دستورکار آزمایشگاه ریزپردازنده و زبان اسمبلی

مولفان:

محمد چوپان

سارا تاجرنیا

هلیا هاشمی پور

ریحانه باقری

با تشکر از:

فاطمه واله – مریم گلی

دانشکده مهندسی کامپیوتر

ریزپردازنده و زبان اسمبلی، پاییز ۱۴۰۱، دکتر فربه

قوانین آزمایشگاه ریزپردازنده

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

1. آزمایش ها دارای پیش گزارش و گزارش کار نمی باشند.
2. کد های آزمایش های هر جلسه در اختیار دانشجویان قرار میگیرد. دانشجویان موظف اند که کد ها را مطالعه کرده و بتوانند آن را توضیح دهند. زیرا که پروژه نهایی درس ترکیبی از همین کد ها است.

نکات امنیتی:

۱. 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.

۲. هیچگاه دیودهای نورانی (یا اصولاً هر قطعه‌ای که در آن دیود نورانی به کار رفته مثل هفت قسمتی‌ها و ماتریس‌های LED) را مستقیماً به خروجی برد یا منبع تغذیه وصل نکنید، بلکه آن را با یک مقاومت بین ۱۰۰ تا ۳۳۰ اهمی سری نموده، و سپس وصل کنید.

آشنایی با برد Arduino MEGA ۲۵۶۰

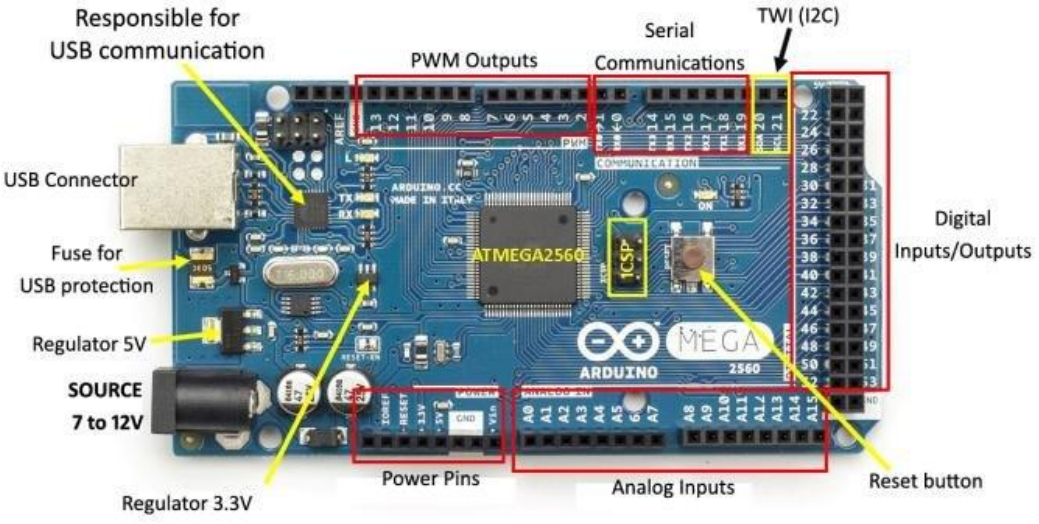
برد Arduino MEGA ۲۵۶۰ یک میکروکنترلر بر پایه ATmega۲۵۶۰ از شرکت Atmel است. این برد دارای ۵۴ پین دیجیتال ورودی/خروجی (که ۱۵ پین می‌تواند به عنوان خروجی PWM استفاده شود)، ۱۶ ورودی آنالوگ، ۴ پورت UARTs (پورت‌های سریال سخت‌افزاری)، یک کلاک ۱۶ مگاهرتزی، یک کابل ارتباطی USB، یک پاور جک، یک ICSP header، و یک دکمه ریست است.

این برد عملا همه‌ی چیزهایی که برای کار با میکروکنترلر نیاز است را شامل می شود. 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 ۲۵۶۰ می‌تواند با اکثر شیلدهای طراحی شده برای Uno و همچنین بردهای قدیمی Duemilanove or Diecimila کار کند.

Arduino MEGA ۲۵۶۰ همانند دیگر بردهای Arduino با برنامه (Arduino Software IDE) برنامه‌ریزی می‌شود که نحوه‌ی نصب و اتصال برد به آن در پیوست آورده شده است.

