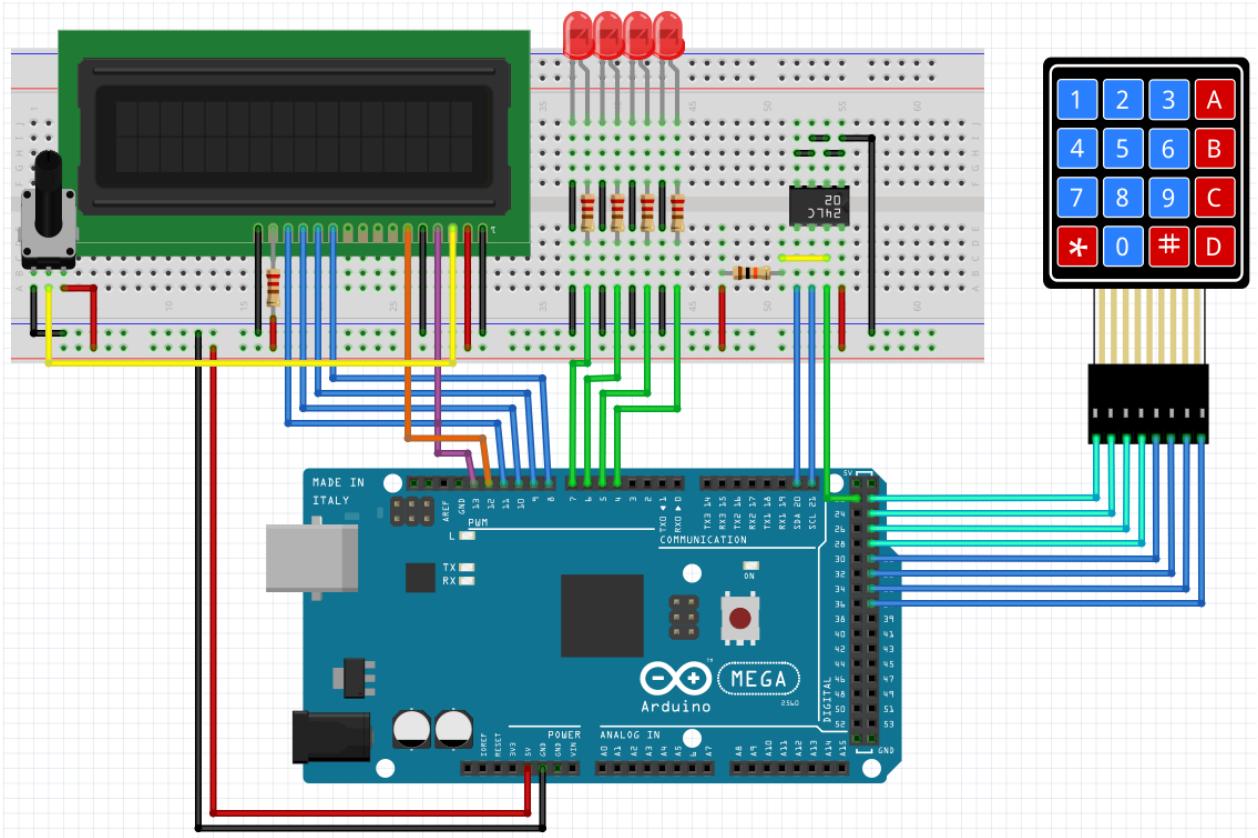
تمرین بیشتر: (تمرین‌های زیر اجباری نیستند، نباید جزوی از آزمایش در نظر گرفته شوند)

- با اسیلوسکوپ موج TWI فرستاده شده به EEPROM را تحلیل کرده و فریم‌های داده و آدرس را مشخص و همچو ای آن‌ها با تنظیمات انجام شده در برنامه را بررسی کنید. در برنامه آدرس سخت‌افزاری EEPROM را تغییر دهید و نتیجه را بررسی کنید.

