برنامه نویسی با برد آردوینو

آشنایی با برنامه نویسی، رباتیک، سخت افزار و الکترونیک

تالیف: سپهر نعیمی

www.NicerLand.ir

سرشناسه	:	جعفری نعیمی، سپهر، ۱۳۶۱-
عنوان و نام پدیدآور	:	برنامه نویسی با برد آردوینو: آشنایی با برنامه‌نویسی، رباتیک، سخت‌افزار و الکترونیک/ نویسنده سپهر جعفری نعیمی.
مشخصات نشر	:	کرج : رهام اندیشه، ۱۳۹۹.
مشخصات ظاهری	:	۲۴۹ ص.
شابک	:	978-622-6800-85-3
وضعیت فهرست نویسی	:	فیپا
عنوان دیگر	:	آشنایی با برنامه‌نویسی، رباتیک، سخت‌افزار و الکترونیک.
موضوع	:	آردوینو (کنترل‌کننده برنامه‌پذیر)
موضوع	:	Arduino (Programmable controller)
موضوع	:	آردوینو (کنترل‌کننده برنامه‌پذیر) -- برنامه‌نویسی
موضوع	:	Programming -- Arduino (Programmable controller)
موضوع	:	میکروکنترلرها -- برنامه‌نویسی
موضوع	:	Microcontrollers -- Programming
رده بندی کنگره	:	TJ۲۲۳
رده بندی دیویی	:	۸۹۵/۶۲۹
شماره کتابشناسی ملی	:	۶۱۳۷۲۱۵

برنامه نویسی با برد آردوینو

آشنایی با برنامه نویسی، رباتیک، سخت افزار و الکترونیک

- تالیف: سپهر نعیمی
- ناشر: رهام اندیشه
- نوبت چاپ: اول ۱۳۹۹
- شمارگان: ۱۰۰ جلد
- تعداد صفحات: ۲۴۹ صفحه
- قیمت: ۴۹۰۰۰ تومان
- شابک: ۳-۸۵-۶۸۰۰-۶۲۲-۹۷۸
- مرکز چاپ و نشر: کرج، میدان نبوت، برج سفید، طبقه ٤، تلفن: ۰۹۱۲۶۶۰۹۷۲۶

جهت ارتباط با نویسنده می توانید از آدرس Sepehr.Naimi@gmail.com استفاده کنید.

درباره کتاب

در طی سالهای اخیر همیشه در گوشه ذهنم به فکر نوشتن کتابی در زمینه کامپیوتر و رباتیک برای عموم بودم. اهدافی که برای تالیف این کتاب داشتم به قرار زیر است:

- خلاقیتها و ابتکار عملی که به طور بالقوه در نوجوانان و سایر افراد وجود دارد بالفعل شود و چه بسا وسایل کاربردی جدیدی برای رفاه عموم پدید آید.
- افراد استعدادها و تواناییهای خود را باور کنند
- دانش آموزان کاربرد دروس ریاضی که در مدرسه می‌خوانند را در عمل تجربه کنند
- کسانی که می‌خواهند در رشته کامپیوتر و یا الکترونیک تحصیل کنند، با خواندن این کتاب دید دقیقتری از ماهیت رشته های کامپیوتر و سخت افزار و الکترونیک پیدا کنند

لذا از سه سال پیش به مرور بر روی موضوعاتی که خوبست در این کتاب پوشش داده شود، کار کردم و کتاب حاضر نسخه اولیه ای از این کتاب است که تقدیم هموطنان عزیز می‌گردد. امیدوارم که در آینده موضوعات دیگری را به این کتاب بیافزایم به گونه ای که بتواند خواننده را از ابتدا تا سطوح پیشرفته همراهی کند.

