



دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

دستور کار آزمایشگاه ریزپردازنده

(مهر 1399)

تغییر و افزایش:

بهداد منصوری

علیرضا صالحی

امیرحسین شیخ الاسلامی

فهرست مطالب

1	قوانين آزمایشگاه ریزپردازنده
2	آشنایی با برد Arduino MEGA 2560
4	نرم افزار Arduino IDE
5	نرم افزار شبیه سازی Proteus
7	آزمایش 1: آشنایی با برد Arduino MEGA2560 و نصب برنامه های مورد نیاز
10	آزمایش 2: کیبورد ورودی و ارتباطات سریال
14	آزمایش 3: مانیتور خروجی
17	آزمایش 4: راه اندازی سرو و موتور و ورودی آتالوگ
21	آزمایش 5: راه اندازی رله با Arduino MEGA2560
27	آزمایش 6: نیم پروژه آردوینوهای همکار
27	کامپیوتر ماشین کنترلی
28	اجاق خورشیدی
28	یعقوب برقی رایگان!
29	پنکه
30	گاوصندوق
31	آزمایش 7: ماشین لباس شویی ساده
38	آزمایش 8: اتاق تحت کنترل
42	آزمایش 9: موسیقی و header

قوانین آزمایشگاه ریزپردازنده

به منظور افزایش کارایی درس آزمایشگاه ریزپردازنده، رعایت عدالت میان همگی 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 سر ساعت در کلاس آنلاین حاضر شوند.
2. در تمام مدت زمان کلاس دانشجویان موظفند در کلاس آنلاین بوده و بنا به صلاححید مدرسین وبکم و میکروفون خود را فعال کرده و همچنین صفحه سکتابخ خود را به اشتراک بگذارند.
3. غیبت در کلاس‌های آنلاین به هیچ عنوان مجاز نیست.
4. اگر دانشجویانی نتوانند در طی بک جلسه آزمایشی را تمام کنند، در طول هفته موظفند تا آزمایش را تکمیل کنند و پیش از آغاز جلسه بعد نتیجه را به مدرس خود تحويل دهند. حداکثر تعداد جلساتی که به درازا کشیده می‌شود ۱ جلسه در طول نیمسال است.
5. آزمایش‌ها در گروههای تک نفره انجام می‌شوند و میزان فعالیت افراد تعیین کننده نمره‌ی 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 مدرس تحويل داده شود. پیش‌گزارش مطلوب هر آزمایش در دستور کار آمده است.
7. ارائه گزارش کار آزمایش‌ها الزامی نیست و تحويل 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" باید با دقت پاسخ داده شود؛ چراکه 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 باشد. 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 بعداً دچار مشکل نشوند.
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 تای آن می‌تواند به عنوان خروجی PWM استفاده شود)، 16 ورودی آنالوگ، 4 پورت‌های UARTs سریال ساخت‌افزاری، یک ساعت 16 مگاهرتزی، یک کابل ارتباطی USB، یک پاور جک، یک ICSP header، و یک دکمه ریست است.

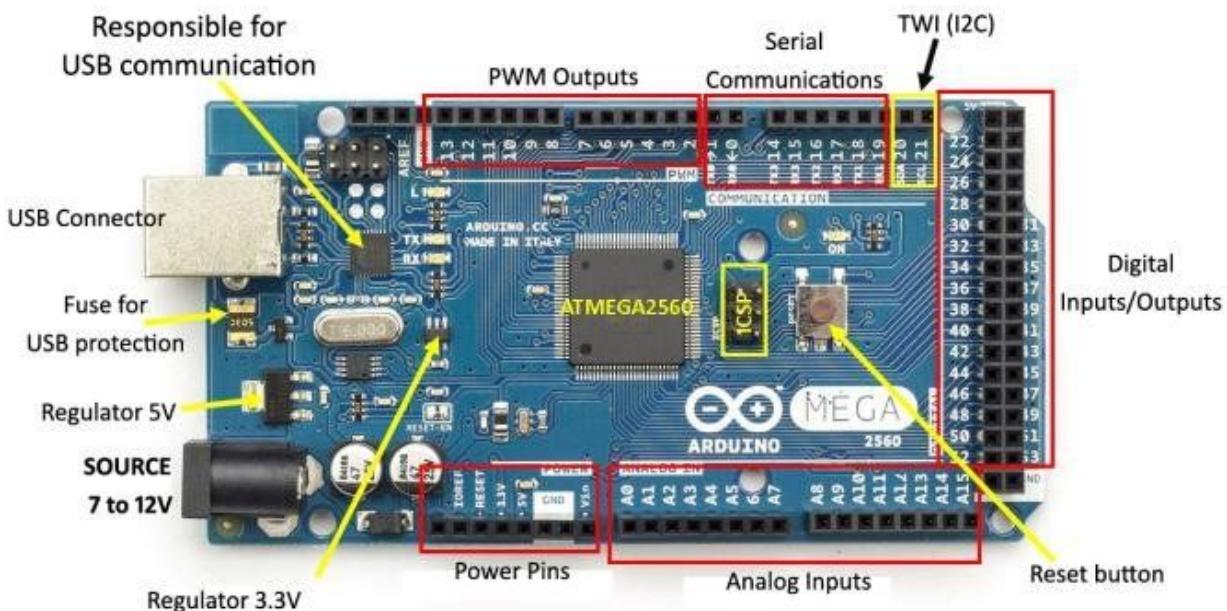
این برد عملایی چیزهایی که برای کار با میکروکنترلر نیاز است را دارد. فقط کافیست با USB به یک کامپیوتر متصل شود یا با آداپتور DC به برق بگیرد، یا این‌که از باتری استفاده کند تا روشن شود. مگا 2560 می‌تواند با اکثر شیلد‌های طراحی شده برای Uno و بردهای قدیمی Duemilanove or Diecimila کار کند.

بردهای Arduino MEGA 2560 همانند دیگر بردهای Arduino با برنامه IDE (Arduino Software IDE) نصب و اتصال برد به آن در پیوست آورده شده است.

مشخصات:

ATmega2560
5 ولت
7 تا 12 ولت
20-6 ولت
54 تای آن خروجی PWM تولید می‌کند.
16 عدد
20 میلی آمپر
50 میلی آمپر
256 کیلوبایت که 8KB برای bootloader است
8 کیلوبایت
16 مگاهرتز

- میکروکنترلر:
- ولتاژ عملیاتی:
- ولتاژ ورودی (پیشنهادی):
- ولتاژ ورودی (محدوده):
- 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

شانگرهای LED:

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

های LED RX و TX: بر روی برد دو بخش به نام‌های TX (ارسال) و RX (دریافت) وجود دارد. یکی در بخش پایه‌های 0 و 1 و 14 تا 19 که برای ارتباط سریال هستند و دومی LED‌های TX و RX (13) مربوط به TX هنگام ارسال داده سریال مناسب با میزان سرعت چشمک می‌زند. میزان سرعت چشمک زدن به baud rate برد بستگی دارد. RX هنگام دریافت داده چشمک می‌زند.

تغذیه:

ولتاژ مورد نیاز Arduino MEGA 2560 می‌تواند از طریق اتصال USB و یا یک منبع تغذیه خارجی تغذیه گردد. منبع تغذیه به صورت خودکار انتخاب می‌شود.

منبع تغذیه خارجی غیر از USB می‌تواند آداپتور AC به DC یا باتری باشد. آداپتور (که سوکت آن از نوع center-positive با قطر 2/1 میلی متری باشد) می‌تواند به پاورجک موجود بر روی برد متصل گردد 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، روى ولتاژ پین‌ها نیز اثر خواهد گذاشت و ممکن است ولتاژ خروجی آن‌ها کمتر از 5 ولت شود، و باعث نوسان گردد. ولتاژ بیش از 12 ولت نیز می‌تواند موجب افزایش دمای رگولاتور و در نتیجه آسیب برد گردد. ولتاژ پیشنهادی مناسب بین 7 تا 12 ولت است.

پین‌های مربوط به منبع تغذیه به شرح زیر است:

- V_{IN} : این پین، پین ورودی ولتاژ آردوینو است که در موقع استفاده از منبع تغذیه خارجی (به جای منبع تغذیه تنظیم شده یا اتصال 5 USB ولتی) 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(5 ولت) USB و یا پین V_{IN} برد (7-12 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 میلی آمپر است.
GND: پین‌های اتصال به زمین.

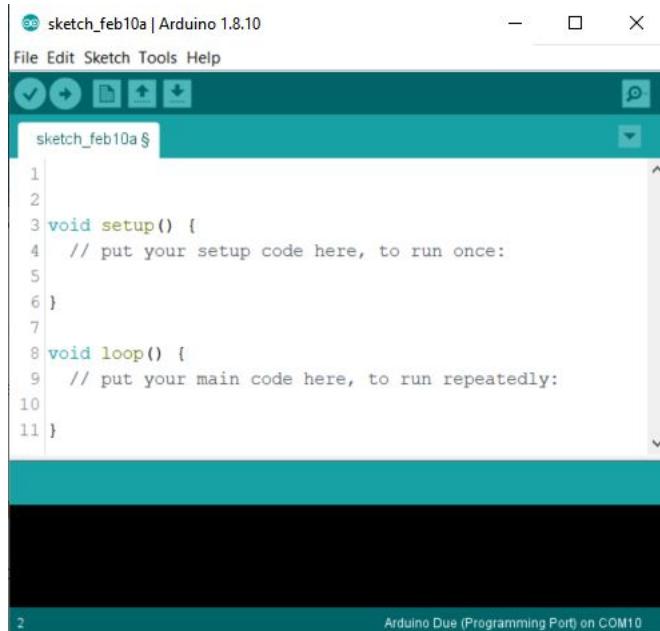
- IOREF: این پین میزان ولتاژ مرتعی که میکروکنترلر با آن کار می‌کند را مشخص می‌نماید. این پین اجازه می‌دهد یک شیلد را با پیکربندی مناسب، جهت تطبیق با ولتاژی که توسط برد فراهم شده است، به برد متصل کنید. یک شیلد که به درستی تنظیم شده باشد، می‌تواند مقدار ولتاژ را از پین IOREF خوانده، منبع تغذیه مناسب خود را انتخاب نماید و یا این‌که مبدل‌های ولتاژ را برای کار کردن با ولتاژ‌های 5 ولت یا 3/3 ولت، بر روی خروجی‌ها فعال نماید. این قابلیت، به شیلد‌ها امکان می‌دهد تا برد 3.3 ولتی همچون DUE و برد‌های AVR-based که با ولتاژ 5 ولت کار می‌کنند، خود را تطبیق دهند.

نرم افزار Arduino IDE

جهت برنامه نویسی بردهای آردوینو و همچنین ارتباط سریال با برد از برنامه Arduino IDE استفاده می شود. این برنامه متن باز و رایگان است که هم می توان نسخه دسکتاپ آن را نصب کرد و یا می توان از نسخه برخط آن در آدرس زیر برای نوشتن برنامه و آپلود آن بر روی برد استفاده کرد.

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

نحوه نصب برنامه و اتصال برد و نصب راه انداز برد Arduino Mega2560 در پیوست به طور کامل آموزش داده شده است. شکل (4-1) محیط برنامه Arduino IDE را نشان می دهد.



محیط برنامه Arduino IDE

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 پروژه‌ی جدید.
- Open: کلید میانبر برای باز کردن نمونه پروژه‌های موجود در نرم افزار.
- Save: ذخیره‌ی پروژه‌ی ایجاد شده.
- Serial Monitor: ترمینال سریال برای دریافت دیتای پورت سریال از برد و ارسال اطلاعات به برد.

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.

برنامه نویسی در آردوینو 3 بخش کلی دارد:

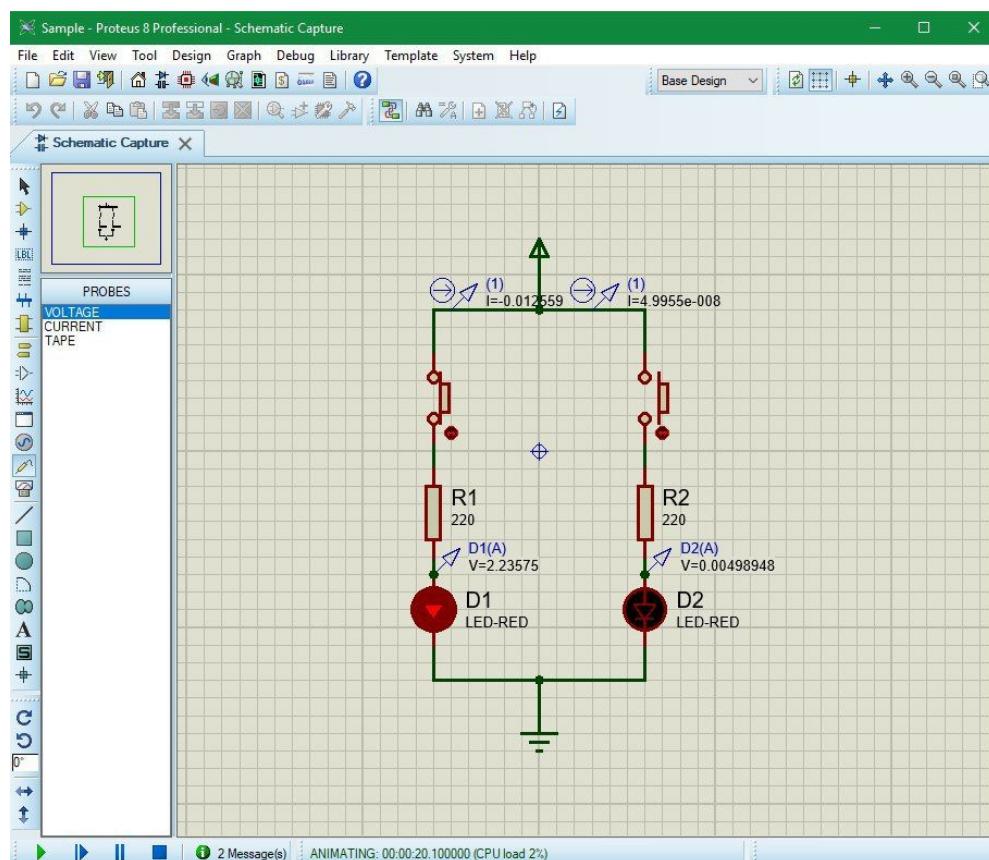
1. پیش‌پردازش‌ها: در بخش پیش‌پردازش، کتابخانه‌ها و یا متغیر‌هایی که محلی نیستند تعریف و فراخوانی می‌شوند.
2. حلقه setup: در بخش setup کارهایی که باید یکبار انجام شوند را تعریف می‌کنیم. به طور مثال، تنها کافی است یکبار تعریف کنیم که یک پین دیجیتال ورودی باشد یا خروجی و یا این‌که به برد اطلاع دهیم که می‌خواهیم از پورت سریال استفاده کنیم.