مشخصات:

• میکروکنترلر:	ATmega2560
• ولتاژ عملیاتی:	۵ ولت
• ولتاژ ورودی (پیشنهادی):	۷ تا ۱۲ ولت
• دامنه مجاز ولتاژ ورودی :	۶–۲۰ ولت
• پین‌های دیجیتال ورودی/خروجی:	۵۴ (۱۵ تای آن خروجی PWM تولید می‌کنند).
• پین‌های ورودی آنالوگ:	۱۶ عدد
• جریان DC هر پین ۳,۳ ورودی/خروجی:	۲۰ میلی آمپر
• جریان DC هر پین ۵ ولت:	۵۰ میلی آمپر
• حافظه فلش:	۲۵۶ کیلوبایت که KB۸ برای bootloader است.
• SRAM	۸ کیلوبایت
• سرعت ساعت:	۱۶ مگاهرتز



نشان‌گرهای LED:

نشانگر 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 ۰، ۱ و ۱۴ تا ۱۹ که برای ارتباط سریال هستند و دومی LEDهای TX و RX. چراغ مربوط به TX هنگام ارسال داده سریال متناسب 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 سوکت های center-positive به قطر ۲,۱ میلی متر) 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 ۶ تا ۲۰ ولت کار کند. اگر ولتاژ منبع تغذیه پایین‌تر از ۷ ولت باشد روی ولتاژ پین‌ها اثر خواهد گذاشت و ممکن است ولتاژ خروجی آن‌ها کمتر از ۵ ولت شود و حتی نوساناتی را به وجود آورد. ولتاژ بیش از ۱۲ ولت نیز می‌تواند موجب افزایش دمای رگولاتور و در نتیجه آسیب به برد گردد. ولتاژ پیشنهادی مناسب، بین ۷ تا ۱۲ 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 یا اتصال USB با ۵ 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 ۵ ولت را از طریق رگولاتور موجود بر روی برد فراهم می‌کند. برد می‌تواند از طریق پاورجک (۱۲-۷ ولت) DC، پورت USB (به اندازه ۵ ولت) و یا پین V_{IN} برد (۱۲-۷ ولت)، تغذیه گردد. ولتاژ پین‌های ۵ ولت و ۳,۳ 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 ۳,۳ ولتی، به وسیله ی رگولاتور روی برد فراهم می‌گردد که حداکثر جریان آن ۵۰ میلی آمپر است.
- GND: پین‌هایی که با اتصال به زمین، ولتاژ صفر را فراهم می کنند.

- 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 ۵ یا ۳,۳ ولت، بر روی خروجی‌ها فعال نماید. این قابلیت، به شیلدها امکان می‌دهد تا با برد ۳,۳ ولتی همچون DUE و بردهای AVR-based که با ولتاژ ۵ 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 متفاوت فراهم میکند و مدیریت سطح پایین سخت افزار میکروکنترلر را انجام میدهد. از این رو برنامه ای که بر بستر آردوینو نوشته می شود قابلیت آن را دارد که بر میکروکنترلر های مختلف با سخت های افزار های متفاوت 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 شئ گرا در C++ مانند کلاس ها، ارث بری، اینترفیس و... بهره مند شد. البته باید توجه داشت که در برنامه نویسی شی گرا در میکروکنترلر ها به دلیل منابع سخت افزاری محدود، همه قابلیت های زبان C++ پیاده سازی نشده است.

توجه :

با توجه به شرایط پیش آمده در این ترم ۴ آزمایش خواهیم داشت که کد های آن ها به شما داده می شودو تنها در پروژه نهایی نیاز به کد زدن دارید. و نیازی به انجام قسمت های مرتبط با کد نیست.

کدهای آزمایش های هر هفته در آدرس زیر موجود می باشد لطفا قبل از آمدن به کلاس کد ها رو آماده کنید.