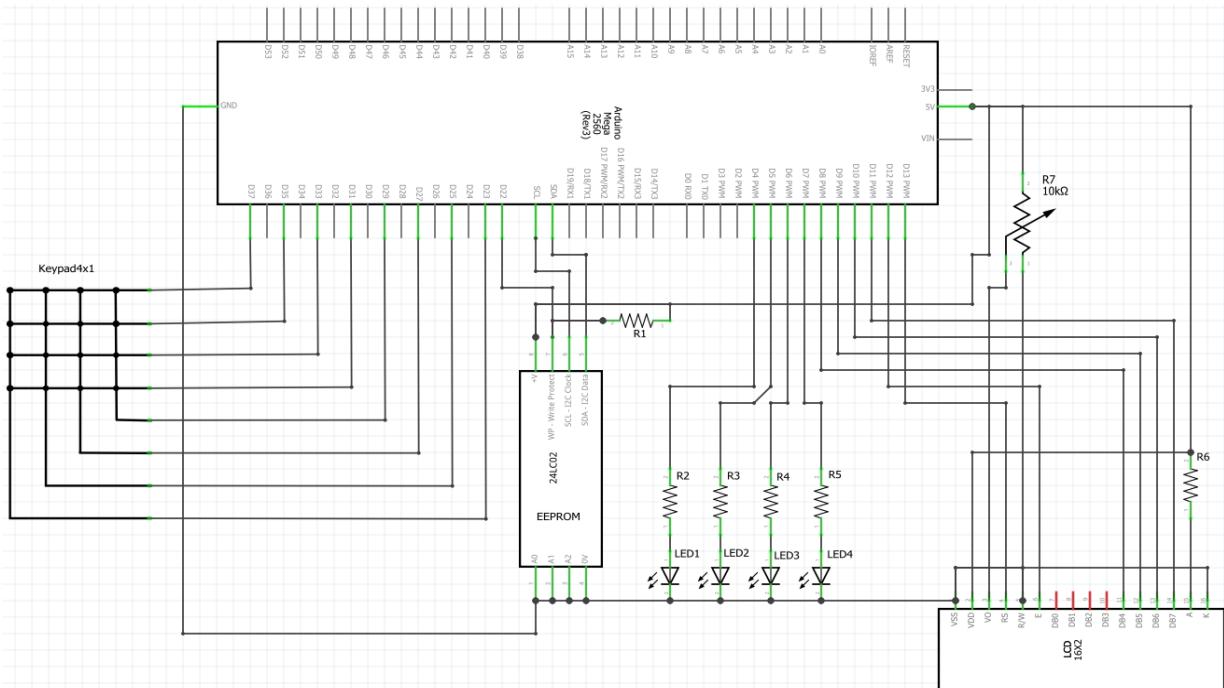
- طراحی را به گونه‌ای تغییر دهید که دو یا چند EEPROM دیگر نیز به باس متصل شده، داده‌ها دسته‌بندی شوند و بر روی EEPROM متناظر با خود ذخیره گرند.

- در صورت موافقت سرپرست آزمایشگاه، دو گروه برد‌های خود را با یکدیگر 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 خروجی‌هارا با یکدیگر پکارچه کنند.

- در صورت موافقت سرپرست آزمایشگاه، دو گروه برد‌های 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 (یا Interrupt-Driven) بررسی کنید. (راهنمایی: تابع‌های (`onRequest()` و `onReceive()`) بخشی از رابط کاربری کتابخانه Wire است که به روش وقفه محور واحد TWI را کنترل می‌کند.)



نمودار مدار آزمایش بر روی برد



نمودار شماتیک آزمایش

آزمایش 8 : اتاق تحت کنترل

هدف آزمایش:

آشنایی با پروتکل SPI

تحلیل موج خروجی آردوینوی مرکزی (master)

راه اندازی حسگر نور و دما

قطعات مورد نیاز:

- Arduino Mega 3 عدد برد مقاومت متغیر فتوسل (ldr) سنسور دمای Lm35

آنچه پاید در پیشگزارش نوشته شود:

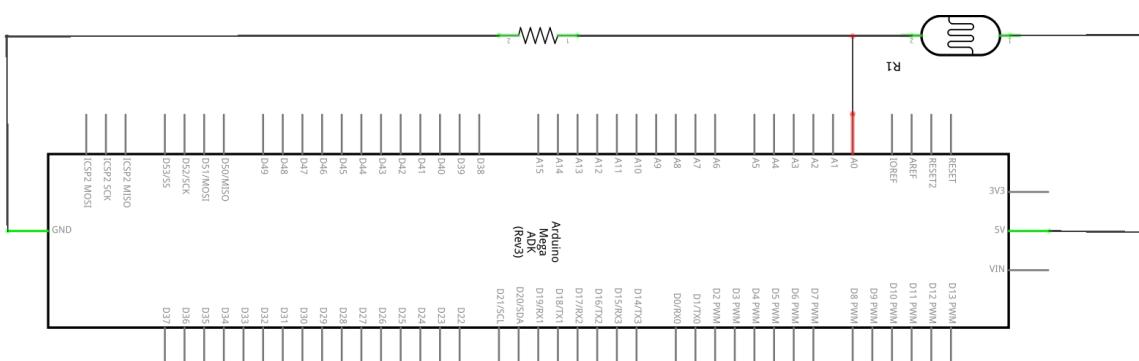
- به پرسش‌های درون مقدمه پاسخ داده شود.

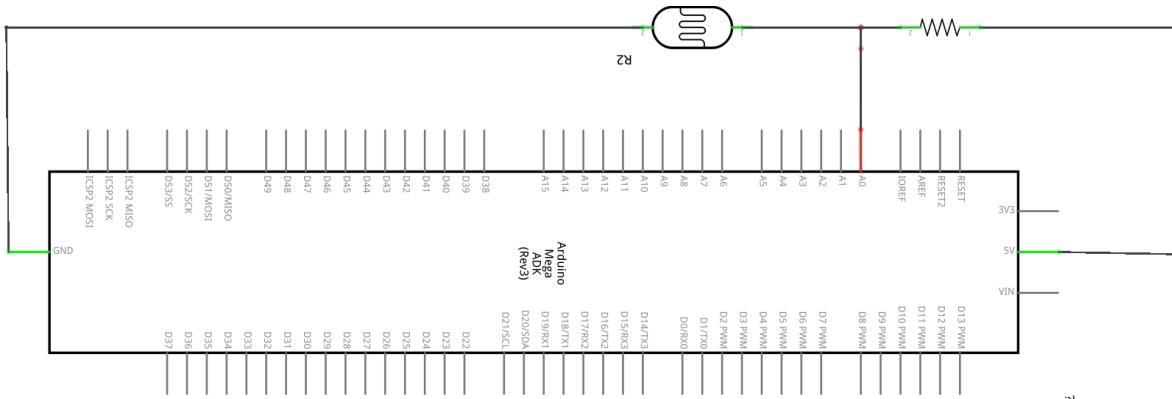
۱۰۷

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 ممکن 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 lm35 می‌پردازیم و در ادامه پروتکل SPI را بررسی می‌کنیم.

راه اندازی سنسور میزان روشنایی :

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 بیشتر.





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



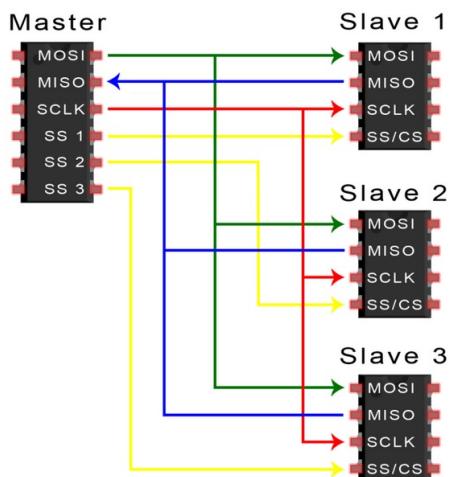
راه اندازی سنسور دما : Im35

سنسور دما، میزان دمای محیط را بر حسب درجه سانتی‌گراد به ولتاژ آنالوگ تبدیل می‌کند.