3. حلقه loop: حلقه بی‌نهایت است و کارهایی باید که همیشه 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 و نوشتند بر روی صفحه نمایش را باید در این حلقه بنویسیم.

برای گرفتن فایل hex از Arduino IDE که به Proteus می‌توانید Sketch از تب Binary می‌توانید فایل باینری کامپایل شده برای پروتوس را بسازید که در Sketch Folder ذخیره می‌شود و می‌توانید با انتخاب Show Sketch Folder در زیر این گزینه، به آن دسترسی پیدا کنید.

نرم‌افزار شبیه‌سازی Proteus

این ترم به دلیل برگزاری مجازی آزمایشگاه، آزمایش‌ها در بستر شبیه‌سازی Proteus انجام خواهد شد. این نرم‌افزار مانند نرم‌افزار 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 این‌رو، می‌توان از این نرم‌افزار برای شبیه‌سازی آزمایش‌های درس ریزپردازندۀ بهره برد.



محیط برنامه Proteus

چگونگی انجام شبیه سازی آزمایش‌ها در محیط Proteus :

نخست مدار آزمایش را در نرم‌افزار Proteus طراحی کنید، سپس در محیط توسعه آردوینو در تابع `(setup()` پیکربندی ورودی/خروجی پایه‌ها را انجام دهید و در تابع `(loop()` منطق کنترلی برنامه را پیاده‌سازی کنید.

پس از آن، برنامه را برای ATmega2560 کامپایل کنید (با انتخاب برد از راه Tools -> Board -> Arduino Mega و سپس انتخاب گزینه Verify).

سپس 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.

همچنین راه دیگری نیز برای به دست آوردن فایل باینری در محیط Arduino IDE، به صورت انتخاب گزینه Export Compiled Binary در منوی Sketch برنامه کامپایل شده است که فایل HEX آن را در پوششی Sketch می‌ریزد و می‌توان با انتخاب گزینه Show Sketch Folder در همان منو به آن دست یافت.

آزمایش 1: آشنایی با برد Arduino MEGA2560 و نصب برنامه‌های مورد نیاز

هدف آزمایش:

آشنایی با برد Arduino MEGA2560 و نصب برنامه‌های مورد نیاز. (نرم‌افزار Arduino IDE و Proteus و راهاندازی آن‌ها)

قطعات آزمایش:

- بورد Arduino MEGA2560
- دیود نورانی LED
- کلید
- مقاومت 220Ω
- مقاومت 10KΩ

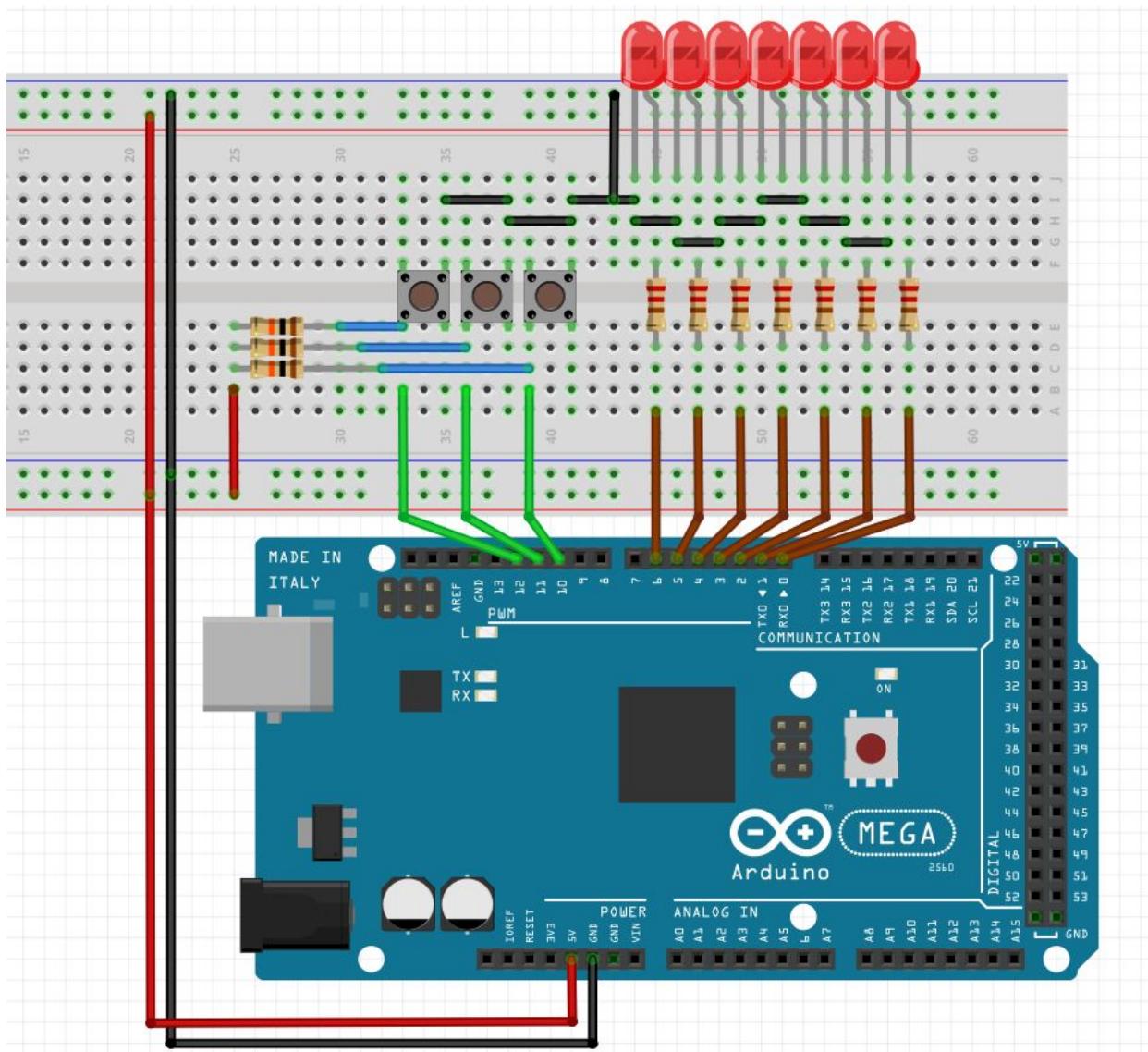
شرح آزمایش

این آزمایش در بردارنده چندین LED و سه دکمه است. در ابتدا LED ها خاموش هستند. با هر بار فشردن دکمه یک LED ها از سمت چپ یکی یکی روشن می‌شوند، با فشردن دکمه دو LED ها از سمت راست یکی یکی روشن می‌شوند. و با فشردن دکمه سوم همه LED ها خاموش می‌شوند.

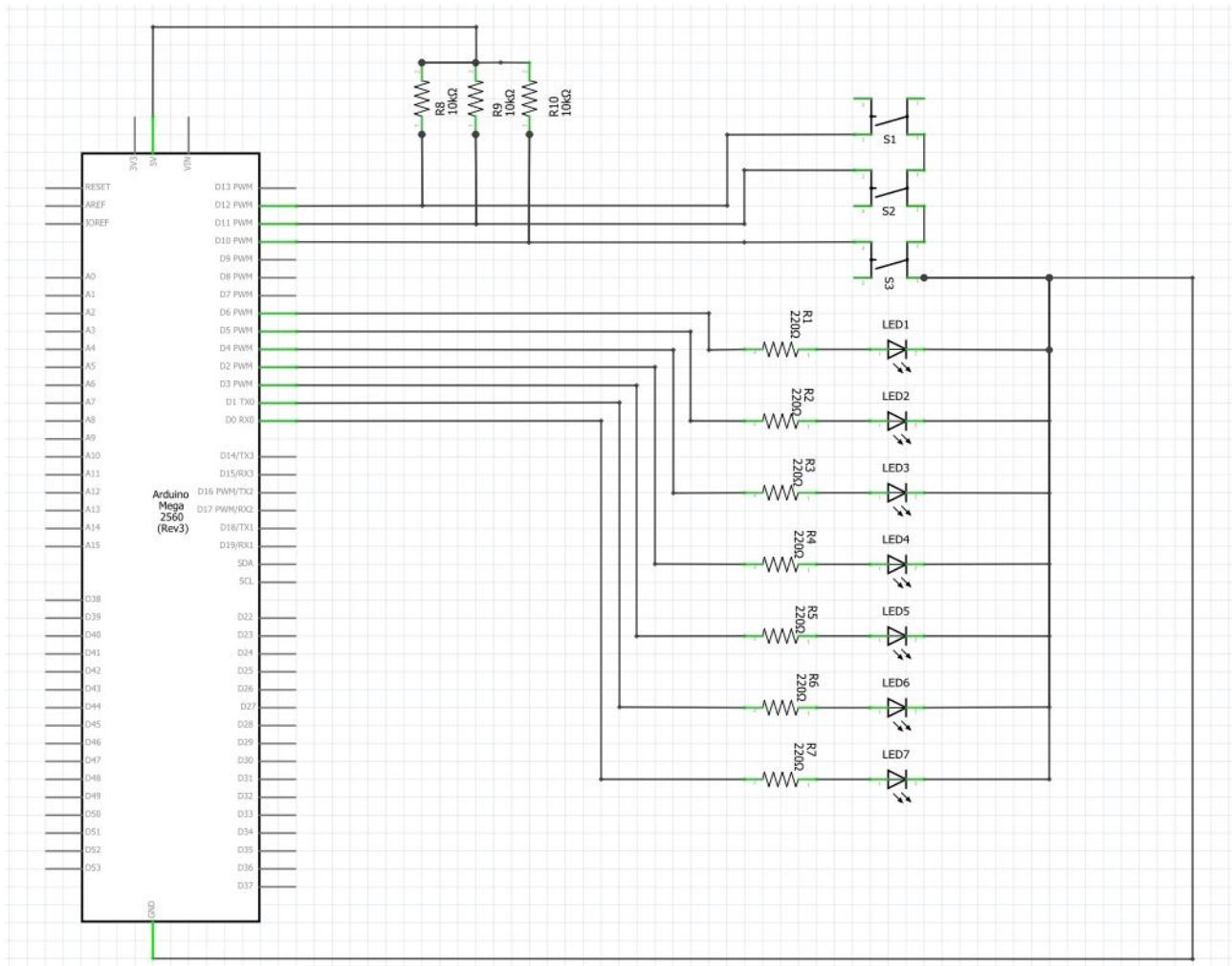
برای انجام آزمایش نخست مدار آن را در نرم‌افزار Proteus طراحی کنید، سپس در کد برنامه در Arduino IDE در تابع setup پیکربندی ورودی/خروجی پایه‌ها را انجام دهید و در تابع loop منطق کنترلی برنامه را پیاده‌سازی کنید.

پس از آن برنامه را برای ATmega2560 کامپایل کنید (با انتخاب برد از راه Tools < Board < Arduino Mega) و سپس انتخاب گزینه Verify.

پس از آن فایل هگز به دست آمده را بر روی میکروکنترلر در Proteus بارگذاری کنید. و شبیه‌سازی را اجرا کنید.



شكل 1-1



شكل 2-1

آزمایش 2: 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 2560 و آشنایی با ارتباطات Serial

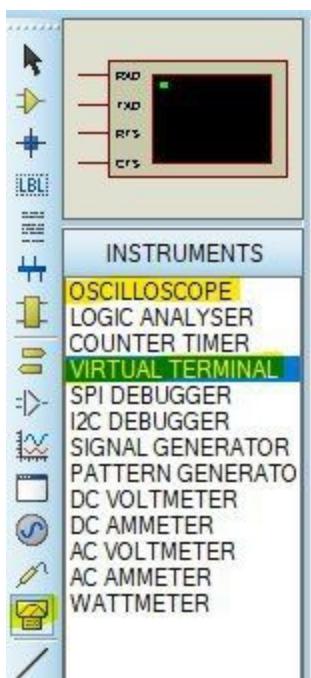
قطعات مورد نیاز:

- برد Arduino Mega2560
- صفحه کلید 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\(\)](#)



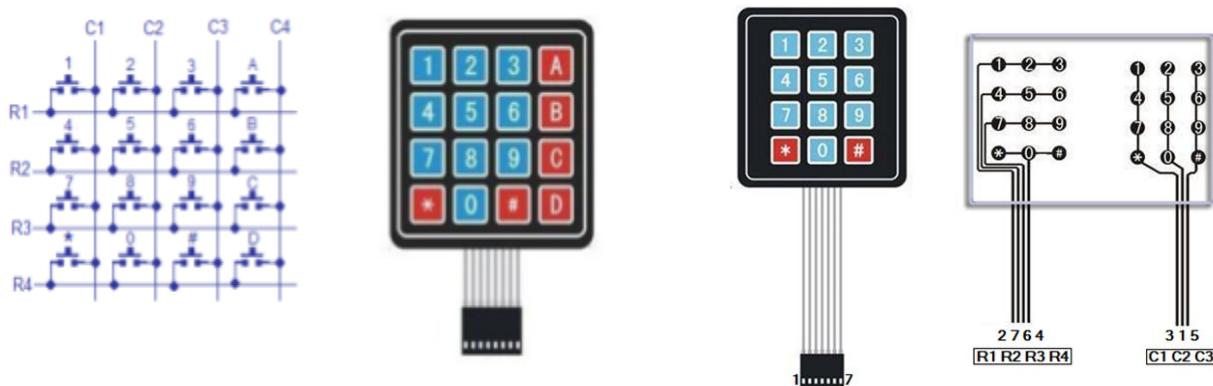
- نحوه و کاربردهای ارتباطات سریال در آردوینو
- تعریف مختصر و نحوه کار با توابع ارتباطات سریال مانند:

- [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 بردها را اشغال کنیم.