پیش نیازها

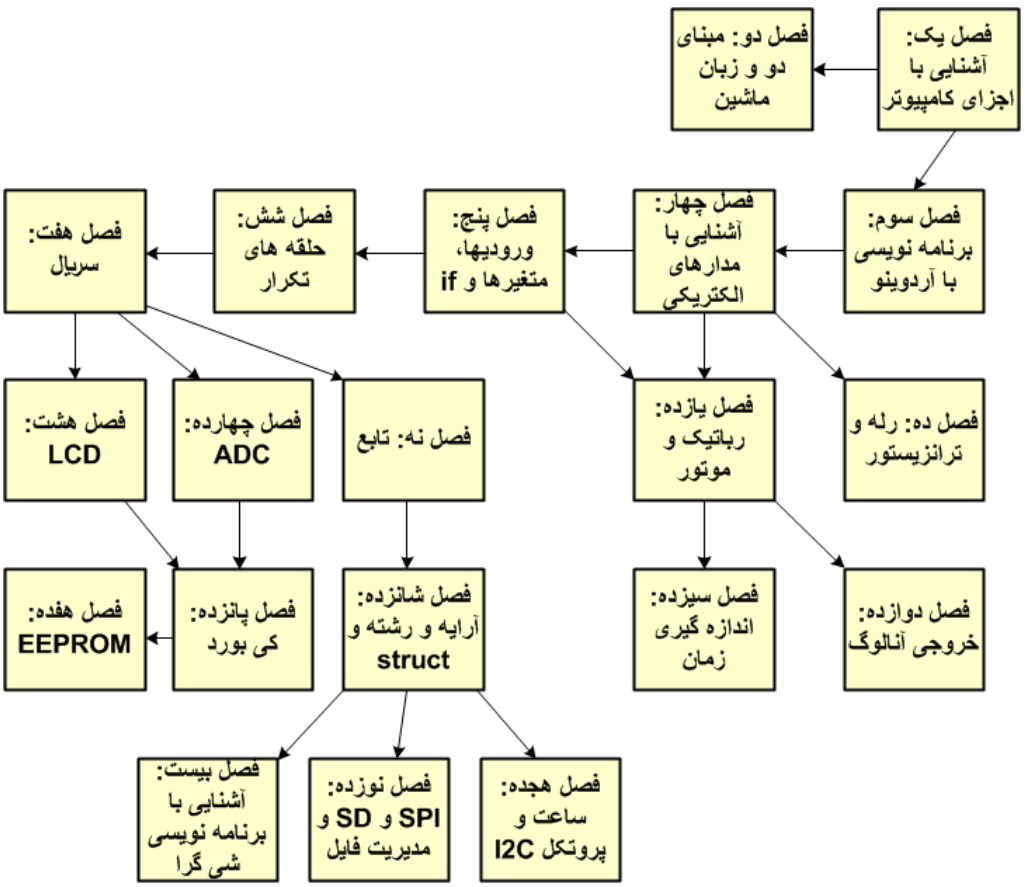
کسانی که مایلند این کتاب را مطالعه کنند می بایست عملیات چهارگانه ریاضی را فرا گرفته باشند. همچنین مقدماتی از انگلیسی را بدانند تا بتوانند دستورات برنامه نویسی را بخوانند و تایپ کنند.

خوانندگان این کتاب برای اینکه فصلهای ۱۲ و ۱٤ را به طور کامل متوجه شوند نیاز به سطوح بالاتری از ریاضی دارند و اگر از مقطع هشتم به بالا باشند راحت تر می‌توانند مطالب این فصول را درک کنند. البته اگر در سطوح پایین تری باشند می توانند کلیاتی از مطالب این فصول را متوجه شوند.

همچنین در بخش۱۳-٤ فرمول مربوط به فاصله ربات از مانع را محاسبه کرده ایم که اگر خوانندگان گرامی مقدمات فیزیک حرکت را خوانده باشند راحت تر محاسبات را متوجه می‌شوند.

ترتیب مطالعه فصول

مطالب فصلهای کتاب به گونه ای نوشته شده که حتی الامکان وابستگی بین فصول مختلف اندک باشد تا خوانندگان گرامی بتوانند بر طبق علایق خود، فصولی که مطالعه می‌کنند را انتخاب نمایند. به طور مثال کسانی که علاقه مند به رباتیک هستند نیازی به خواندن فصل LCD ندارند و می‌توانند فصلهای ۱ تا ۱۱ (به جز LCD) را مطالعه کنند. در شکل زیر رابطه نیازمندی بین فصول با استفاده از فلش نمایش داده شده است. به طور مثال از فصل سه که به سمت فصل چهار فلش کشیده شده است بدین معنی است که قبل از فصل چهار می بایست، فصل سوم را مطالعه کنید. همچنین فلشی که بین فصل چهار و ده کشیده شده است، نشان می‌دهد که قبل از فصل ده می بایست حتما فصل چهار را مطالعه کرده باشید.