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

راه اندازی ارتباط SPI در حالت چند برده ای :

همان‌طور که می‌دانید این نوع ارتباط از نوع ارتباطات (master/slave) است. در این نوع ارتباطات یک دستگاه می‌تواند با چند دستگاه دیگر ارتباط برقرار کند. در ارتباط SPI بورد مرکزی (master)، برده که می‌خواهد با آن ارتباط برقرار کند را انتخاب کرده و برای آن پیامی ارسال می‌کند و در صورت نیاز، از آن درخواست پاسخ می‌کند.



در این ارتباط می‌بایست سه پایه MOSI (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 کلاک مرکزی تمام دستگاه‌ها است)، در تمام دستگاه‌ها 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 Slave‌ها یک پایه خاص است. ولی در دستگاه مرکزی (master) می‌توان آن را به تعداد Slave‌ها تعیین کرد.

پرسش: در مورد پایه‌های MOSI، SCLK، SS و MISO تحقیق کنید. پایه‌ی پیشفرض برای SS کدام پایه است؟ برای مشاهده آن می‌توانید به محل نصب آردوینو رفته، مسیر زیر را دنبال نمایید و در انتهای فیل داخل پوشه را باز نمایید:

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

پرسش : در مورد نحوه انتخاب برد **SPI** توسط **Slave** تحقیق نموده و نحوه پیاده‌سازی برنامه را برای اینکه برد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از بردات **Slave** داده ارسال کند، شرح دهد. (برای این‌کار 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.

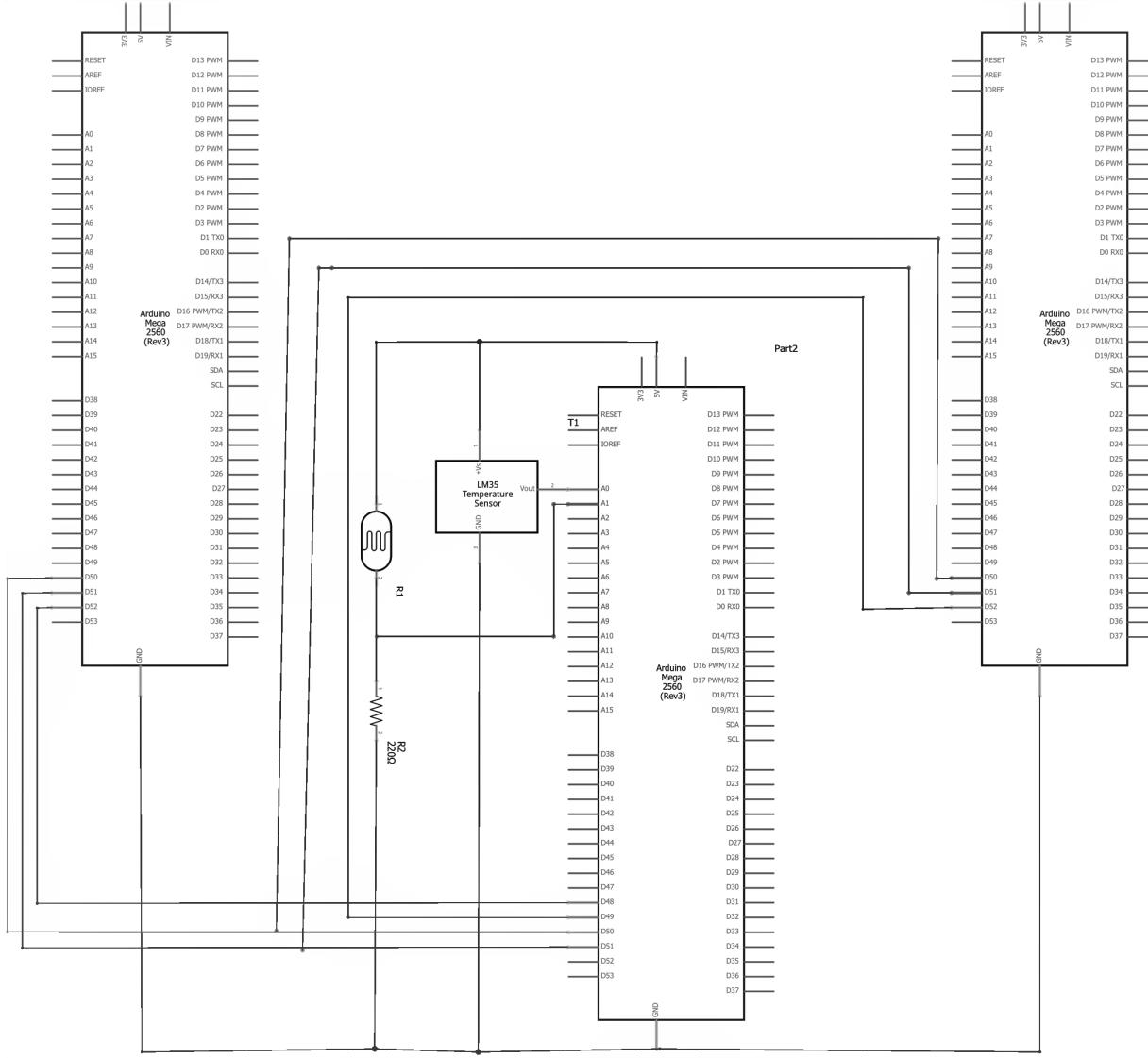
پرسش : دستور مورد نیاز تا آردوینو در حالت **Slave** قرار گیرد را نوشته و در مورد کارایی آن تحقیق نمایید.