در یک صفحه کلید (کیپ) استاندارد، کلیدهای اصلی به صورت ۴×۳ 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 ۴ ردیفی به یکدیگر متصل می‌شوند. برد آردوینو می‌تواند از طریق پین‌های دیجیتال به این سطراها 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 سطراها و ستون‌ها وجود ندارد. شکل زیر ساختمان داخلی یک صفحه کلید 3×4 را نشان می‌دهد.



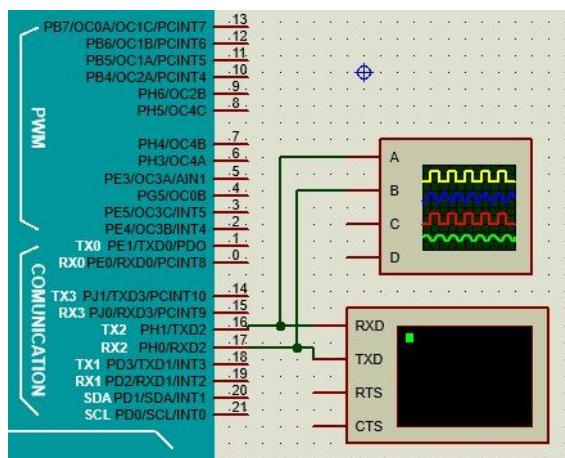
نمایی از ساختار درونی و بیرونی یک کیپ

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>



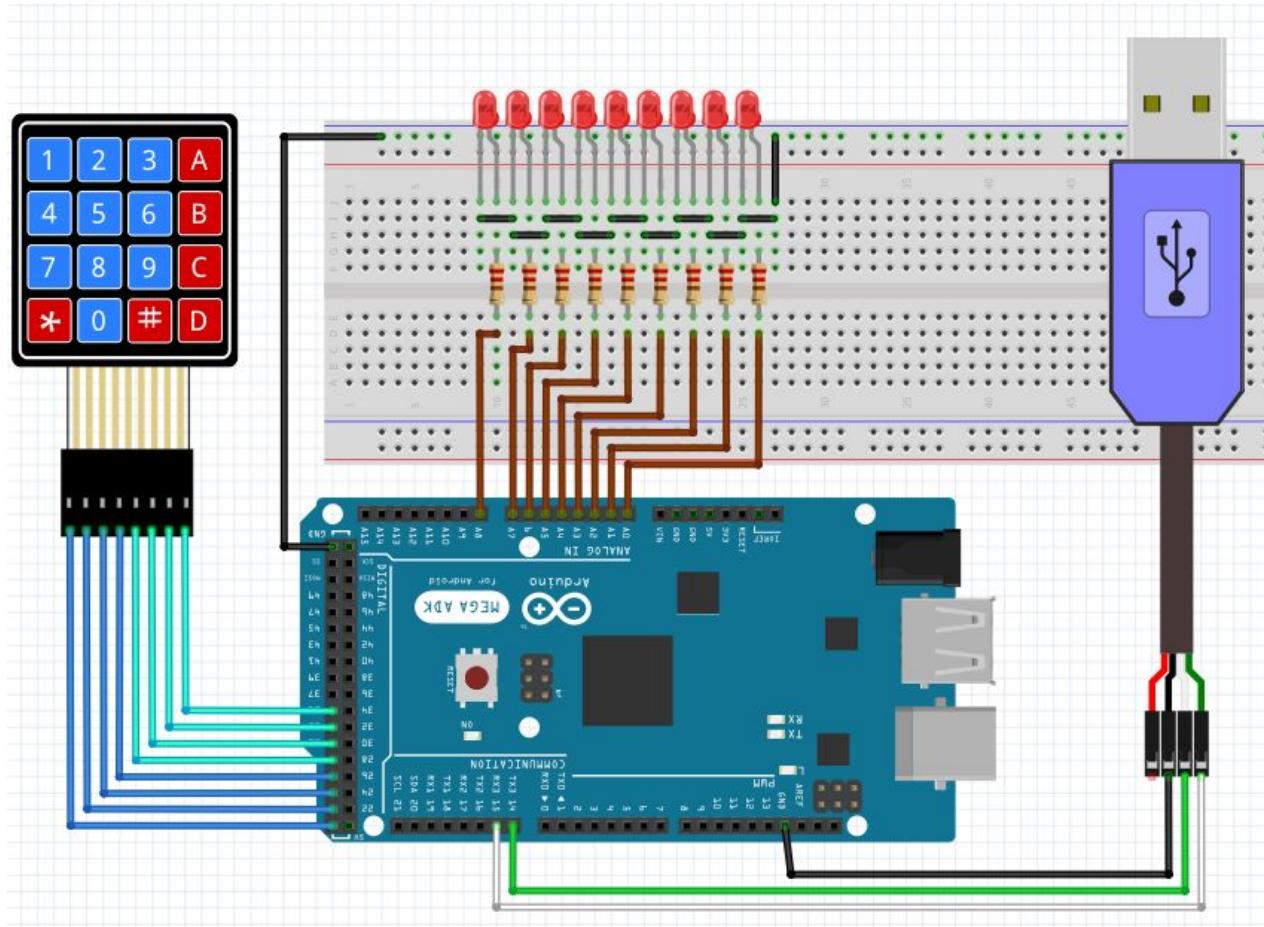
همچنین در این آزمایش قابلیت ارتباطات سریال آردوینو را بررسی می‌کنیم. از نوار ابزار پروتئوس (سمت چپ) و از تب INSTRUMENTS یک اسیلوسکوپ و یک ترمینال مجازی را انتخاب می‌کنیم. ترمینال را به یک TX و RX از پین‌های COMMUNICATION برد وصل می‌کنیم و اسیلوسکوپ را برای مشاهده سیگنال‌های ارتباطی، به همان دو سر ترمینال مجازی متصل می‌کنیم. در آردوینو واحدهای USART و USART با نام Serial به کار برده می‌شوند و چنانچه از واحد USART یا USART n استفاده می‌کنید (RXn، TXn)، اگر ۰ باشد Serial در غیر این صورت Serials نام دارد.

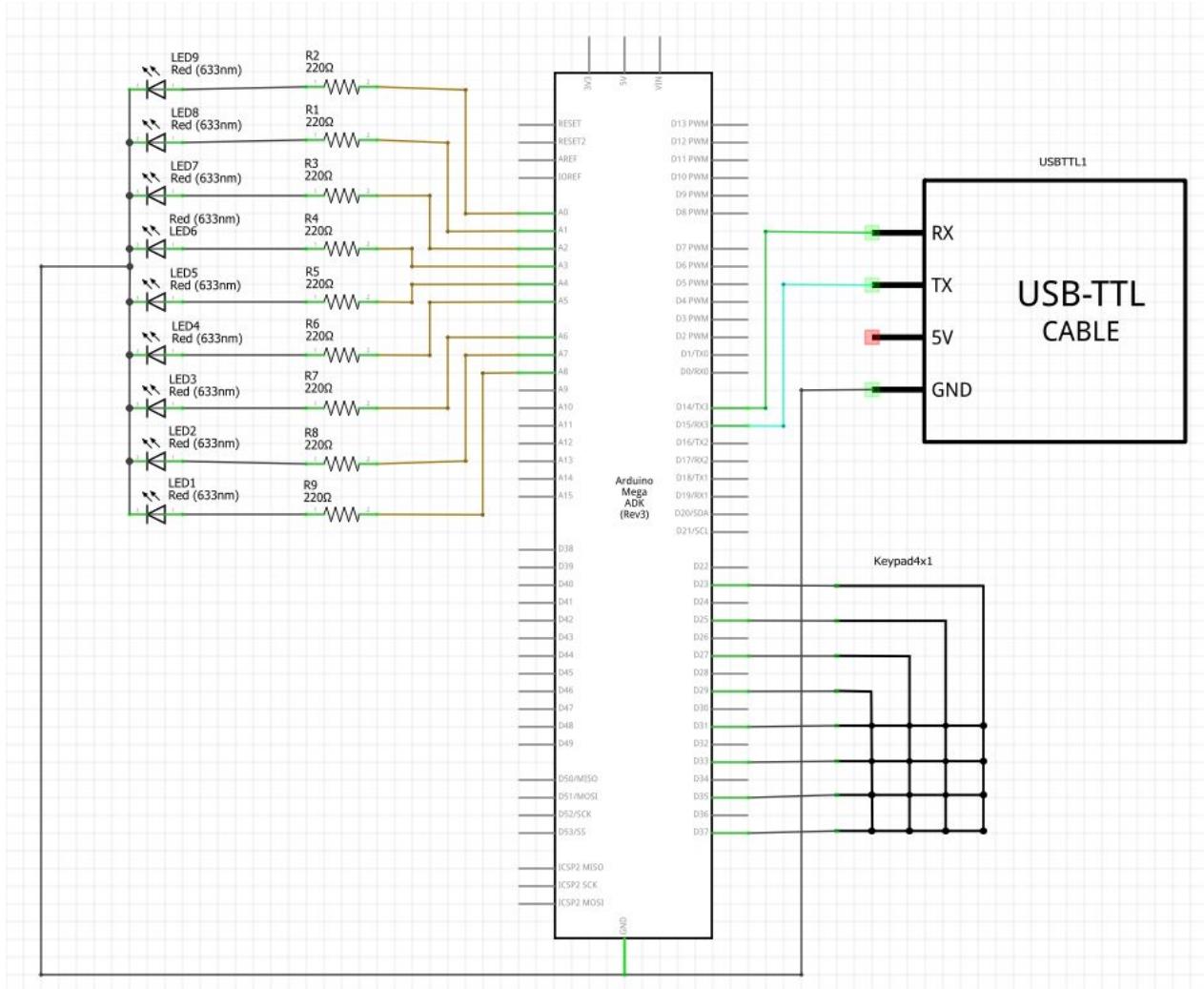
شرح آزمایش:

1. نخست نه عدد LED و یک صفحه کلید را از میان component‌های پروتئوس اضافه کنید و به برد Arduino Mega2560 وصل کنید. برنامه‌ای بنویسید که بنا بر عدد انتخاب شده در صفحه کلید، LED متناظر روشن شود.

2. سپس ترمینال مجازی را (که در شکل زیر به صورت یک تبدیل USB-TTL نشان داده شده) 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؟

3. برنامه‌ای بنویسید که ترمینال مجازی، یک عدد بین ۱ تا ۹ را ورودی بگیرد و LED مربوطه روشن شود.





آزمایش ۳: مانیتور خروجی

هدف آزمایش:

اتصال صفحه نمایش کاراکتری به میکروکنترلر برای نمایش اطلاعات

قطعات مورد نیاز:

- برد Arduino Mega2560
- صفحه نمایش کاراکتری 16x2
- مقاومت 220 Ω یا پتانسیومتر 10k Ω
- ترمینال مجازی برای سریال مانیتور
- کیبورد 4x4

آنچه باید در پیشگزارش نوشته شود:

- کدهای مورد نیاز برای برنامه‌ریزی برد
- مشخصات فنی مژول نمایشگر ال‌سی‌دی کاراکتری 2x16 و دلیل استفاده از پتانسیومتر در مدار
- 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 نمایشگرهای کاراکتری LCD نمایش دهیم. به صورت کلی LCD‌های موجود به دسته LCD گرافیکی و LCD کاراکتری تقسیم می‌شوند.