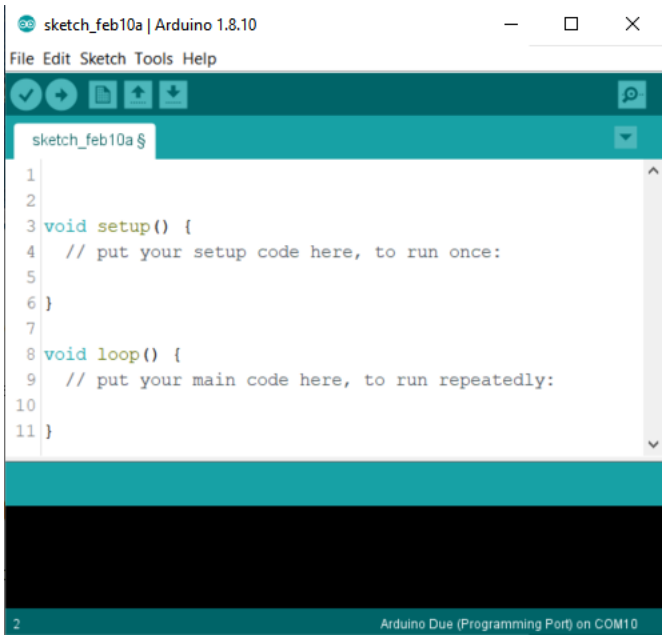
https://github.com/mohamadch91/microprocessor-LAb

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.

پروژه های نمونه ای اضافه 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 تحویل کد های آزمایشگاه لازم می باشد، این پروژه های نمونه را بررسی کنید.

پلتفرم آردوینو یک IDE نیز برای برنامه‌نویسی بردهای آردوینو و همچنین ارتباط سریال با برد به نام Arduino IDE فراهم میکند. این برنامه متن‌باز و رایگان است. می توانید نسخه دسکتاپ آن را نصب نمایید یا از نسخه برخط آن (https://create.arduino.cc/) برای نوشتن برنامه و آپلود آن بر روی برد استفاده کنید.

شکل زیر محیط برنامه Arduino IDE را نشان می‌دهد.



البته این IDE برخی توانایی های محیط توسعه مدرن را ندارد، اما خوشبختانه تنها محیط توسعه آردوینو موجود نیست و برای نمونه PlatformIO که بر روی ویرایشگر هایی مانند VS Code وAtom اجرا می شود، قابلیت های بیشتری را ارائه میکند.

همچنین می توان از محیط Eclipse C/C++برای توسعه پروژه های آردوینو بهره برد.

عملکرد منوهای موجود در Arduino IDE

Verify: برای بررسی خطاهای برنامه‌ی نوشته شده از این گزینه استفاده می‌کنیم.

Upload: برای آپلود کردن برنامه‌ی نوشته شده روی برد از این گزینه استفاده می‌کنیم. (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 خواهید کرد. چند ثانیه صبر نمایید، در زمان آپلود دو عدد Led با نام‌های Rx و Tx روی برد چشمک خواهند زد. در صورتی که آپلود برنامه با موفقیت انجام شود، در نوار وضعیت، پیغام "Done Uploading" را خواهید دید.

برنامه‌نویسی در آردوینو ۳ بخش کلی دارد:

۱. پیش‌پردازش‌ها: در بخش پیش‌پردازش، کتابخانه‌ها و یا متغیرهایی که محلی نیستند تعریف و فراخوانی می‌شوند.