درباره نویسنده

نویسنده این کتاب از سال ۱۳۸۲ در زمینه طراحی سیستمهای کنترل و سیستمهای یکپارچه و رباتهای صنعتی فعالیت می‌کند و از سال ۱۳۸٤ در اوقات فراغت کتاب نیز می‌نویسد که از جمله تالیفات او، سری کتابهای دانشگاهی مزیدی و نعیمی است که توسط انتشارات Prentice Hall و Amazon منتشر شده اند. برخی از این کتابها عبارتند از:

- The AVR Microcontroller and Embedded Systems
- ARM Assembly Language Programming and Architecture
- Freescale ARM Cortex-M Embedded Programming
- TI MSP432 ARM Programming for Embedded Systems
- TI Tiva ARM Programming for Embedded Systems
- The STM32F103 Arm Microcontroller & Embedded Systems

از بین کتابهای فوق، کتاب نخست، به فارسی نیز ترجمه شده و در ایران چاپ گردیده است.

فهرست

لطفا جهت دریافت منابع تکمیلی مرتبط با کتاب، به آدرس اینترنتی www.NicerLand.ir/fa مراجعه فرمایید.

فصل ۱: آشنایی با اجزای کامپیوتر

بخش ۱-۱: مقدمه

انسان با هدف غلبه بر ضعف ها و محدودیتهای خود دست به اختراع می زند. به طور مثال، چون که نمی‌تواند فواصل دور را ببیند، دوربین و تلسکوپ را اختراع می کند و چون نمی تواند پرواز کند، هواپیما را می سازد.

از جمله مشکلات بشر، انجام محاسبات ریاضی بوده. به عبارت دیگر، بشر محاسبات ریاضی را خیلی کند و همراه با خطا انجام می دهد. همچنین برخی از کارها، سخت و طاقت فرسا و تکراری هست که بشر پیوسته کوشیده از انجام آنها فرار کند و یا حیوانات و یا انسانهای دیگری را تحت عنوان برده، اسیر و یا سیاه پوست و یا برده اقتصادی به انجام این کارها وادار نماید. (مقصودم از بردگان اقتصادی، افرادی هستند که در ازای پول ناچیز مجبورند کارهای طاقت فرسای تکراری انجام دهند.)

اما پیدایش کامپیوتر، حلال مشکلات فوق است. یعنی کامپیوترها و رباتها، وسایلی هستند که می توانند محاسبات ریاضی را با سرعت زیاد و بدون خطا انجام دهند و کارهای طاقت فرسا و تکراری را سریعا انجام دهند، بدون آنکه خسته شوند.

در طول تاریخ، بشر در آرزوی رسیدن به یک مدینه فاضله (آرمان شهر) بوده که همه انسانها در نهایت آسایش و صلح و دوستی در کنار هم زندگی کنند و اکنون امکانات تاسیس چنین دنیایی بر روی کره زمین فراهم شده است. به عبارت دیگر، برای ایجاد چنین جهانی لازم است که از یک سو، دنیای مادی ترقی و پیشرفت کند و از سوی دیگر می بایست انسانها دارای کمالات اخلاقی و معنوی گردند. اختراعات جدید مثل اینترنت و کامپیوتر، وسایل مورد نیاز جهت ایجاد جهانی متحد و یکپارچه و پیدایش رفاه همگانی را فراهم می‌کند.

بخش ۱-۲: اجزای کامپیوتر

هدف نهایی از اختراع کامپیوترها و رباتها، به وجود آوردن موجوداتی مشابه انسان بوده که تا حد امکان دارای قابلیتهای انسان باشد. کامپیوتر برای اینکه بتواند، دارای قابلیتهایی مشابه انسان باشد، لازم است که اجزایی مشابه انسان داشته باشد.

انسان، با حواس پنج گانه اطراف خویش را حس می کند، کامپیوتر نیز با ورودیهای خود که عبارتند از موس (mouse) و کی بورد (keyboard) و صفحه لمسی (touch) و اسکنر (scanner) و یا سنسورهایی مثل نور و شتاب سنج و قطب نما، اطلاعات را دریافت می کنند.