LCD کاراکتری 2x16 یکی از پایه‌ای‌ترین 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 = زمین | VSS .1 |
| VCC = تغذیه ۵ ولت | .2 |
| VO (Display Contrast Pin) = تنظیم شدت نور صفحه | .3 |
| RS (Register Select) = انتخاب رजیستر | .4 |
| RW (Read/Write) = پایه Read و Write | .5 |
| E = پایه Enable | .6 |

D0 – D7 = پایه‌های دیتا (داده 8 بیتی) .7

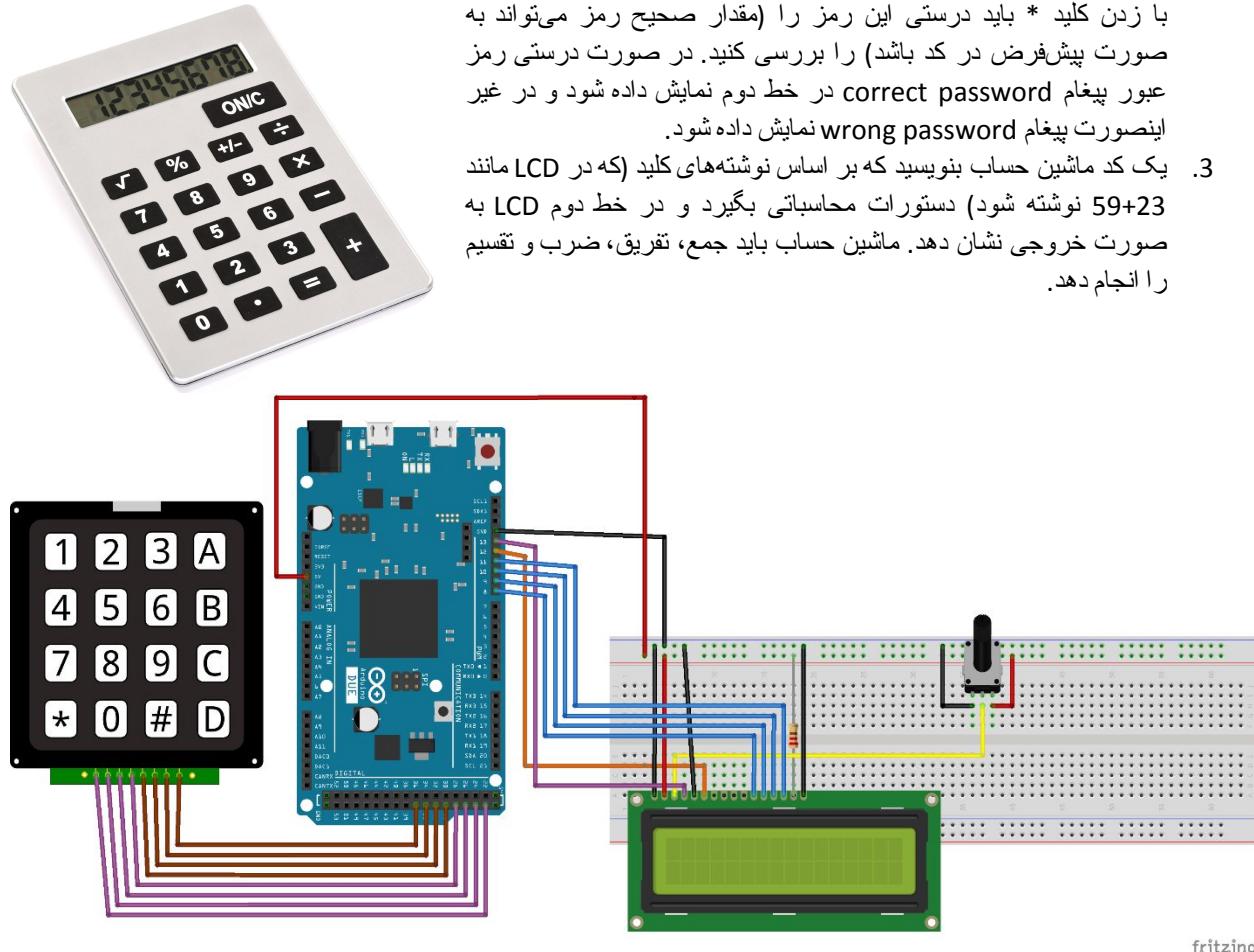
Anode = A .8

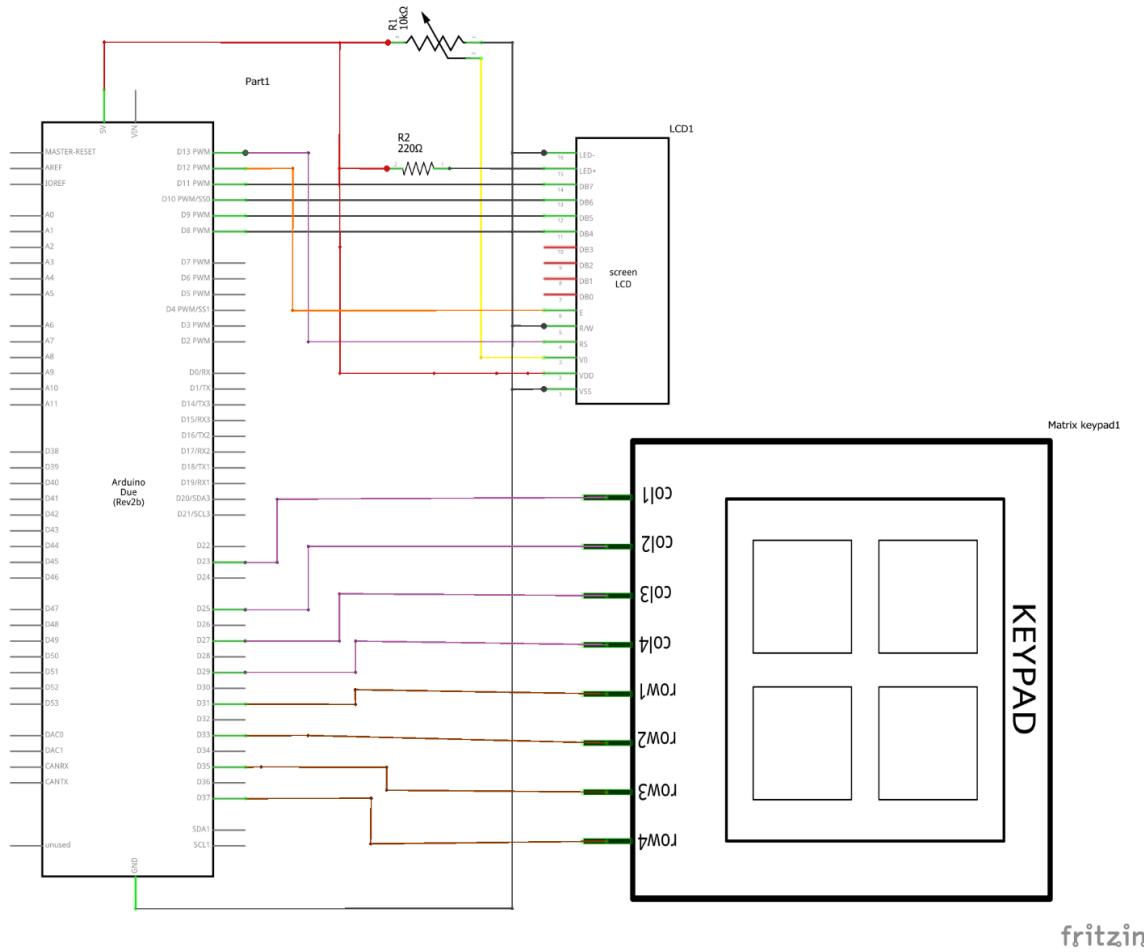
Cathode = K .9

برای برنامه‌ریزی LCD موجود می‌توانید از کتابخانه LiquidCrystal استفاده کنید که به صورت پیش‌فرض در نرم 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 و لذا آن VO، میزان نمایان بودن اعداد و شدت نور صفحه را تنظیم می‌کند.

شرح آزمایش:

1. همانطور که در عکس زیر می‌بینید، ابتدا یک صفحه نمایش را از بین component‌های پروتئوس اضافه کنید و به برد Arduino Mega 2560 وصل کنید. با استفاده از توابع کتابخانه‌ای برنامه‌ای بنویسید که اسم خودتان در ابتدای سطر اول 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 و به طور پیوسته این کار را تکرار نماید.
2. یک صفحه کلید از پیش‌فرض های پروتئوس اضافه کنید و به برد نصب کنید. با استفاده از آن، رمزی 4 رقمی را دریافت کنید و روی LCD نمایش دهید، و با زدن کلید * باید درستی این رمز را (مقدار صحیح رمز می‌تواند به صورت پیش‌فرض در کد باشد) را 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 نوشه‌های 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.





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 پین‌های ورودی آنالوگ و تحلیل و استفاده از آن‌ها

قطعات مورد نیاز

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

آنچه باید در پیش‌گزارش نوشته شود:

- مفهوم PWM و استفاده‌های آن
- کاربردهای Servo Motor
- کدهای مورد نیاز برای برنامه‌ریزی برد
- توضیح در مورد ورودی آنالوگ و تحلیل آن در آردوینو و تابع مورد استفاده این آزمایش

● `analogRead()`

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

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

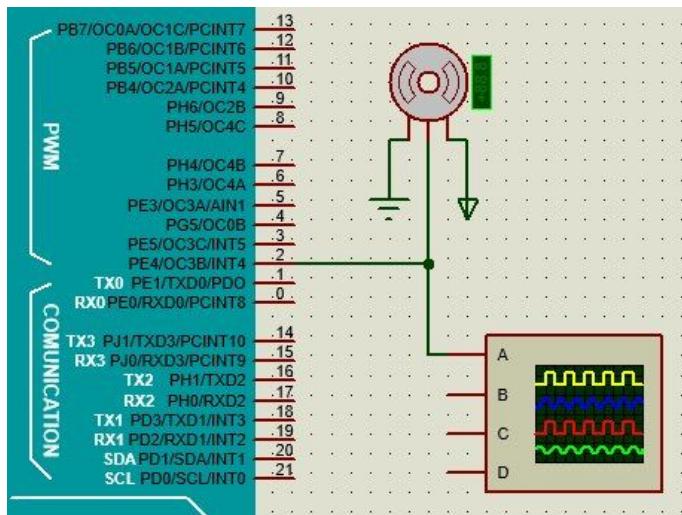
مقدمه:

سرو 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.

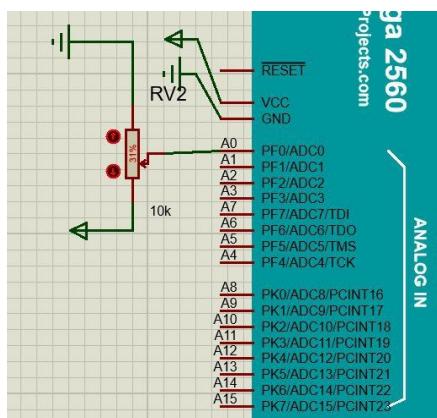
برای مدیریت سرو موتور، نیاز به موج مریبی یا PWM است. به عبارت دیگر، سرو موتور یک موتور کوچک دارای یک محور پا شافت خروجی است. این محور خروجی قادر است در یک موقعیت و زاویه خاص با سیگنال دریافتی قرار گیرد.

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

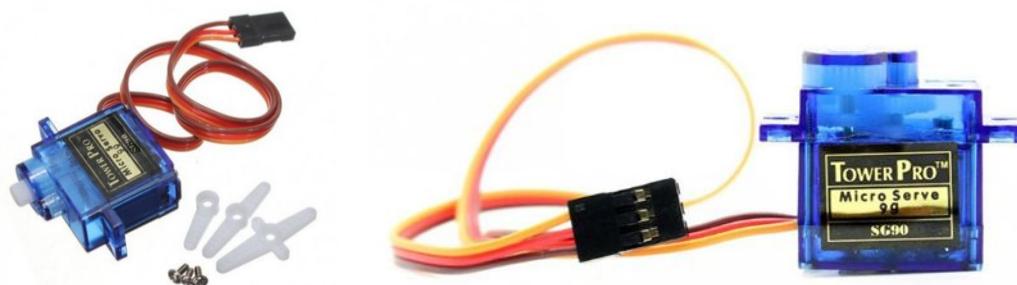
لازم به ذکر است که در محیط آزمایشی پروتوس، سروو موتور با کمی خطا عمل می‌کند و با تنظیم و کالیبره کردن pwm می‌توان از مقدار آن کاست.



سپس برای اتصال پتانسیومتر، پایه‌های کناری را به تغذیه 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.



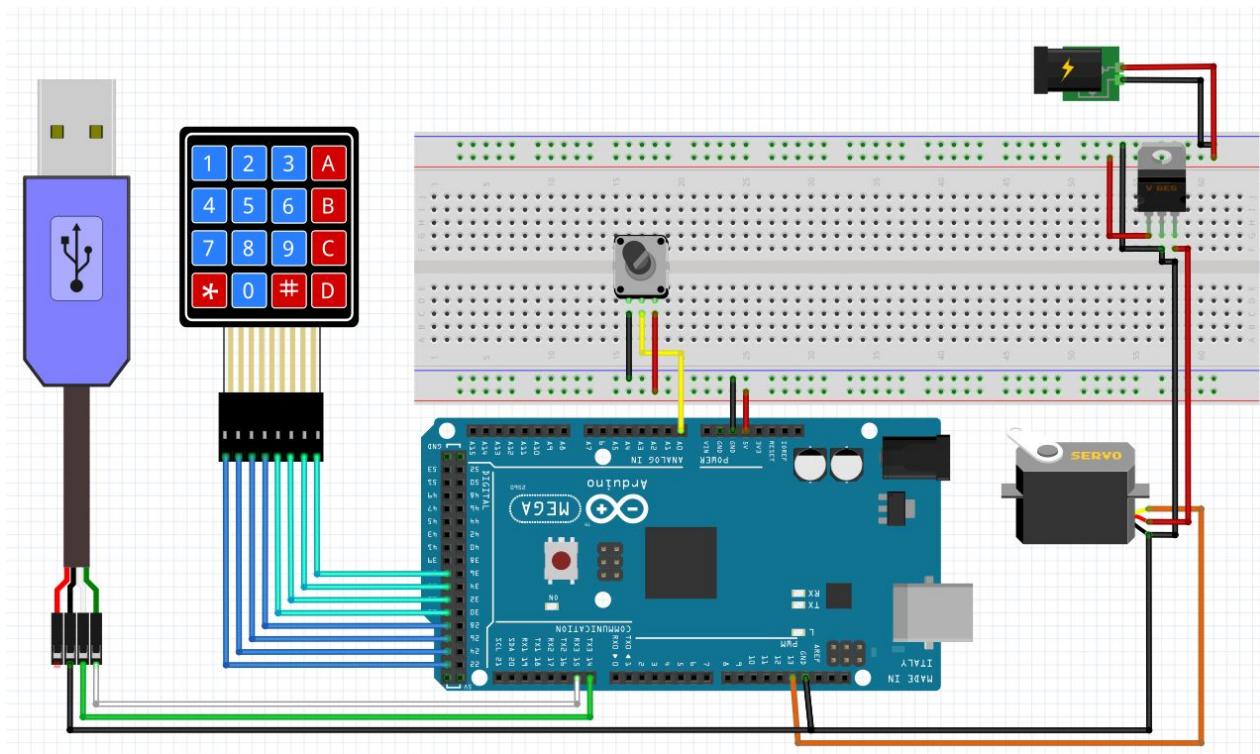
پیشنهاد: تابع map آردوینو در کار با سروو موتور و خواندن داده آنالوگ پتانسیومتر خیلی به شما کمک می‌کند.