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.

1. 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 برد **Slave** ارسال می‌کند. حتما پایه **SS** در آردوینو **master** را پایه‌ای به جز پایه پیش‌فرض آردوینو قرار دهید.
2. موج خروجی سه پایه **SCLK**، **MOSI**، **SS** را برای سه مقدار **Clock** توسط اسیلوسکوپ مشاهده کنید.
3. 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 "your" name را دریافت کند.
4. داده‌های **Hello world** و **Hi** را با اطلاعات مربوط به دو سنسور دما و نور جای‌گزین نمایید. برای خواندن ولتاژ خروجی هر یک از سنسور ها کافیست از دستور **analogRead** استفاده نمایید. سپس عدد به دست آمده را به بازه مناسب **map** نمایید.
5. موج خروجی را برای چهار پایه **SCLK**، **MOSI**، **SS1**، **SS2** توسط اسیلوسکوپ مشاهده نمایید.



آزمایش ۹: موسیقی و header

هدف آزمایش:

آشنایی با عملکرد و نحوه کار با اسپیکر های piezo
استفاده از فایلهای header در برنامه نویسی آردوینو
پیاده سازی هر گونه موسیقی بر روی میکرو کنترلر

قطعات مورد نیاز:

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

آنچه باید در پیش کارش نوشته شود:

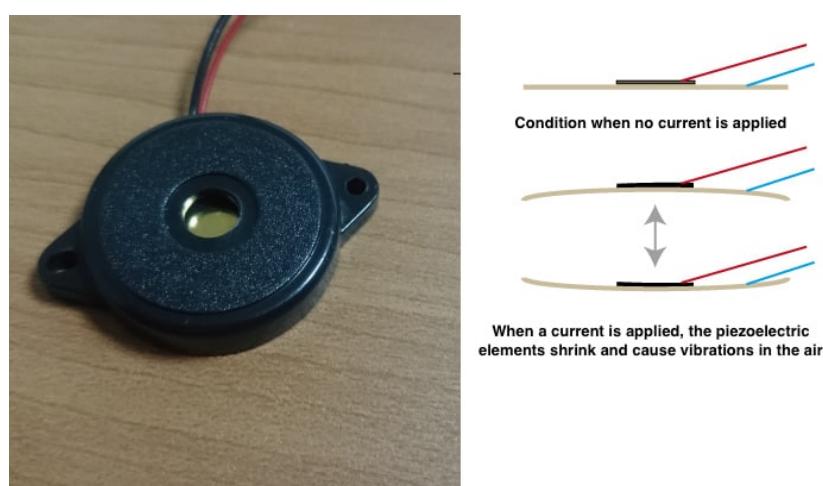
- به پرسش های درون مقدمه پاسخ داده شود.