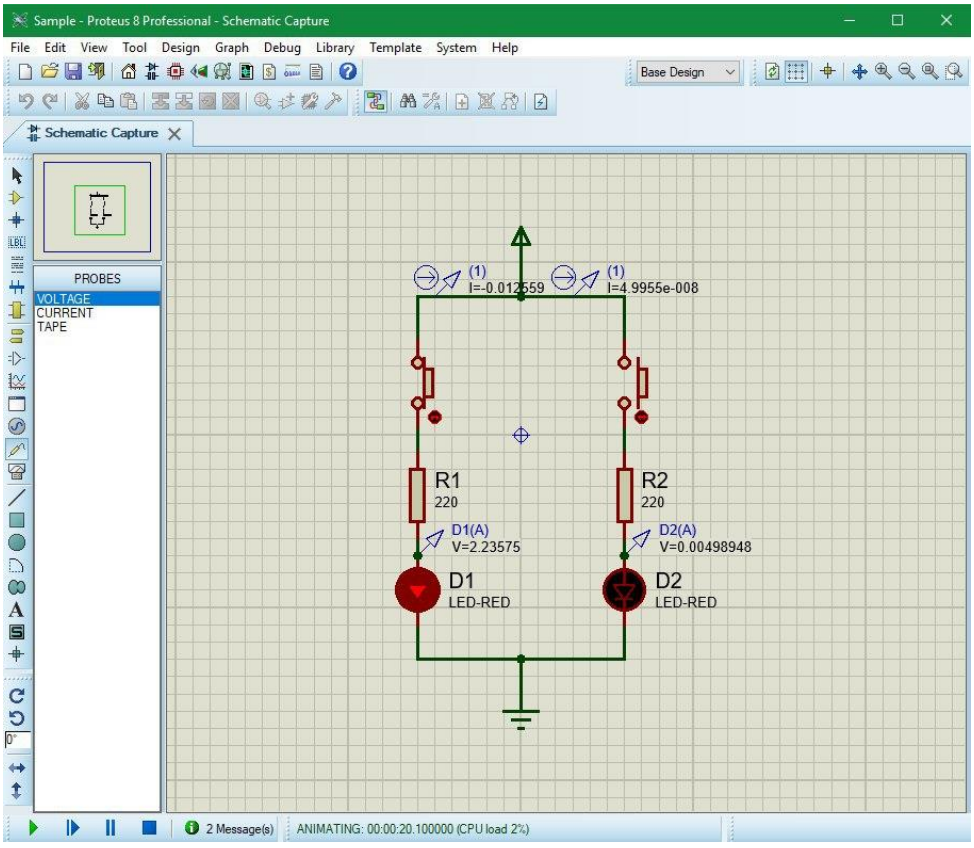
۲. حلقه setup: در این بخش کارهایی که باید یک‌بار انجام شوند را تعریف می‌کنیم. به طور مثال، تنها کافی است یک‌بار تعریف کنیم که یک پین دیجیتال ورودی باشد یا خروجی و یا این‌که به برد اطلاع دهیم که می‌خواهیم از پورت سریال استفاده کنیم.

۳. حلقه loop: حلقه loop یک حلقه بی‌نهایت است و کارهایی که باید به طور متناوب انجام شوند در این حلقه نوشته می‌شود. به طور مثال، اگر بخواهیم یک سنسور دما را به برد وصل کنیم و برد دمای محیط را بر روی 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 باینری کامپایل شده برای پروتئوس را بسازید که در Sketch Folder ذخیره می‌شود و می‌توانید با انتخاب Show Sketch Folder در زیر این گزینه، به آن دسترسی پیدا کنید.

نرم‌افزار شبیه سازی Proteus

این ترم به دلیل برگزاری حضوری آزمایشگاه، آزمایش‌ها در بستر شبیه‌سازی Proteus انجام نخواهند 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 میکروکنترلرها در مدار را فراهم می‌کند. از این رو، می‌توان از این نرم‌افزار برای شبیه‌سازی آزمایش‌های درس ریزپردازنده بهره برد.



چگونگی انجام شبیه سازی آزمایش‌ها در محیط Proteus:

نخست مدار آزمایش را در نرم‌افزارProteus 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 ATmega۲۵۶۰ کامپایل کنید. بدین منظور، ابتدا برد را از طریق Tools -> Board -> Arduino Mega انتخاب کنید و سپس گزینه Verify را بزنید.

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

هم‌چنین راه دیگری نیز برای دریافت فایل باینری در محیط Arduino IDE وجود دارد، پس از کامپایل شدن برنامه، گزینه‌ی Export Compiled Binary در منوی Sketch را انتخاب کنید. این کار، فایل HEX برنامه را در پوشه‌ی Sketch می‌ریزد و می‌توان با انتخاب گزینه‌ی 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 (Polling) و وقفه محور (Interrupt-Driven) برای مدیریت واحد های جانبی
- مقایسه دو روش سرکشی و وقفه محور

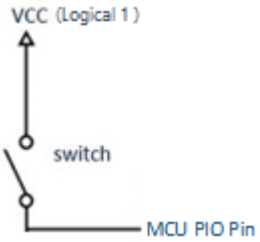
قطعات مورد نیاز:

- برد Arduino - UNO
- دیود نورانی LED
- کلید
- مقاومت ۲۲۰Ω
- مقاومت ۱۰KΩ

مقدمه:

مدارهای Pull-Up و Pull-Down:

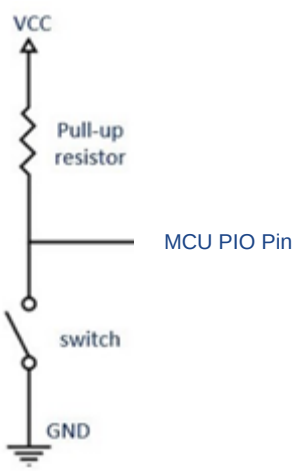
مدار زیر را در نظر بگیرید:

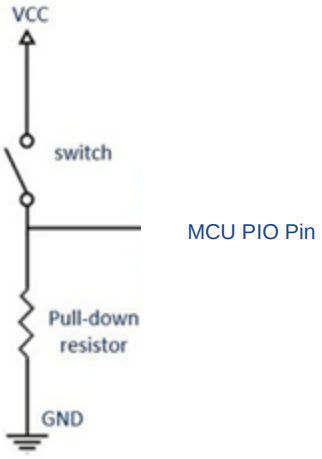


فرض کنید میخواهیم از این مدار برای دریافت اینکه چه زمانی کلید (Switch) بسته شده است استفاده کنیم. برای این کار سطح ولتاژ منطقی (۱ یا ۰) بر روی پایه میکرو (MCU PIO Pin) را پیوسته بررسی میکنیم و زمانی که برابر با ۱ شد را زمان بسته شدن کلید در نظر میگیریم. به عبارتی دیگر، زمانی که کاربر دکمه مربوط را فشار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

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

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





بدیهی است تفاوت 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 بایست از دستورات delay()، digitalWrite()، digitalRead()،pinMode استفاده کنید.

وقفه:

وقفه پاسخی است که پردازنده به هنگام رخ دادن 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 برعهده 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 تا رخ 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 رخ دادن 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 رخ دادن اتفاق مورد نظر یعنی فشرده شدن کلید در شرایط گوناگون مدیریت پذیر خواهد بود.

پرسش: آیا رخ دادن 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 ۲۵۶۰ و شیوه پیاده سازی وقفه ورودی را بدست آورید.

دستوری که برای فعال سازی مدیریت وقفه روی پایه مد 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 ATmega۲۵۶۰ می تواند رخ دادن آن ها را بفهمد و اعلام کند بنویسید.

شرح آزمایش:

این آزمایش دربردارنده چندین LED (دلخواه اما بیشتر از ۵ عدد) و سه دکمه می باشد. که مدار آن بر روی برد در شکل ۱-۱ و مدار شماتیک آن در شکل ۱-۲ نمایش داده شده است. در ابتدا LED ها خاموش می باشند. با هر بار فشردن دکمه ی 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