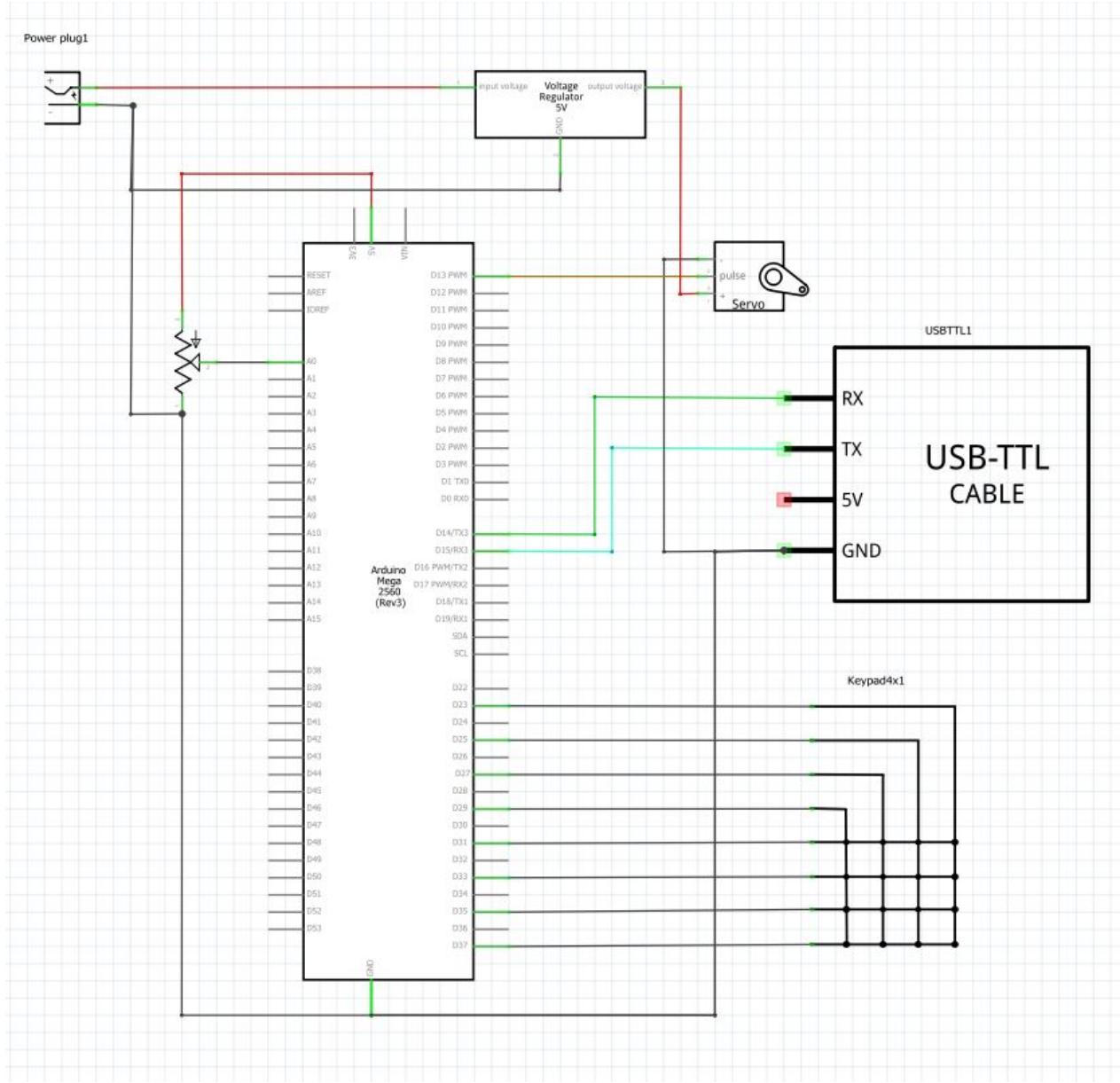


سروو موتور SG90

شرح آزمایش:

1. برنامه‌ای بنویسید که به صورت خودکار سروو از زاویه ۰ تا ۱۸۰ درجه تغییر کند و سپس از زاویه ۱۸۰ به ۰ بازگردد.
2. برنامه‌ای بنویسید که کاربر با کیبورد یک عدد بین ۰ تا ۱۸۰ انتخاب می‌کند که سروو موتور باید آن را بین -۹۰ درجه و +۹۰ درجه نشان دهد.
3. برنامه‌ای بنویسید که با استفاده از سریال مانیتور، مقدار زاویه مورد نظر را وارد کنیم و سروو موتور مطابق همان مقدار وارد شده 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 Duty Cycle) و دوره کاری (Duty Cycle) موچ را به ازای زاویه‌های گردش مختلف سرو موتور به دست آورید؟





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 بالابی هستند و مستقیم با پایه های برد قابل فعال سازی نیستند.

قطعات مورد نیاز:

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

آنچه باید در پیشگزارش نوشته شود:

- رله چیست، انواع آن و از آن به منظوری استفاده می شود؟
- آشنایی با پایه های رله، نحوه کار کرد آن و نحوه تشخیص پایه های رله
- نحوه تشخیص پایه های رله

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 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 می توانند جریانی قوی تر از جریان ورودی را هدایت کند، به معنی وسیع تر می توان آن ها را نوعی تقویت کننده نیز دانست. رله ها چندین ساختار مختلف دارند. اما ساده ترین و پر کاربرد ترین نوع آن ها، رله های SPDT هستند. نمونه ای از این رله ها در شکل 6-1 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.



○ represents the terminals of the relay

دو نمونه از رله های 5 ولت SPDT

نمایی از ساختار درونی رله

1. پایه Common (COM): پایه مشترک میان دو پایه NO و NC که بسته به شرایط به یکی از این دو پایه متصل است. می توانیم این پایه را به یک سر موتور متصل کنیم.
2. پایه های COIL: این پایه ها مربوط به COIL درون 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 کار کرد این مدار توضیح دهید.

شرح آزمایش:

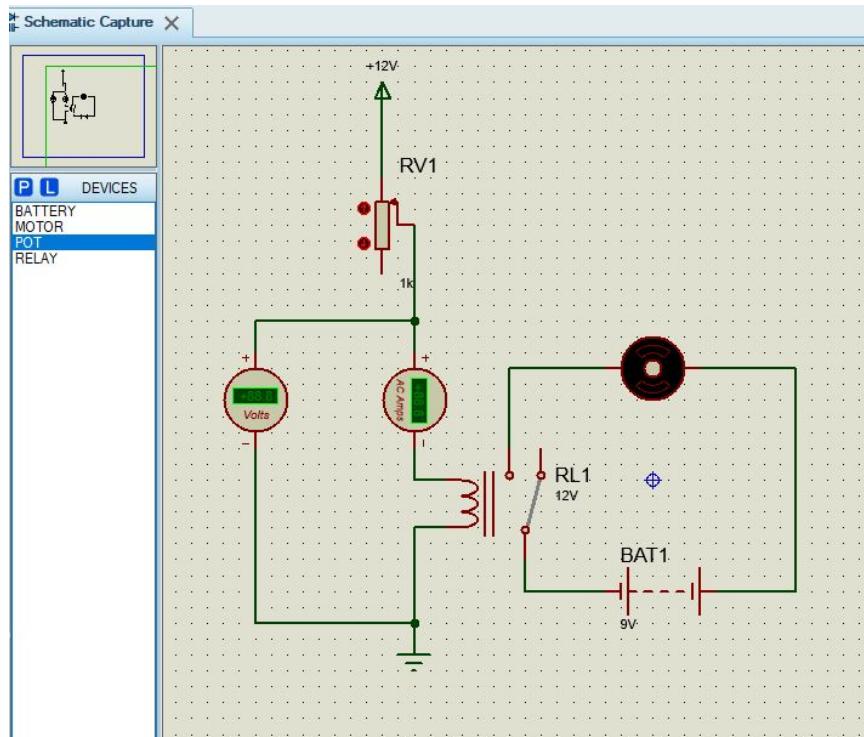
در این آزمایش همانند مدار های شکل 1 و 2 که در زیر آورده شده است، می خواهیم با رله یک موتور را راه اندازی کنیم، هنگامی که موتور خاموش است باید LED روشن باشد و وارون آن. گام یک:

نخست مدار زیر را با پتانسیومتر (POT) ، رله (Relay) ، موتور (Motor) ، باتری و آمپر متر و ولت متر رسم کنید. و آن را اجرا کنید. ابتدا POT را بر روی بیشترین مقاومت قرار دهید. در این حالت ولت متر کمترین و آمپر متر کمترین مقدار را نشان خواهد داد. و 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 های 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 مراتب بیشتر از ولتاژ یک منطقی ATMega2560 است. (5v)

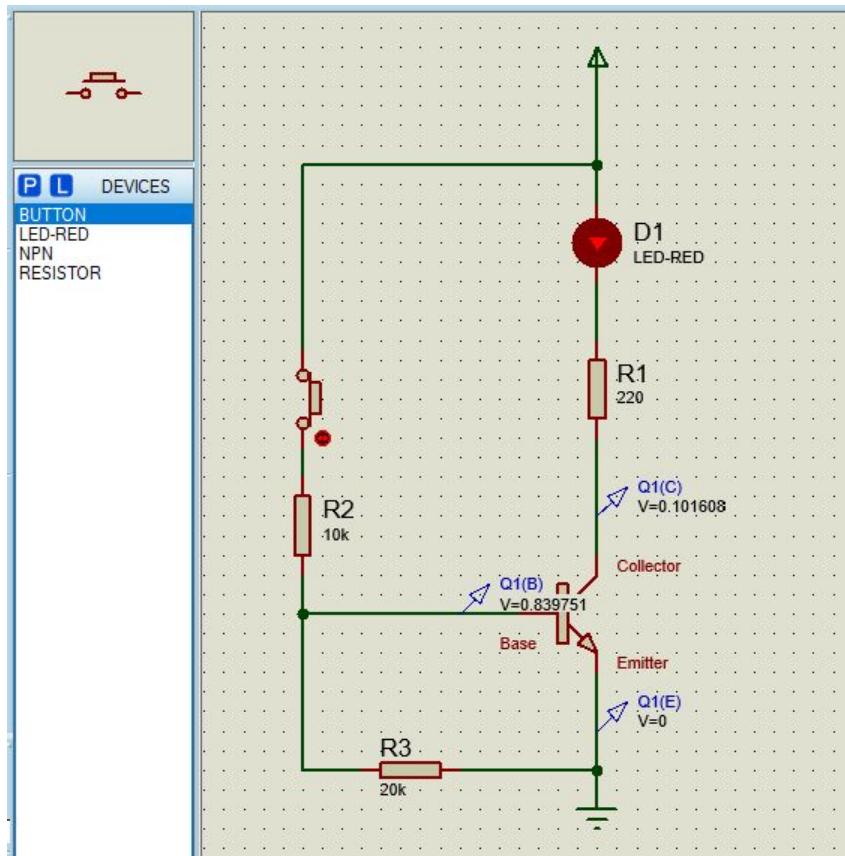
همچنین جریانی که برای فعال کردن رله نیاز هست نیز معمولاً از بیشینه جریانی که هر پین ورودی/خروجی ATmega2560 فراهم می‌کند بیشتر است. (40mA)



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.

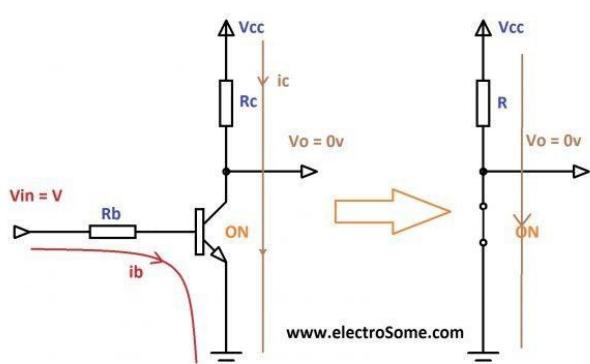
گام دو:

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

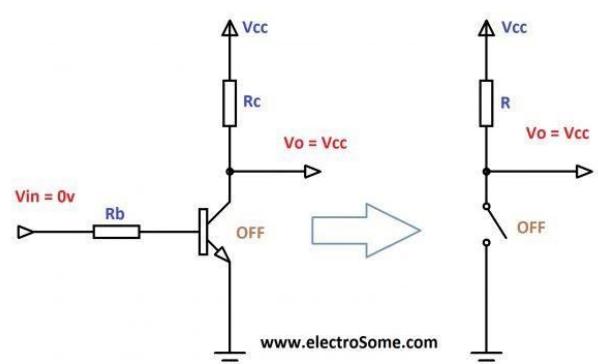


همان گونه که دیده می شود به طور خلاصه می توان گفت به از ای افزایش جریان از پایه Base (I_B)، بیشینه جریانی که از Emitter به Collector می تواند بگذرد به نسبت بیشتری افزایش می باید. به گونه ای که در حالت اشباع ترانزیستور می توان از مقاومت ترانزیستور از Collector به Emitter در مدار چشم پوشی کرد.