انسان، با استفاده از دستها و پاهای خود بر محیط اطراف خود تغییراتی ایجاد می کند و با زبان خود به بیان افکار خویش می پردازد. کامپیوتر نیز خروجی هایی مثل LCD، پرینتر (printer)، بلند گو (Speaker)، دست ربات و یا چرخ ربات دارد که با آن مطالبی را بیان می کند و یا تغییراتی را در محیط اطراف خود ایجاد می کند.

انسان، با عقل خود به بررسی اطلاعات و تصمیم گیری می پردازد، کامپیوتر نیز CPU دارد.

انسان، حافظه دارد که در آن دانش خویش را نگهداری می کند، کامپیوتر نیز حافظه دارد که دیتاهای خود را در آن ذخیره سازی می کند.

جدول زیر اجزای انسان را با اجزای کامپیوتر مقایسه می کند.

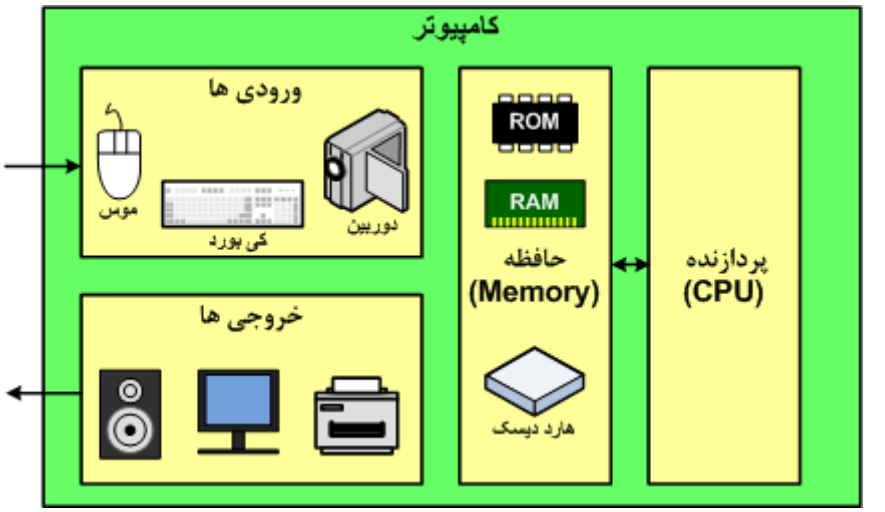
انسان	کامپیوتر
حواس پنج گانه	ورودیها (inputs) مثل موس و کی بورد
زبان و دست و پا	خروجی ها (outputs) مثل بلندگو، صفحه نمایش، پرینتر، دست ربات
عقل انسان	پردازنده (CPU)
حافظه انسان	حافظه کامپیوتر (Memory)

جدول ۱- ۱: مقایسه اجزای انسان و اجزای کامپیوتر

در شکل ۱-۱ که در صفحه بعد آمده است، ساختار کلی کامپیوتر، نمایش داده شده است.

مثلا رباتی را تصور کنید که قرار است کارهای منزل را برایمان انجام دهد. ممکن است ما از این ربات بخواهیم که سر کوچه برود. ماست بگیرد و سپس نان بگیرد و به منزل بازگردد. این ربات از طریق میکروفن خود (که یک ورودی محسوب می شود) صدای ما را می شنود و درخواست ما را در حافظه خود ذخیره می کند. سپس پردازنده (CPU)، دستورات را یکی پس از دیگری اجرا می کند؛ یعنی ابتدا سر کوچه می‌رود. بعد ماست می‌گیرد و سپس نان می گیرد و در نهایت، به منزل باز می‌گردد.

حال به بررسی یکایک این اجزا می پردازیم.



شکل ۱-۱: اجزای یک کامپیوتر

حافظه (Memory)

حافظه می تواند اطلاعات را برایمان ذخیره کند و هر موقع نیاز داشتیم در اختیار ما قرار دهد. برخی از حافظه هایی که در داخل کامپیوتر وجود دارند عبارتند از: هارد دیسک (Hard disk)، CD، DVD، RAM و ROM.