مقدمه:

اسپیکر پیزوالکتریک، از پدیده‌ی پیزو الکتریک (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 ارزان‌مانند رادیوهای پرتابل و لوازم خانگی کاربرد دارد. این طراحی نسبت به بقیه طرح‌های بلندگو بسیار ساده‌تر 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؟



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، صدای توافقی 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 تایمر داخلی برد میسنجد، پس اگر نت های مشخص میخواهید، بعد از دستور، به همان اندازه `delay` یک `duration_in_milliseconds` قرار دهید.

پرسش: تایمرباکس دستور `tone` استفاده میکند با خیلی از پین های برد مشترک است. بررسی کنید که به چه روش هایی میتوانید آن تایمرباکس را به هم پریزید که دستور `tone` خراب شود و صدای مطلوب را اجرا نکند.

تئوری موسیقی:

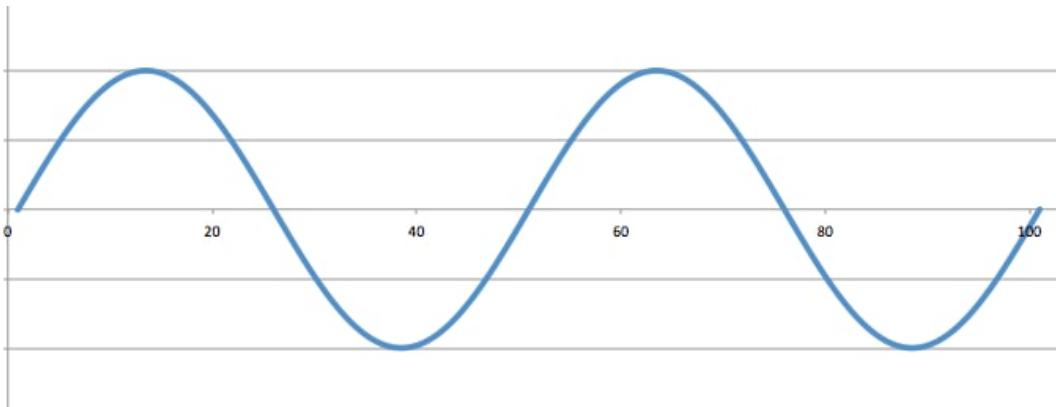
اسپیکر های `piezo` برای اجرای فرکانس های کمتر از ۳۱ هرتز نیستند. این قضیه مشکل زیادی ندارد، چون گوش انسان از ۲۰ هرتز شروع به شنیدن میکند. پایین ترین اکتاو موسیقی (هر بار که تمامی نت های موسیقی را از پایین به بالا میشمارند یک اکتاو نامیده میشود) با فرکانس ۳۲ هرتز شروع میشود که دقیقاً بالای حداقل `piezo`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

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 سیون یک اکتاو را نشان میدهد که نت ها (و بعض از نت های نت ها) با فرکانس مربوطه شان نشان داده شده اند. دقت کنید که نت های B و C و نت های E و F بین شان # 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 A4 است. به این معنا که ۴۴۰ هرتز / ۴۴۰ نوسان در ثانیه در هوا و اسپیکر piezo ساخته می‌شود.