حالت کلید بسته:

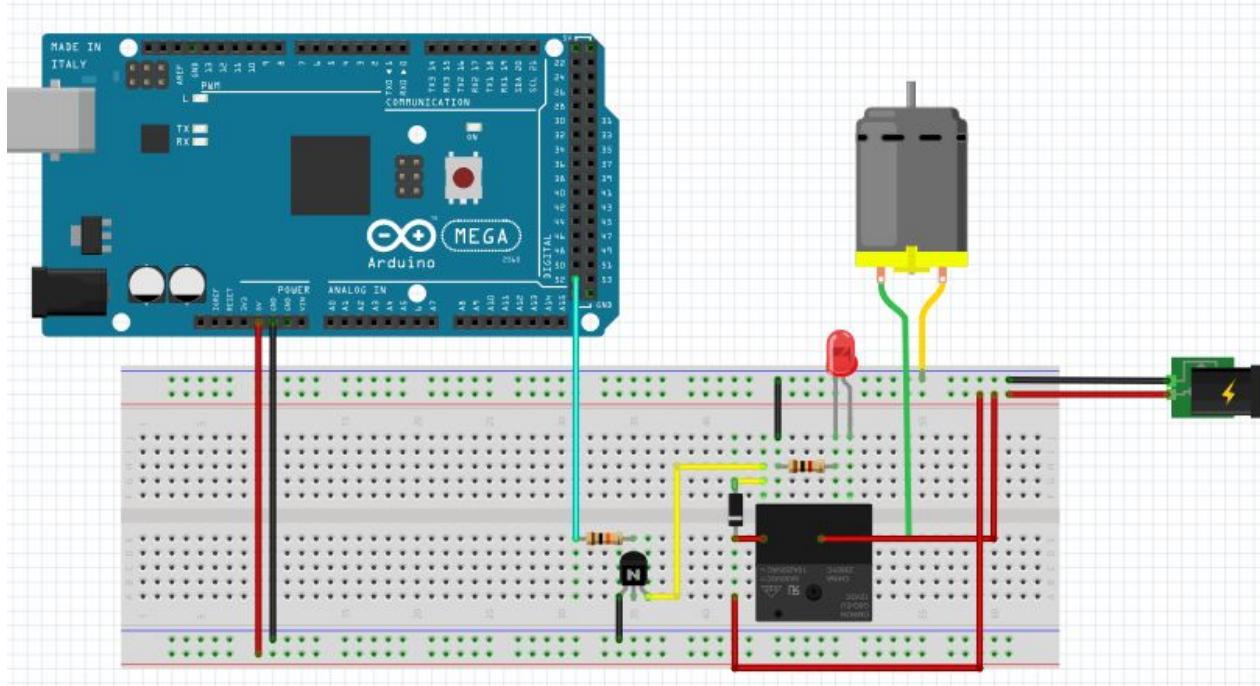


حالت کلید باز:

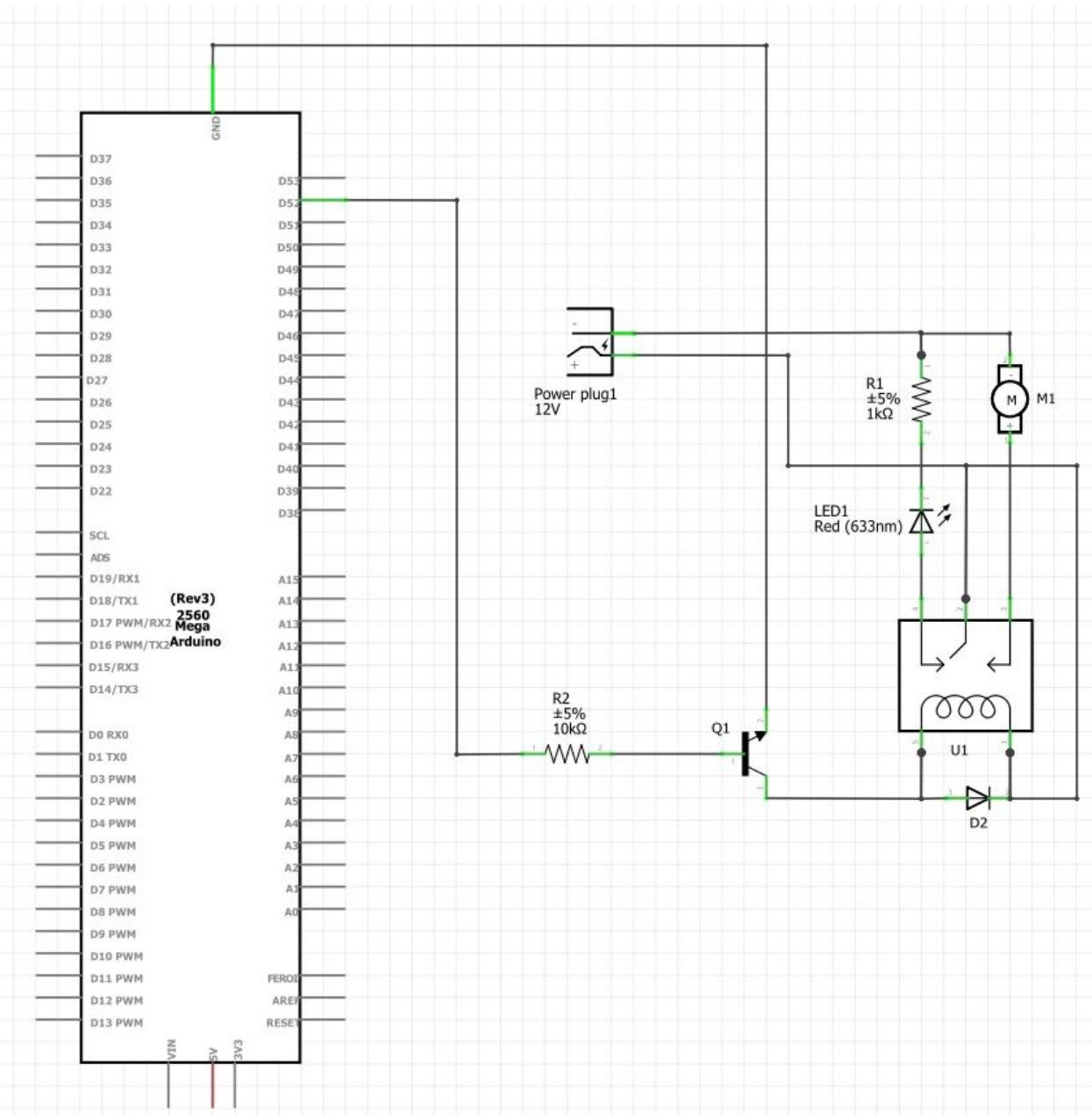


گام سوم:

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

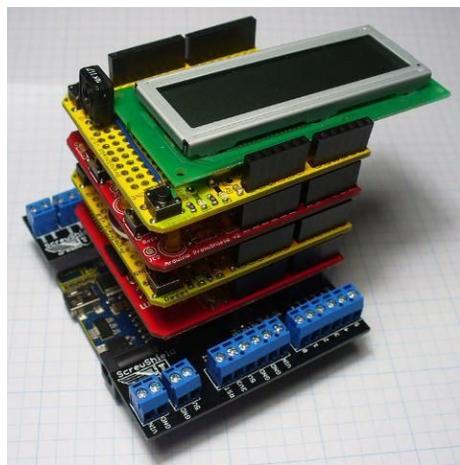
هدف آزمایش:

مروری بر مطالب که در جلسه‌های گذشته گفته شد و پیاده‌سازی یک کاربرد ساده از این مطالب با بهره‌گیری از خلاقیت و ایده پردازی خودتان. به عنوان مثال، تعدادی ایده پیشنهادی در این 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، می‌توانند یکسری مسائل بازگشتی (مانند بررسی زوج‌فرد بودن بازگشتی) را حل کنند.

قطعات مورد نیاز:

- دو عدد برد Arduino Mega2560
- کیبورد
- LCD کاراکتری

کامپیوتر ماشین کنترلی

هدف پروژه

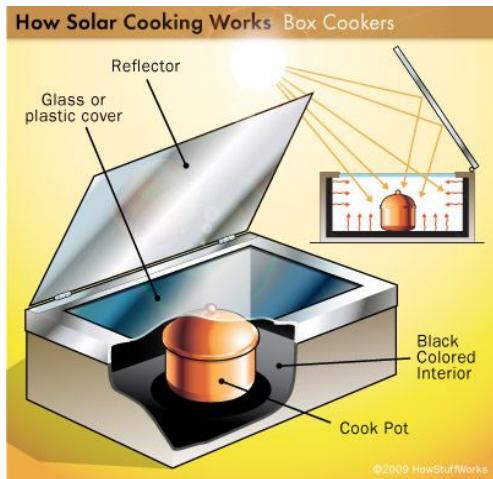
این ماشین به سادگی باید از ارتباطات سریال دستور بگیرد (که فرضاً به کنترلر رادیویی متصل شده) و چراغهای ماشین (LED) را روشن کند، به چرخهای جلو زاویه بدهد (سرورو موتور) و چرخهای عقب را به حرکت بیندازد (رله و آرمیچر) و بوق بزند (بازر)



قطعات مورد نیاز:

- برد Arduino Mega2560
- دیود نور ای (LED) به مقدار لازم
- سرورو موتور
- آرمیچر
- رله
- بازر

اجاق خورشیدی



هدف پروژه

اجاق خورشیدی دستگاه نور خورشید را از شیشه می گذراند و توسط دیوارهای سیاه درونی، 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 اندازه گرفته شده، با بازار (buzzer) کاربر را از پخت غذا آگاه کند.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- سروو موتور
- حرارت سنج
- کیبورد
- بازار

یعقوب بر قی رایگان!

هدف پروژه

سیر کردن داشجوها. یک گذرو از هفت رقی (کیبورد) در راباز می کند (یک سروو موتور) همه پیچش ها را (تعداد زیادی سروو موتور) به عقب می چرخاند تا غذا کار گذاشته شود. همه ی چراغها (LED) در این مرحله خاموش است. دکمه دوباره در رامی بند و چراغها روشن می شوند. سپس داشجوها با زدن شمارهای دور قمی غذا ها، 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 کار اکثری



پنکه



هدف پروژه

خنک کردن داشنجوها. یک دکمه (button) به کمک رله پنکه را به راه می‌اندازد (آرمیچر و رله) و هنگام فشرده شدن دکمه، بازر بوق می‌زنند (buzzer) و یک دکمه دیگر بین سه سرعت مختلف آرمیچر سوییج می‌کند که هر سه توسط یک نور (LED) به کاربر نشان داده می‌شوند. یک دکمه دیگر هم چپ و راست شدن پنکه را فعال و غیرفعال می‌کند (به کمک سروو موتور)

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 به کار ببرید.

قطعات مورد نیاز:

- بورد Arduino Mega2560
- سه عدد LED
- چهار دکمه (push button)
- آرمیچر
- رله
- بازر
- صفحه کلید
- LCD کاراکتری

گاوصندوق

هدف پروژه



کاربر گزروازه را وارد می‌کند (صفحه کلید) و سپس در گاوصندوق باز (سرورو 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 رابیند، 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 گزروازه، وضعیت در و زمان سنج بسته شدن در نمایش داده شود. همچنین باید بازه زمانی گفته شده و گزروازه تازه را بتوان از صفحه کلید دریافت کرد و باز تنظیم نمود.

قطعات مورد نیاز:

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

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 خواندن از آن

قطعات مورد نیاز:

- Arduino Mega
- حافظه AT24C02
- LED
- صفحه کلید (Key Pad)
- 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 EEPROM‌ای با نام AT24C02 که با این 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 غیر فرار 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). در این کاربرد گفته شده برای پیاده‌سازی نیازمندی خود از EEPROM بسیار بسیار است.

پرسش: در چه کاربردهایی EEPROM به کار برد می‌شود؟ چرا در اینجا حافظه Flash یا RAM را به کار نمی‌بریم؟ تفاوت حافظه EEPROM با RAM در چه است؟

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 دستگاهی می‌توان EEPROM‌ها 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 سربار استفاده از پروتکل ارتباط سریال برای نوشتن و خواندن را از میان برداشت. از این رو حافظه‌های درونی تندتر هستند، ولی اندازشان محدود است.

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

گونه‌ای دیگر از EEPROM‌ها بیرونی هستند که بر روی یک IC جداگانه 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 نیز 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).

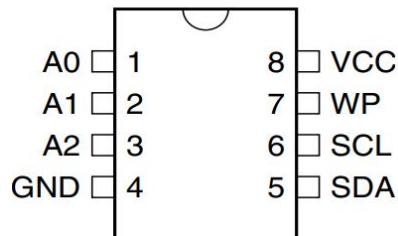
نمای بیرونی و پایه‌های AT24C02

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.

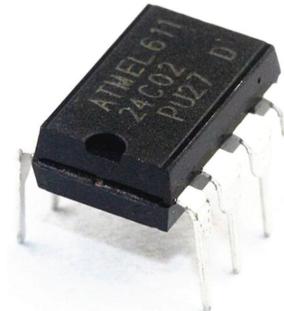
Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply

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؟

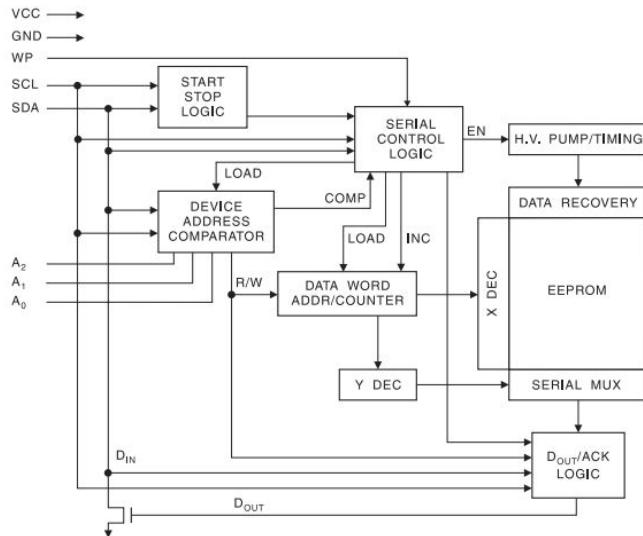
AT24C02 از پروتکل TWI بهره می‌برد؛ با این تفاوت که به جای 7 بیت، تنها 3 بیت برای آدرس دهی اختصاص یافته (A0-A3) و 4 بیت دیگر (پر ارزش‌تر) مقداری ثابت و برابر با 0b1010 دارند.

کارکرد دیگری که معمولاً در EEPROM‌های بیرونی وجود دارد، حفاظت نوشتن (Write Protection) یا Write Enable است. و 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.