انسان دارای دو نوع حافظه هست کوتاه مدت و دراز مدت. حافظه کوتاه مدت انسان به طور موقت مطالبی را نگه می دارد. مثلا هنگامی که می خواهیم حاصل 2+5+4+9 را حساب کنیم با خود می‌گوییم "۲ و ۵ میشه ۷ با ٤ میشه ۱۱ و ۹ میشه ۲۰." مسلما اینکه حاصل جمع دو عدد نخست ۷ می‌شود و یا سه عدد اول ۱۱ می شود را یک لحظه در حافظه کوتاه مدت خود نگه داشتیم و بعد از چند ثانیه نیز فراموش کردیم. اما اگر از شما بپرسند که اسم شما چیست جواب آن را می دانید. چون که در حافظه دراز مدت خود ذخیره کرده اید.

در کامپیوتر نیز دو نوع حافظه وجود دارد: حافظه های فرّار و حافظه های غیر فرّار. حافظه های فرّار حافظه هایی هستند که با قطع شدن برق و روشن و خاموش شدن کامپیوتر اطلاعات خود را از دست می‌دهند. حافظه RAM، حافظه فرّار محسوب می شود. اما حافظه های غیرفرّار با قطع شدن برق اطلاعات خود را از دست نمی دهد. از جمله حافظه های غیرفرّار، هارد دیسک و Flash memory و CD و ROM هستند.

پردازنده (CPU)

پردازنده، عقل کامپیوتر است و وظیفه اش این است که لیستی از کارهایی که می بایست انجام شوند را یکی پس از دیگری انجام می‌دهد. به طور نمونه، در مثال خرید ماست و نان، به حافظه خود رجوع می‌کند و می‌بیند که اولین کاری که می بایست انجام دهد رفتن به سر کوچه است پس این کار را انجام می‌دهد. سپس به حافظه مراجعه می کند و می بیند که بایست ماست بخرد، پس ماست می خرد و الی آخر.

تمرین

۱. هدف از اختراع چیست؟
۲. هدف بشر از اختراع کامپیوتر چه بود؟
۳. چهار تا از ورودیهای کامپیوتر را نام ببرید.
٤. چهار تا از خروجی های کامپیوتر را نام ببرید.
٥. آیا برنامه های گوشی موبایل در حافظه فرّار ذخیره می شوند و یا غیر فرّار؟ چرا؟

فصل ۲: مبنای دو و زبان ماشین در کامپیوترهای دیجیتال

برای اینکه بتوانیم با کسی صحبت کنیم، می بایست که به زبان او سخن بگوییم. هر زبانی الفبایی دارد که از کنار هم قرار گرفتن آن حروف، کلمات درست می شوند و هنگامی که کلمات با قواعد صحیح در کنار هم قرار می گیرند، جملاتی حاصل می شوند.

الفبای کامپیوترهای دیجیتال، از دو حرف تشکیل شده است: ۰ و ۱. به عبارت دیگر، کامپیوترهای دیجیتال جز ۰ و ۱ چیز دیگری متوجه نمی شوند و جمیع کلمات و دستوراتی که می توانیم به کامپیوتر بگوییم فقط از ۰ و ۱ تشکیل شده اند. مثلا اگر بخواهیم به کامپیوتری بگوییم که "*۵ را با ۷ جمع بزن*" ممکن است که به او بگوییم 010101010111.

از لحاظ دستور زبان فارسی، جمله "*۵ را با ۷ جمع بزن*" یک جمله امری است. جمیع جملاتی که به کامپیوتر می گوییم جنبه امری دارد و ما از کامپیوتر درخواست می کنیم که کاری را برای ما انجام دهد.

در جمله "*۵ را با ۷ جمع بزن*" فعل جمله، "*جمع بزن*" است و بقیه جمله، بیان می کند که عمل جمع زدن بر روی چه چیزهایی انجام شود. دستوری که به زبان ماشین نوشته شده است نیز، در عمل از دو قسمت تشکیل شده است. قسمتی از آن، از کامپیوتر درخواست میکند که جمع بزند و قسمت دیگر می‌گوید که عمل جمع زدن بر روی ۵ و ۷ انجام شود.

در قسمت بعد می بینیم که چگونه اعداد ۵ و ۷ را به زبان ماشین بیان کنیم.