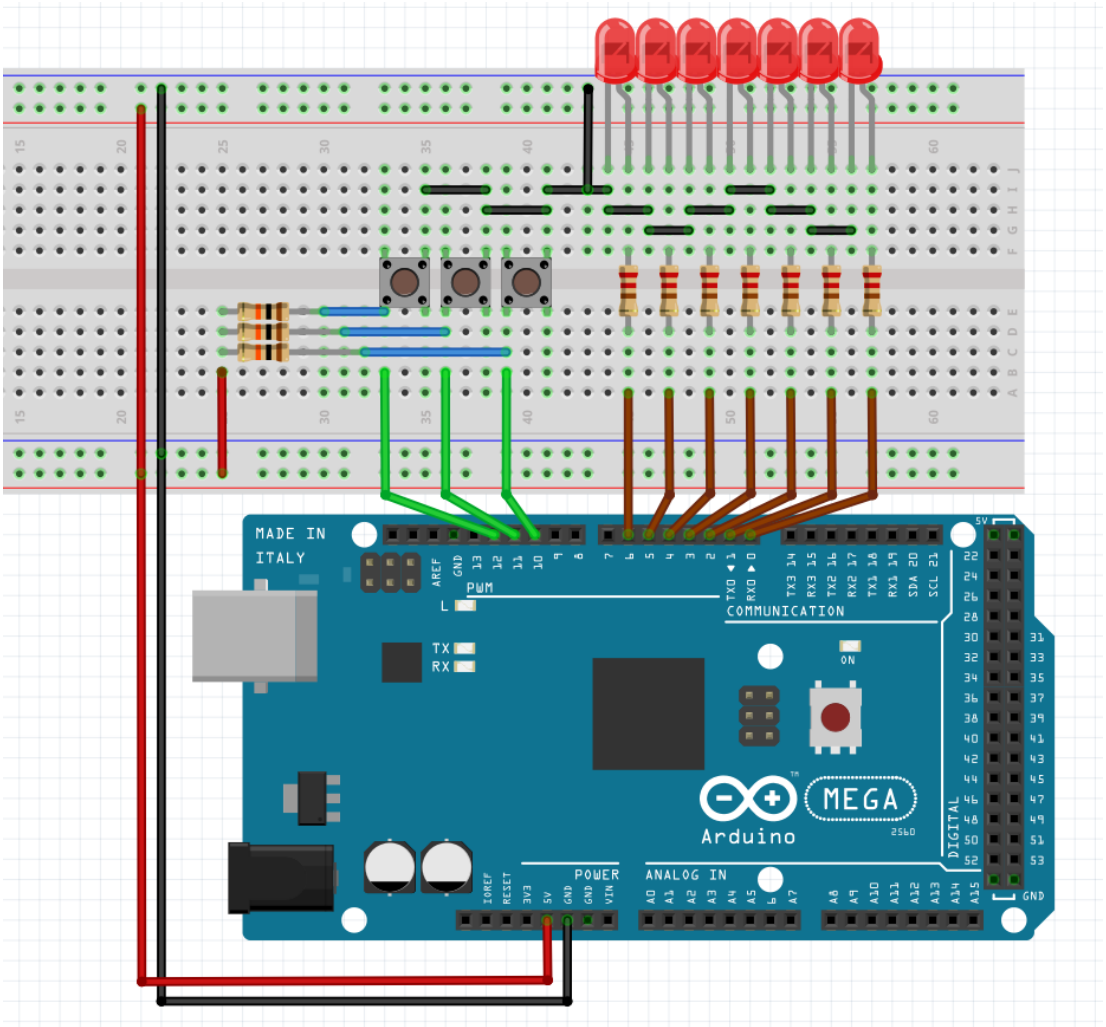
۱. برنامه آزمایش را به روش سرکشی بنویسید و پس از آن که از درستی کارکرد مدار و برنامه خود مطمئن شدیدگام دومرو انجام دهید.

۲. به پرسشهای زیر پاسخ دهید:

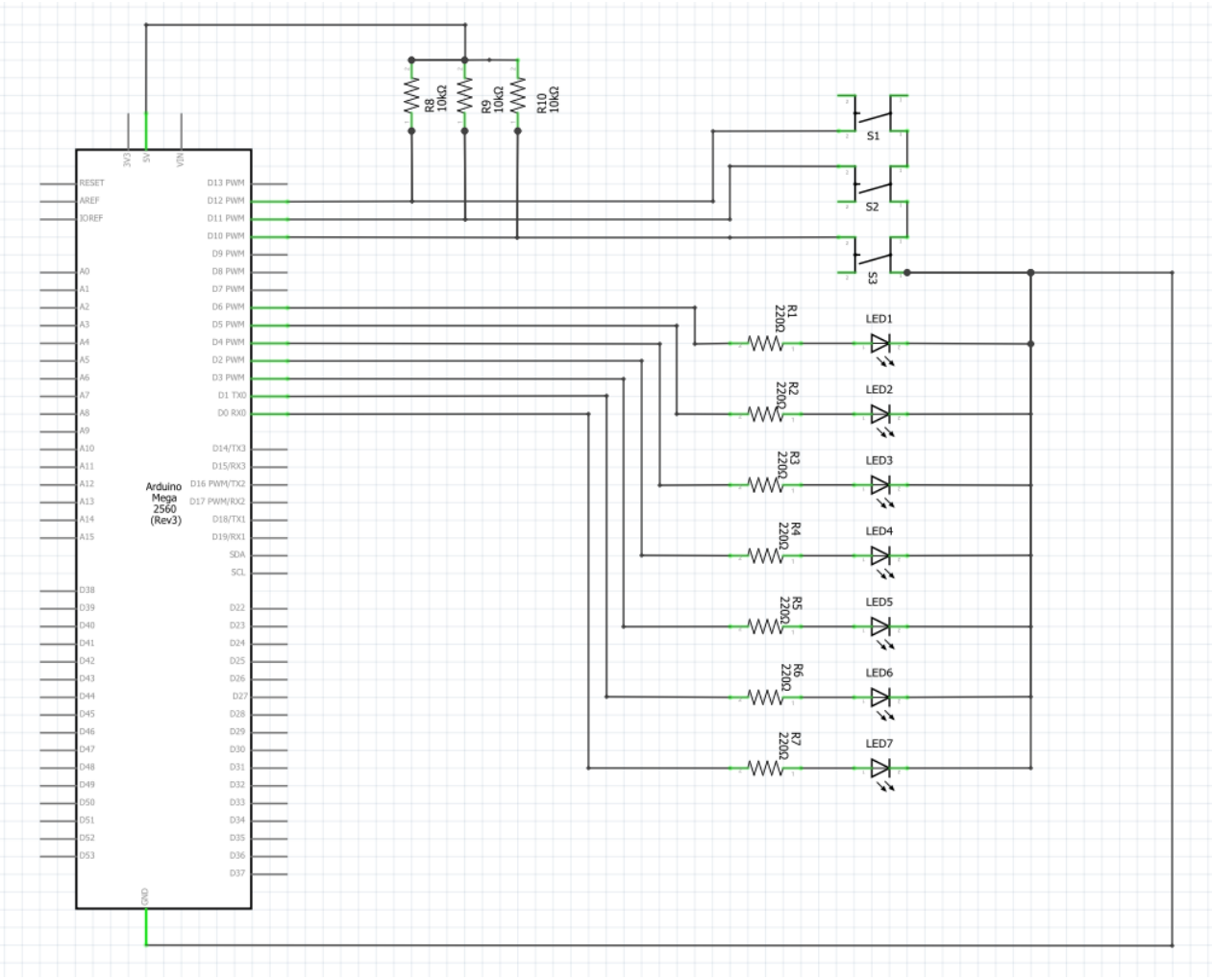
- اگر دکمه را در حالت فشرده برای زمان طولانی نگه داریم چه اتفاقی خواهد افتاد؟ آیا با منطق کارکرد خواسته شده سازگار است؟ چه راه حلی برای این مشکل (در صورت وجود) می توان پیشنهاد کرد؟
- فرض کنید می خواهیم برد مورد نظر علاوه بر فراهم کردن کارکرد خواسته شده در بالا، عمل دیگری را نیز به صورت زمان دار انجام دهد. برای نمونه در کنار کارکرد بالا، وضعیت روشن یا خاموش بودن یک LED را نیز هر ۵ ثانیه یک بار تغییر دهد. روشی برای افزودن این کارکرد تازه به برنامه پیشنهاد دهید.
- فرض کنید میخواهیم کارکرد دیگری را به دستگاه اضافه کنیم به این صورت که در صورت یک شدن یک پایه عملیات مشخصی را به عنوان پاسخ انجام دهد.(محدودیت زمانی برای پاسخ دادن وجود دارد) هیچ یک از اتفاق های یک شدن پایه نباید از دست برود (بی پاسخ بماند). و یک شدن پایه نیز در هر زمانی ممکن است رخ دهد.آیا برنامه شما –که به روش سرکشی واحد های جانبی را بررسی میکند– می تواند در هر شرایطی (مثلا هنگام فشرده شدن کلید) این کارکرد را فراهم کند؟

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

۳. با پاسخ به پرسش های بالا می توان دریافت که روش سرکشی برای کنترل واحد های 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.



شکل ۱-۱


R8
10kΩ
R9
10kΩ
R10
10kΩ
S1
S2
S3
R1
220Ω
R2
220Ω
R3
220Ω
R4
220Ω
R5
220Ω
R6
220Ω
R7
220Ω
LED1
LED2
LED3
LED4
LED5
LED6
LED7
Arduino
Mega
2560
(Rev3)
RESET
AREF
IOREF
3V3
5V
VIN
GND
D13 PWM
D12 PWM
D11 PWM
D10 PWM
D9 PWM
D8 PWM
D7 PWM
D6 PWM
D5 PWM
D4 PWM
D2 PWM
D3 PWM
D1 TX0
D0 RX0
D14/TX3
D15/RX3
D16 PWM/TX2
D17 PWM/RX2
D18/TX1
D19/RX1
SDA
SCL
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D38
D39
D40
D41
D42
D43
D44
D45
D46
D47
D48
D49
D50
D51
D52
D53
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
D36
D37

شکل ۲-۱