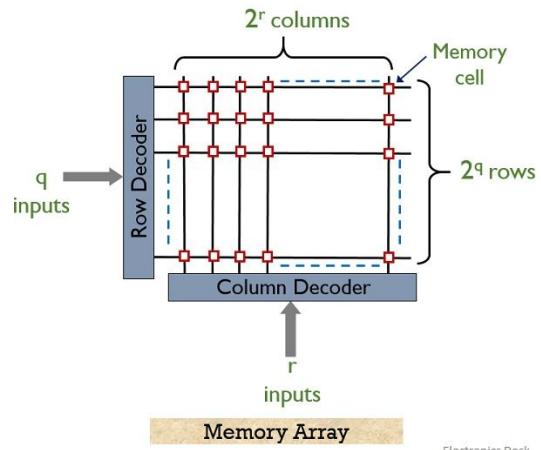
پرسش: نمودار شماتیک برای این‌که دو AT24C02 را به یک باس مشترک وصل کرد و حفاظت نوشتن غیر فعال باشد را رسم کنید.
(آدرس دهی سخت‌افزاری دلخواه - باس را هم به پایه‌های میکروکنترلر متصل کنید)

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 نوشتمن در آغاز باید آدرس خانه‌ی مورد نظر در شمارنده آدرس (Data Word Addr/Counter) قرار بگیرد. همان‌گونه که دیده می‌شود مقدار این شمارنده ورودی دیکتر آدرس (X) است و به این صورت 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.

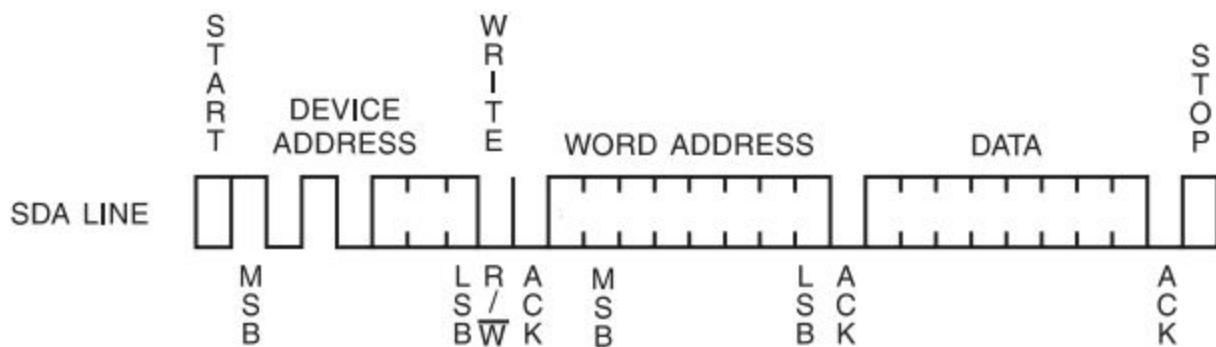
از این رو، برای این‌که بتوان از یک خانه حافظه خواند باید نخست با انجام یک عملیات خواندن نیمه کاره (Dummy Write)، این شمارنده را دوباره مقدار دهی کرد تا خانه مورد نظر در ماتریس حافظه انتخاب گردد.

عملیات نوشتمن 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).

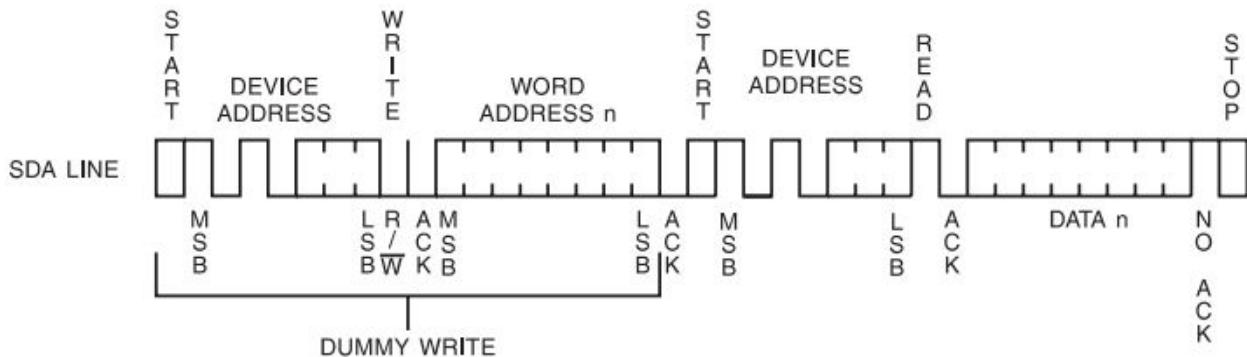
پیام‌های عملیات خواندن و نوشتمن

در زیر نمونه فریم‌های پیام برای عملیات خواندن و نوشتمن باقی نشان داده شده است:

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

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 می‌یابد. به طور کلی فرکانس 100 KHz را می‌توان برای فرکانس کلک (SCL) ارتباط TWI استفاده کرد.

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

کتابخانه Arduino TWI

کتابخانه [Wire](#) که یکی از [کتابخانه‌های استاندارد](#) آردوینو است (از این رو نیازی به نصب آن نیست)، رابط 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 (یا 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 -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 روشن 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 تعذیه نیز در دستگاه بمانند.

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.

پایان فرآیند نیز با روشن شدن همه LED 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.



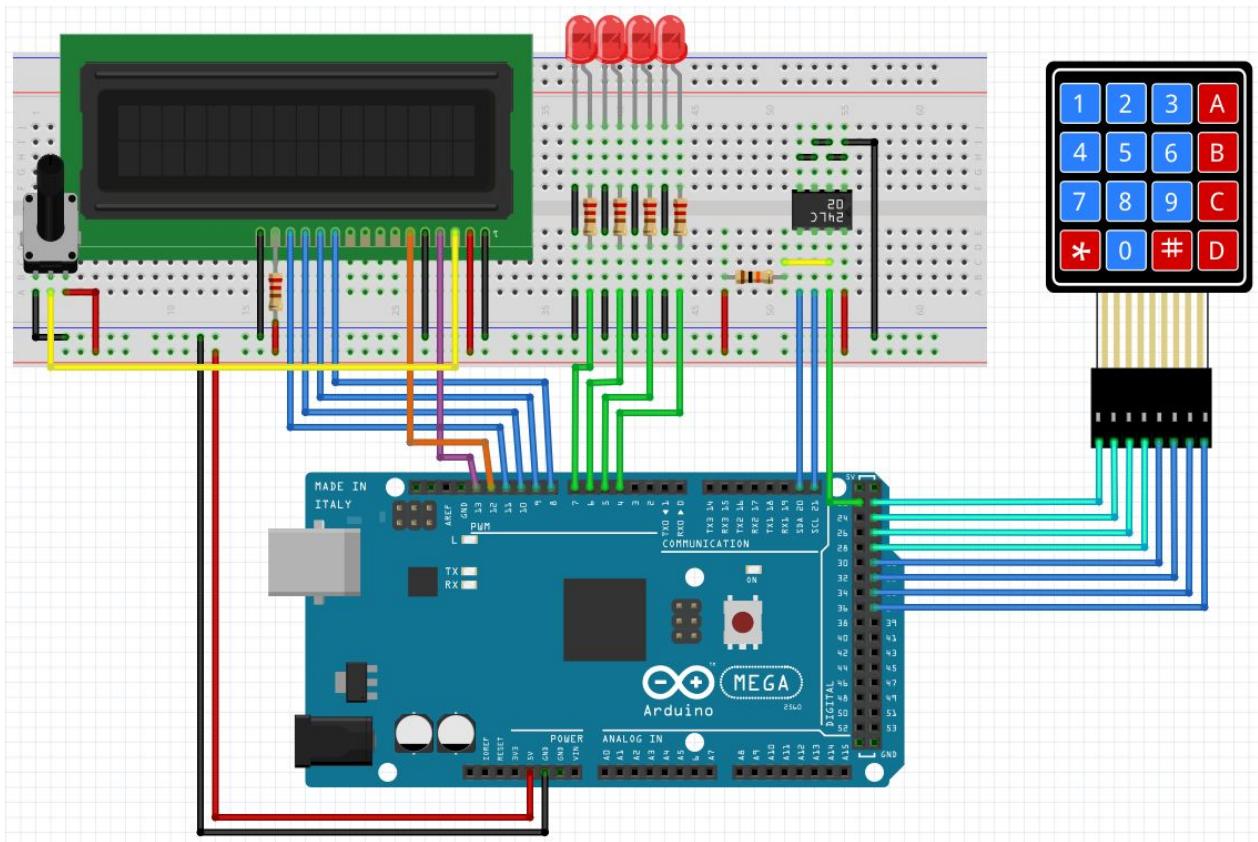
تمرین بیشتر: (تمرین‌های زیر اجباری نیستند، نباید جزوی از آزمایش در نظر گرفته شوند)

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

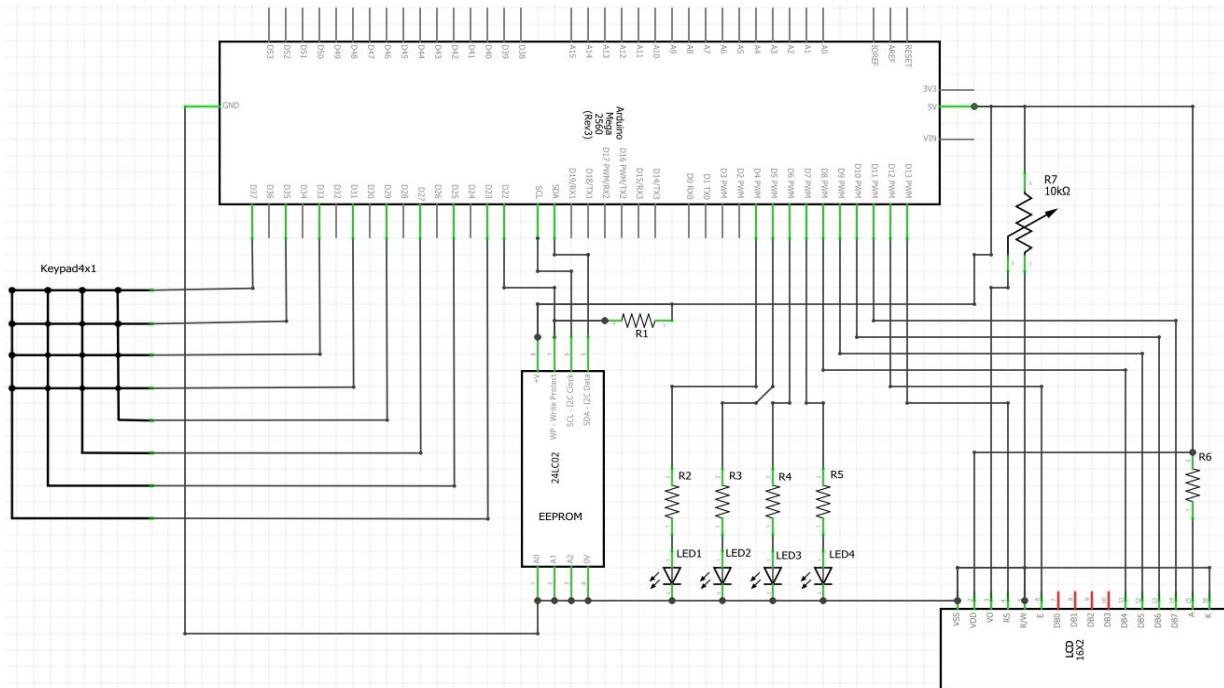
- طراحی را به گونه‌ای تغییر دهید که دو یا چند EEPROM دیگر نیز به باس متصل شده، داده‌ها دسته‌بندی شوند و بر روی EEPROM متناظر با خود ذخیره گردد.

- در صورت موافقت سری‌پرست آزمایشگاه، دو گروه بردهای خود را با یکدیگر به اشتراک گذاشته و دو میکروکنترلر را با پروتکل TWI به هم مرتب‌سازند و انجام یک الگوریتم مانند مرتب‌سازی یک آرایه را میان این دو پخش کنند، سپس خروجی‌ها را با یکدیگر یکپارچه کنند.

- در صورت موافقت سری‌پرست آزمایشگاه دو گروه بردهای خود را با یکدیگر به اشتراک گذاشته و دو میکروکنترلر را با پروتکل TWI با یکدیگر مرتب‌سازند. کنترل واحد TWI را به دو روش سرکشی (Polling) و وقفه محور (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:

آشناپی با پروتکل SPI

تحلیل موج خروجی آردوینوی مرکزی (master)

راه اندازی حسگر نور و دما

قطعات مورد نیاز:

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

آنچه باید در پیشگزارش نوشته شود:

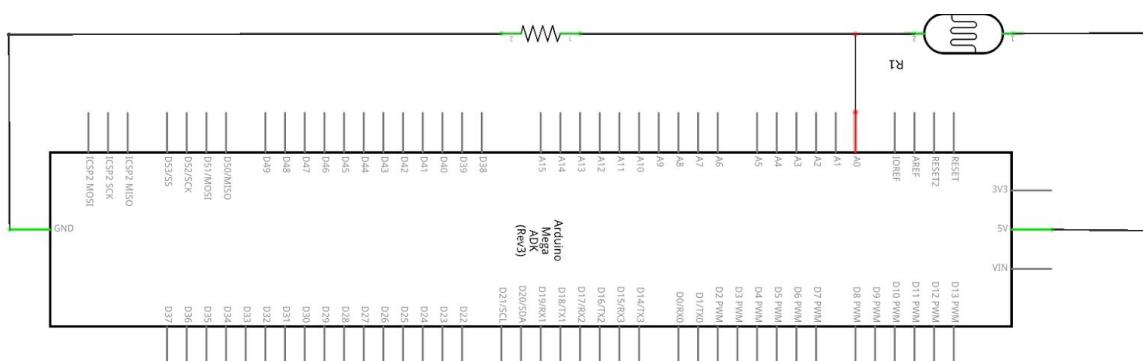
- به پرسش‌های درون مقدمه پاسخ داده شود.