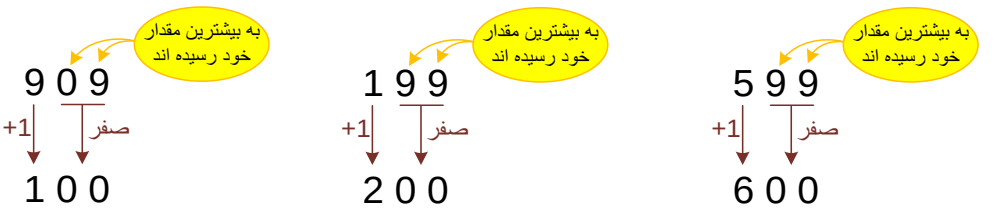
بخش ۲-۱: بیان کردن اعداد، در مبنای دو

در زندگی روزمره هنگامی که بخواهیم اعداد را بیان کنیم از مبنای ۱۰ استفاده می‌کنیم. بدین معنی که هر رقم می‌تواند ۱۰ حالت مختلف داشته باشد و هر یک از اعداد ۰ تا ۹ را می‌تواند شامل شود. زمانی که داریم می‌شماریم، شمارش را از ۰ شروع می‌کنیم و هنگامی که به ۹ رسیدیم، رقم دهگان را یکی زیاد می‌کنیم و رقم یکان را صفر می کنیم که عدد ۱۰ حاصل می‌شود. شکل ۲-۲ را ببینید.

شکل ۲-۱: ترتیب شمارش در مبنای ۱۰ (که در زندگی روزمره از آن استفاده می کنیم)

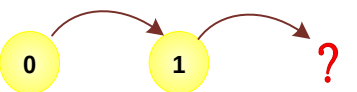
شکل ۲-۲: در مبنای ده، هنگامی که رقم یکان به حداکثر خود برسد، یکان را صفر می کنیم و دهگان را یکی زیاد می کنیم

زمانی که شمارش را ادامه می دهیم و به عدد ۹۹ میرسیم نیز، از آنجایی که هر یک از ارقام به بالاترین مقدار خود رسیده اند، همه آنها صفر می‌شوند و رقم سمت چپ آنها، یکی زیاد می شود و عدد ۱۰۰ به وجود می آید.



شکل ۲-۳: در مبنای ۱۰ ، هنگامی که ارقام یکان و دهگان به حداکثر مقدار خود برسند، صفر می شوند و صدگان یکی زیاد میشود

در مبنای ۲ نیز، طریقه شمارش به همین صورت است. در مبنای دو هر رقم دو حالت دارد: ۰ و ۱. پس اعداد تک رقمی مبنای ۲ عبارتند از ۰ و ۱. شکل زیر را ببینید.

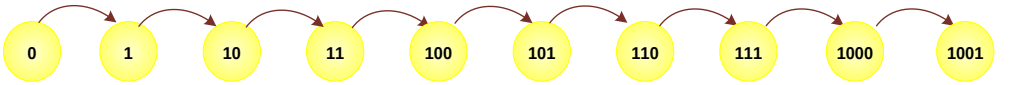


شکل ۲ - ۴: ترتیب شمارش در مبنای ۲

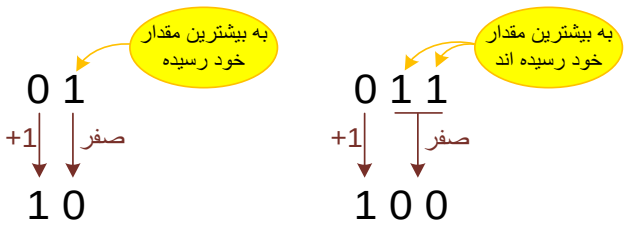
در مبنای ده، هنگامی که به عدد ۹ رسیدیم، با توجه به اینکه ۹ بزرگترین عدد مبنای ۱۰ بود، رقم یکان را صفر کردیم و رقم دهگان را یکی زیاد کردیم. در مبنای دو نیز، عدد ۱ بزرگترین مقدار هر رقم است. پس هر گاه ارقام به ۱ رسیدند، آنها را صفر می کنیم و رقم سمت چپ را یکی زیاد می کنیم و سپس به شمارش خود ادامه می دهیم.