یک ملodi، یک سری نت است که در یک ترتیب خاص و به مدت معلوم نوخته می‌شود. مثلًا ملodi زیر را در نظر بگیرید:

Jingle Bells

TRADITIONAL
arr. A.L.Christopherson

Piano

این ملodi را می‌توانیم به عنوان یک سری از نت‌های نوخته شده بخوانیم. در این مثال به ترتیب EEEEEE GCDE. ولی هر نت ارزش زمانی خاصی دارد که به آن اندازه کشیده می‌شود. می‌توانیم نت‌های توپر (سیاه) را به عنوان استاندارد در نظر بگیریم. عدهای نوشته شده اول نت 4/4 به ما می‌گوید در هر جعبه، ما اندازه‌ی چهار نت سیاه زمان سپری می‌کنیم. پس از جعبه اول همچنین می‌توان فهمید که هر نت سفید دو برابر یک سیاه کشیده می‌شود، 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

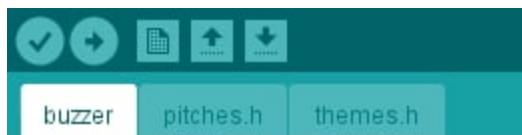
این جدول برای دیدن ارزش‌های زمانی پر کاربرد است. البته اینجا نت دایره معیار قرار داده شده است. سفید، یک دوم دایره است. سیاه یک چهارم، چنگ یک هشتمن دایره است. همچنین دو لامپ یک شانزدهم دایره است.

برای نواختن ملودی های این درس، یک تکنیک دیگر در نت خوانی 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 اندازه های 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 ...

همچنین این جدول علامت های نت های سکوت را به ما نشان می دهد. مثلاً نماد سکوت نت سیاه به این معناست که اگر این علامت را دیدیم، به اندازه مدت زمانی یک سیاه هیچ چیزی نمی نوازیم. و اگر آن علامت با یک نقطه بود، به اندازه یک سیاه و نیم هیچ آهنگی نمی نوازیم.