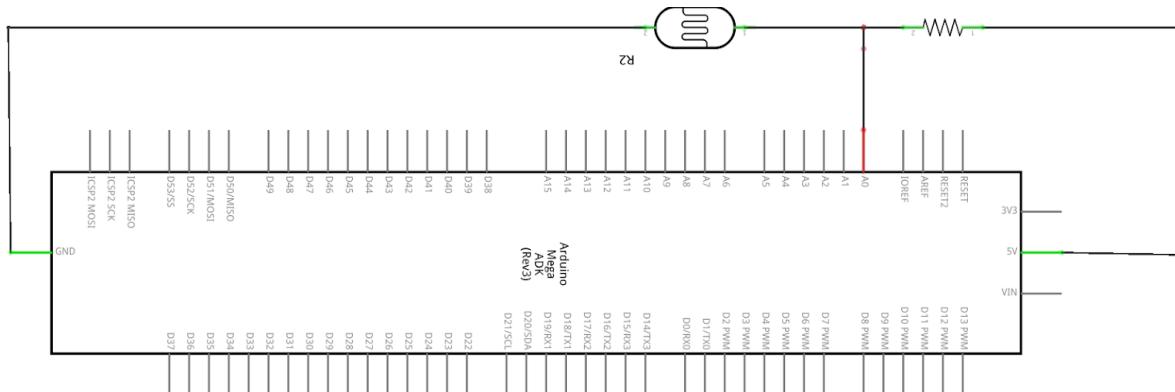
مقدمة

در این آزمایش می‌خواهیم با پروتکل ارتباط SPI بیشتر آشنا شویم و از آن برای ارسال میزان نور اتاق و دمای آن اقدام کنیم. این کار در حالت عادی به نظر منطقی نمی‌رسد، زیرا نمایش اطلاعات در همان آردوینوی مرکزی هم امکان پذیر است. اما این‌کار 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 این قطعه 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 می‌کند.



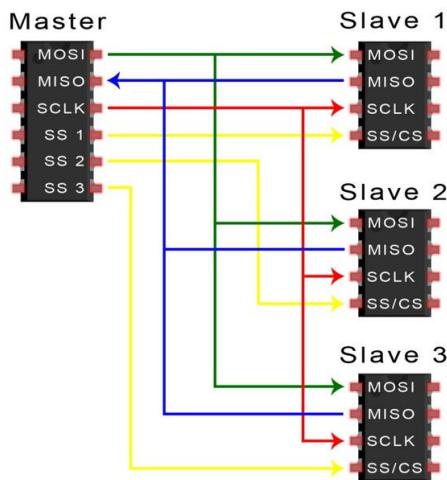
راه اندازی سنسور دما : Im35

این سنسور 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 پرده ای :

همان طور 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)، بوردنی که می‌خواهد با آن ارتباط برقرار کندر را انتخاب کرده و برای آن پیامی ارسال می‌کند و در صورت نیاز، از آن درخواست پاسخ می‌کند.



در این ارتباط می‌باشد سه پایه MISO (که برای ارسال داده از سوی بورد Slave به بورد مرکزی در نظر گرفته شده) و 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 دستگاه‌ها یکی شده باشد. از آنجا که تنها یک دستگاه مرکزی (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 SCLK، MISO، MOSI در آردوینو Mega 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 فایل داخل پوشه را باز نمایید:

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

پرسش : در مورد نحوه انتخاب بورد **SPI** توسط **Slave** تحقیق نموده و نحوه پیاده‌سازی برنامه را برای این‌که بورد 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.

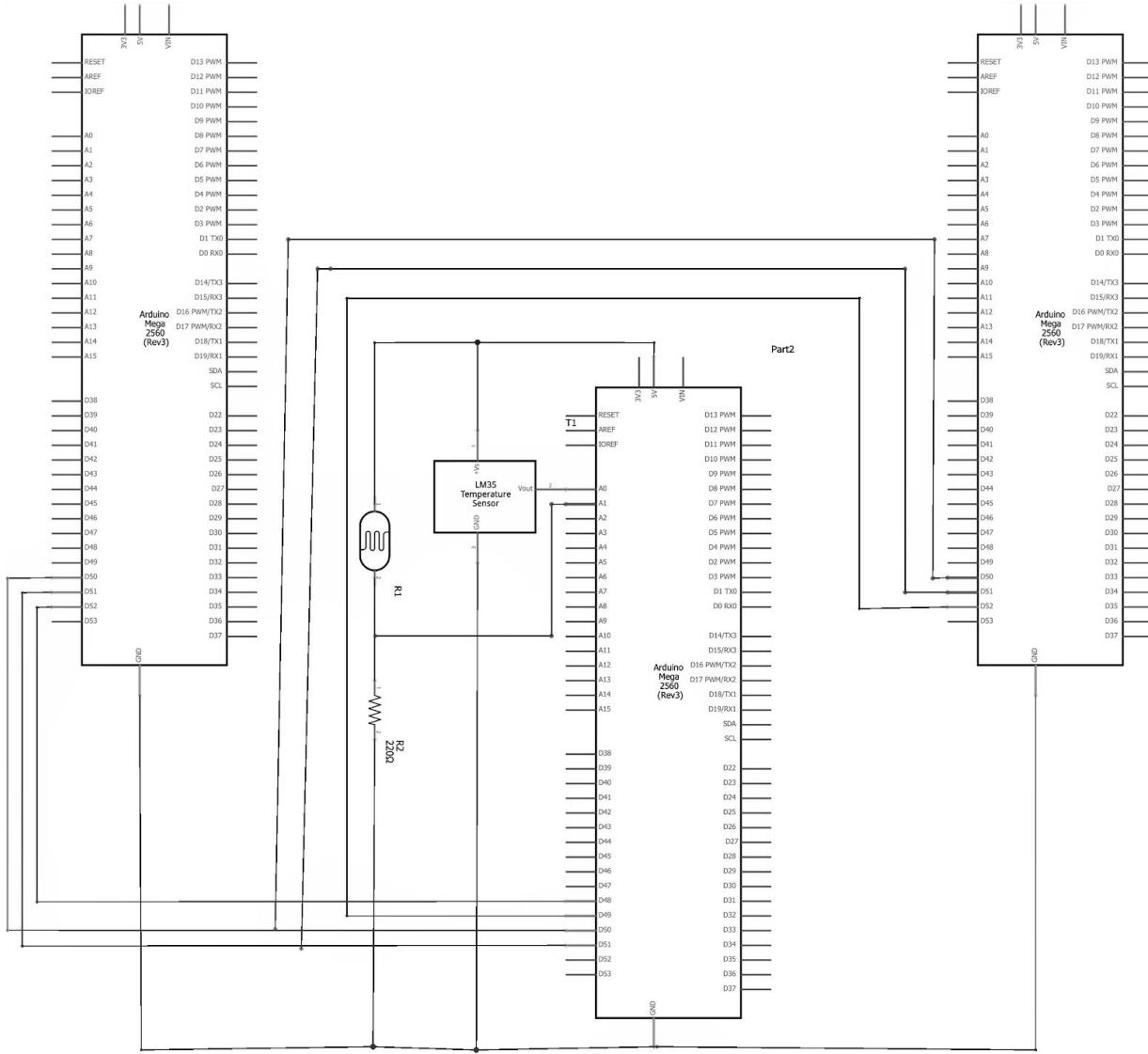
پرسش : دستور مورد نیاز برای این‌که آردوینو در حالت **Slave** قرار گیرد، را نوشته و در مورد کارایی آن تحقیق نمایید.

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 **Hello world** 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 متناسب و هر ثانیه به آردوینو دوم کلمه **Hi** ارسال شود و در ثانیه‌ی بعدی آردوینو اول کلمه **Hello world** را دریافت کند.
4. داده‌های **Hello world** و **Hi** را با اطلاعات مربوط به دو سنسور دما و نور جای‌گذین نمایید. برای خواندن ولتاژ خروجی هر یک از سنسور‌ها کافی است 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 پروتتوس به نام sounder)
- پتانسیومتر
- کلید

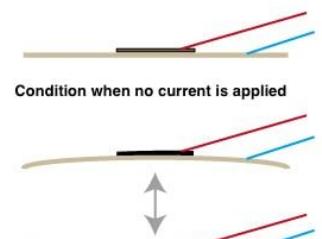
آنچه باید در پیشگزارش نوشته شود:

- به پرسش های 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 صوت تولید کند. امروزه در ساعت‌های دیجیتال کوارتز و دستگاه‌های الکترونیک دیگر استفاده می‌شود و برای اسپیکر سیستم‌های ارزان مانند رادیوهای پرتاپل و لوازم خانگی. این طراحی نسبت به بقیه طراحی‌های بلندگو بسیار ساده‌تر کار می‌کند.

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



ستور در tone

`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 تأثیری ندارد و اگر در حال اجرا بر همان پین باشد، صدا زدنش فرکانس را آپدیت می‌کند. اگر دو اسپیکر piezo را به دو پین بورد 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 یک تایمر داخلی بورد می‌سنجد، پس اگر نت‌های مشخص می‌خواهید، بعد از دستور، به همان اندازه delay یک duration_in_milliseconds قرار دهید.

پرسش: تایمیری که دستور tone استفاده می‌کند با خیلی از پین‌های برد مشترک است. بررسی کنید که به چه روش‌هایی می‌توانید آن تایمیر را به هم بربزیزید که دستور tone خراب شود و صدای مطلوب را اجرا نکند.

تئوری موسیقی:

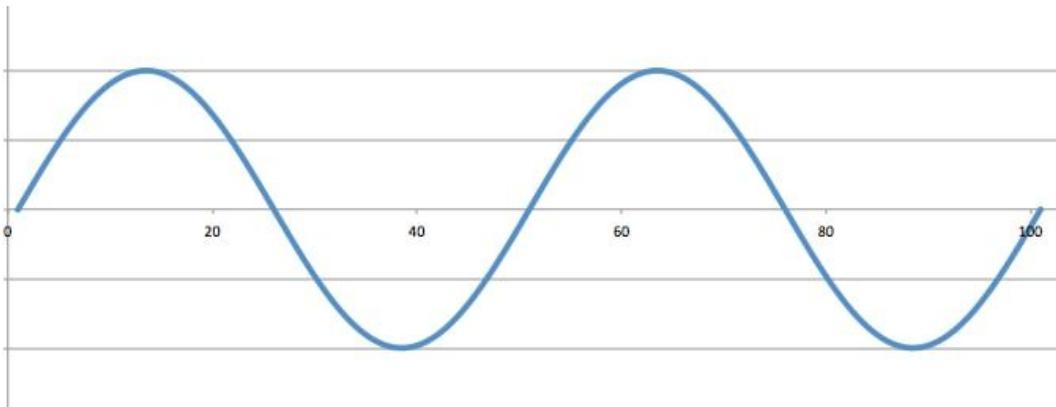
اسپیکر های piezo برای اجرای فرکانس‌های کمتر از ۳۱ هرتز نیستند. این مشکل زیادی ندارد، چون گوش انسان از ۲۰ هرتز شروع به شنیدن می‌کند. پایین ترین اکتاو موسیقی (هر بار که تمامی نت‌های موسیقی را از پایین به بالا می‌شمارند یک اکتاو نامیده می‌شود) با فرکانس ۳۲ هرتز شروع می‌شود که دقیقاً بالای حداقل piezo است. در این زیر، فرکانس‌های نت‌های 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 ستون یک اکتاو را نشان می‌دهد که نت‌ها (و بعضی #‌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 A4 است. به این معنا که ۴۴۰ هرتز / ۴۴۰ نوسان در ثانیه در هوا و اسپیکر piezo ساخته می‌شود.

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

Jingle Bells

TRADITIONAL
arr. A.L.Christopherson

Piano

E E E E E E E G C D

³ *f* ⁵ ₁ ²

E F F F E E E E

₃ ₄ ₃

E D D E D G

₂ ₂ ₃ ₂ ₅

این ملودی را می‌توانیم به عنوان یک سری از نتهای نوخته شده بخوانیم. به ترتیب EEEEGCDE. 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 هشتم دایره است و 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 کار 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 ...

همچنین این جدول علامت های نت های سکوت را به ما نشان می ده، 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.



فایل های هدر:

تاکنون در آردوینو از کتابخانه ها استفاده کرده ایم. برای LCD و Servo، و اخیرا هم TWI. هر چند ساختن یک فایل هدر در آردوینو خوش بسیار ساده است. برای اینکه یک فایل جدید کنار فایل اصلی درست کنید، از دکمه **Ctrl+Shift+N** یا **New Tab** استفاده کنید، و فایل جدید را با پسوند **.h** ذخیره کنید. اکنون میتوانید در فایل اصلی خود آن را **include** کنید.

برای مثال در این آزمایش، دو تا از قطعات حجیم کد را به جای گذاشتند 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 های 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
```

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

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 **i** اول شامل اسم نت ها (که بخارط **#define** های pitches.h با فرکانس برای tone استفاده می شوند) و آرایه **d** دوم شامل ارزش زمانی است. ارزش زمانی بر مبنای نت دایره حساب شده است، و همه **i** 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

- در پروتئوس یک دکمه و یک اسپیکر را به برد وصل می کنیم. برنامه ای بنویسید که به هنگام فشرده شدن دکمه، ملودی Jingle Bells را پخش کند.

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

2. هنگام پخش موسیقی از اسپیکر، پروتونس 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 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 استقاده 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

Ode to Joy

from the "Choral" Symphony

LUDWIG VAN BEETHOVEN

(1770-1820)

Op.125

arr. A.L. Christopherson

5. ملودی امتیازی: (همان # است ولی بجای یک قدم جلو، یک قدم عقب. ۰ نت را عادی می‌کند. ۰ های اول خط تا آخر خط باقی می‌مانند، مگر اینکه با ۰ عادی سازی شوند)

The Imperial March

Music by
John Williams