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

تاکنون در آردوینو از کتابخانه ها استفاده کرده ایم. برای LCD و Servo، و اخیرا 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 دکمه **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 کرده ایم. برای جای گذاشتن در فایل اصلی، به عنوان هدر می گذاریم. مثلاً یک فایل **pitches.h** که در آن نت ها و فرکانس های مربوطه را آن جا ثبت می کنیم، و یک فایل **themes.h** برای ملودی ها. فایل **pitches.h** در اختیار شما قرار می گیرد (فقط شامل یک سری **#define** است که اسم نت ها را به فرکانس های مربوطه می کند، همان طور که خودتان از جدول فرکانس ها در بالا می توانید 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
```

ions[] = {

8,

برای مثال، در پیاده سازی ملودی **Jingle Bells** که در بالا دیدید، می توان برای راحتی کار این دو آرایه را در **themes.h** ذخیره کنید. آرایه **notes** شامل اسم نت ها (که با خاطر **#define** های **pitches.h** با فرکانس برای tone استفاده می شوند) و آرایه **durations** شامل ارزش زمانی است. ارزش زمانی بر مبنای نت دایره حساب شده است، و همه ای این **duration** ها کسری از دایره هستند. برای مثال 2 یعنی یک دوم دایره، پس سفید. 4 یعنی یک چهارم دایره، پس سیاه. 6 یعنی یک سیاه و نصف، پس یعنی دایره نقطه دار.

نحوه آزمایش

1. در پروتئوس یک دکمه و یک اسپیکر و یک صفحه کلید را به برد وصل می کنیم. برنامه ای بنویسید که به هنگام فشرده شدن هر کلید فایل ملودی متناظر با آن را پخش کند.

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

2. 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 کلک آن را پابین بیاورید تا به جایی 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.
3. یک پتانسیومتر به برد وصل کنید. همان‌طور که قبلاً تابع map ورودی آنالوگ را استفاده کردید، اکنون برنامه‌ای بنویسید که هر دفعه که دکمه فشرده می‌شود، بنا به وضعیت پتانسیومتر یک نت زیرتر یا بمتر اجرا کند.
4. با دانشی که دارید، برنامه‌ای بنویسید که ملودی Ode to Joy بتهوون را پخش کند.

N BEETHOVEN
(1770-1820)
Op.125
Christopherson



5. ملودی امتیازی: (همان # است ولی بجای یک قدم جلو، یک قدم عقب. ۲ نت را عادی می‌کند. ۳ های اول خط تا آخر خط باقی می‌مانند، مگر اینکه با ۴ عادی سازی شوند)

The Imperial March

Music by
John Williams

آزمایش 11: پروژه نهایی

هدف آزمایش:

در طول این ترم، شما با مفاهیم اساسی کار با میکروکنترلر، و مازول های جانبی پایه ای آن آشنا شدید. برای اختتم این آزمایشگاه و تثبیت دانسته هایتان در یک پروژه نهایی که می تواند تجلی 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 مسئول آزمایشگاه خود هماهنگ کنید، و در رابطه با پیچیده بودن آن، ایشان را قانع کنید. و گرنه دو تا از پروژه های پیشنهادی زیر را میتوانید پیاده سازی کنید. توجه داشته باشید که پروژه خلاقانه می تواند به اندازه ی صلاححید استاد، نمره اضافی هم در برگیرد. استفاده از مازول 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 دهد 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 بنوایند بین چند مود (لایک و دیسلایک یا بله و خیر ، رای دادن از پنج 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. وقت 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 ثبت شده است" چاپ شود.

قطعات مورد نیاز:

- پنج عدد بورد Arduino Mega2560
- 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 بگیرید. دست راست با سه دکمه K O M نواختن سیم سیم 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 از آنکه 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. هر چه بیشتر چرخانده شود، نت های سیم به تر و زیر تر می شود.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- 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 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 متناسب با دما باشد.

اگر مخزن در یک بازه 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 برقی، 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 پالآمدن میله ها، واکنش دوباره از سر گرفته و آژیر و ال ای دی خاموش می شود.

قطعات مورد نیاز:

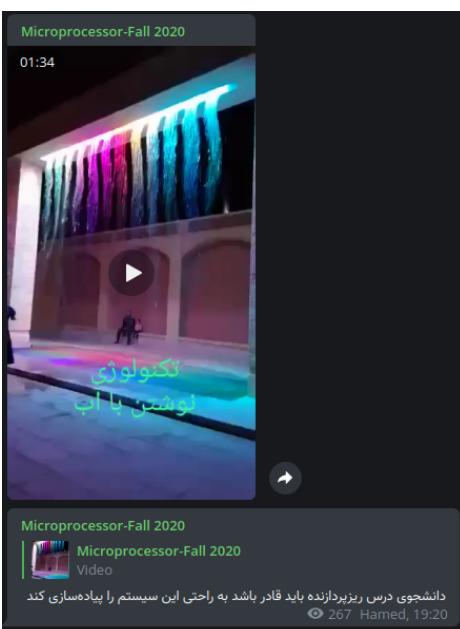
- حداقل سه بورد Arduino Mega2560
- حرارت سنج (برای آزمایش، راحت تر و بهتر است که حرارت در نرم افزار شبیه سازی شود)
- تعداد زیادی سرو و موتور
- بازر و ال ای دی به مقدار کافی
- LCD کاراکتری

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 ببینند LED بالایی 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 ای که گرفته شود یا چند کاراکتری که جا شوند را، بصورت سرشار شونده روی LED ها پدیدار کند.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- بازر و ال ای دی به مقدار کافی



ربات سوپرمارکت (ربات کارتزین)

هدف پژوهه

یک سیستم هایپر مارکت رباتیک به همراه رابط کاربری مثل صد عدد کالای مختلف در قسمه هایی که شامل ده سخنون 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 متی 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:

کالاها باید دارای کد منحصر بفرد باشند و هر کالا هم در یک خانه به خصوص از قسمه فروشگاهی قرار دارد. با وارد کردن کد هر کالا یک سبد خرید توسط یک ربات کارتزین به نقطه مختصات مربوط به کالای درخواستی می رود و توسط یک سروو موتور به تعداد کالای مورد نظر سروو حرکت می نماید و پس از پایان خرید به نقطه مختصات صفر و صفر باز می گردد و سبد را به مشتری تحویل می دهد.

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

قطعات مورد نیاز:

- بورد Arduino Mega2560
- استپر موتور
- تعداد زیادی سروو موتور
- کیبورد و LCD کاراکتری

● بازر و ال ای دی به مقدار کافی

پ