همانطوری که در مبنای ده، پس از عدد ۲۰، رقم یکان را مجددا زیاد می کنیم و اعداد ۲۱ و ۲۲ و غیره به وجود می آیند، در مبنای دو نیز، پس از ۱۰، رقم یکان را زیاد می کنیم که ۱۱ به وجود می آید و در

عدد ۱۱ با توجه به اینکه همه ارقام به بالاترین مقدار خود رسیده اند، پس مقدارشان صفر می‌شود و رقم سمت چپ آنها، یکی زیاد می شود و عدد ۱۰۰ به وجود می آید. شکلهای زیر را ببینید.



شکل ۲-۵: ترتیب شمارش در مبنای ۲



شکل ۲-۶: شمارش در مبنای دو، هنگامی که ارقام به حداکثر مقدار خود می‌رسند

مبنای ده
۰
۱
۲
۳
۴
۵
۶
۷
۸
۹
۱۰
۱۱
۱۲
۱۳
۱۴
۱۵
۱۶
۱۷
۱۸
۱۹
۲۰

الف) مبنای ده

۰
۱
۱۰
۱۱
۱۰۰
۱۰۱
۱۱۰
۱۱۱
۱۰۰۰
۱۰۰۱
۱۰۱۰
۱۰۱۱
۱۱۰۰
۱۱۰۱
۱۱۱۰
۱۱۱۱
۱۰۰۰۰
۱۰۰۰۱
۱۰۰۱۰
۱۰۰۱۱
۱۰۱۰۰

ب) مبنای دو

مبنای دو	مبنای ده
۰۰۰۰	۰
۰۰۰۱	۱
۰۰۱۰	۲
۰۰۱۱	۳
۰۱۰۰	۴
۰۱۰۱	۵
۰۱۱۰	۶
۰۱۱۱	۷
۱۰۰۰	۸
۱۰۰۱	۹
۱۰۱۰	۱۰
۱۰۱۱	۱۱
۱۱۰۰	۱۲
۱۱۰۱	۱۳
۱۱۱۰	۱۴
۱۱۱۱	۱۵
۱۰۰۰۰	۱۶
۱۰۰۰۱	۱۷
۱۰۰۱۰	۱۸
۱۰۰۱۱	۱۹
۱۰۱۰۰	۲۰

ج) مبنای دو و ده در کنار هم

جدول ۲-۱: ترتیب شمارش در مبناهای ۲ و ۱۰

در مبنای ده، هر چقدر صفر به سمت چپ اعداد بگذاریم، در مقدار عدد تاثیری نمی کند. یعنی ۲۱ و ۰۲۱ و ۰۰۲۱ و ۰۰۰۲۱ با هم برابرند. در مبنای دو نیز ما هر چقدر ۰ به سمت چپ اعداد بگذاریم تغییری در مقدار آنها نمی کند. به طور مثال ۱۰ و ۰۱۰ و ۰۰۱۰ و ۰۰۰۱۰ با هم برابر هستند.

در جدول ۲-۱، معادل اعداد ۰ تا ۲۰ را در مبنای دو می بینید.

بخش ۲-۲: طریقه تبدیل اعداد از مبنای ده به مبنای دو

در مبنای ده، رقم یکان دارای ارزش ۱ برابر، رقم دهگان دارای ارزش ۱۰ برابر و رقم صدگان دارای ارزش ۱۰۰ برابر است. به عبارت دیگر، هنگامی که عددی در موقعیت nمین رقم از سمت راست قرار می گیرد، ارزش آن در 10^{n-1} ضرب می شود. مثلا در عدد ۹۵٤، عدد ٤ دارای ارزش 4*1 و عدد ٥ دارای ارزش 5*10 و رقم صدگان دارای ارزش 9*100 است و ارزش ۹٥٤ دارای ارزش زیر است:

954 = 4*1 + 5*10 + 9*100

در مبنای دو، اولین رقم از سمت راست دارای ارزش ۱ برابر، دومین رقم از سمت راست دارای ارزش ۲ برابر، رقم سوم از سمت راست دارای ارزش ٤ برابر است. به عبارت دیگر، هنگامی که عددی در موقعیت nمین رقم از سمت راست قرار می گیرد، ارزش آن در 2^{n-1} ضرب می شود. بنابراین اگر عدد ۱۱۰۱۰ را در مبنای دو، داشته باشیم و بخواهیم ارزش آن را محاسبه کنیم، کافی است که مقدار هر رقم را در ارزش جایگاه آن ضرب کنیم و ارزش کل ارقام را با هم جمع کنیم. یعنی به صورت زیر عمل می کنیم:

$$۱۱۰۱۰ = 0*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 1*2^4$$

$$= 0*1 + 1*2 + 0*4 + 1*8 + 1*16 = 2 + 8 + 16 = 26$$

مثال ۲-۱

اگر کامپیوتر به عنوان جواب، هر یک از اعداد زیر را اعلام کرده باشد، اعلام کنید که جواب، در مبنای ده چند است.

الف) ۱۱۱ ب) ۱۱۰۱۱ ج) ۱۰۰۱۰۱

پاسخ:

الف)

$$111 = 1*2^0 + 1*2^1 + 1*2^2 = 1*1 + 1*2 + 1*4 = 1 + 2 + 4 = 7$$

پس عدد ۱۱۱ در مبنای ده، برابر ۷ میشود.

ب)

$$۱۱۰۱۱ = 1*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 1*2^4$$

$$= 1*1 + 1*2 + 0*4 + 1*8 + 1*16 = 1 + 2 + 8 + 16 = 27$$

ج)

$$۱۰۰۱۰۱ = 1*2^0 + 0*2^1 + 1*2^2 + 0*2^3 + 0*2^4 + 1*2^5$$

$$= 1*1 + 0*2 + 1*4 + 0*8 + 0*16 + 1*32 = 1 + 4 + 32 = 37$$

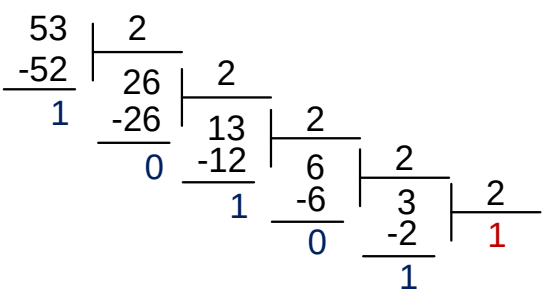
پس حاصل جواب ۳۷ بوده است.

بخش ۲-۳: تبدیل از مبنای ۱۰ به مبنای ۲

در قسمت قبل دیدیم که چگونه اعدادی که کامپیوتر به ما می دهد را به زبان خودمان تبدیل کنیم. حال میخواهیم عددها را از مبنای ده (زبان انسان) به مبنای دو (زبان ماشین) تبدیل کنیم.

برای اینکه عددی را از مبنای ده به مبنای دو تبدیل کنیم، آن را بر دو تقسیم می کنیم. سپس اگر خارج قسمت تقسیم، بزرگتر از ۱ بود، مجدداً خارج قسمت را بر ۲ تقسیم می کنیم و این روند را ادامه می دهیم تا اینکه خارج قسمت، ۱ شود.

به طور مثال، اگر بخواهیم عدد ۵۳ را به مبنای ۲ تبدیل کنیم، ۵۳ را بر ۲ تقسیم می کنیم که خارج قسمت تقسیم ۲۶ و باقیمانده ۱ می شود. (شکل ۲-۷) با توجه به اینکه خارج قسمت بزرگتر از ۱ است، مجدداً خارج قسمت را بر ۲ تقسیم می کنیم که خارج قسمت ۱۳ و باقیمانده ۰ می‌شود.



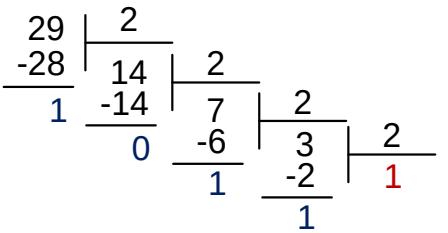
شکل ۲-۷: تبدیل کردن عدد ۵۳ از مبنای ده به مبنای دو

با توجه به اینکه خارج قسمت بزرگتر از ۱ است، مجدداً خارج قسمت را بر ۲ تقسیم می کنیم و این روند را آن قدر تکرار می کنیم که خارج قسمت تقسیم برابر با ۱ شود.

سپس، آخرین خارج قسمت را یادداشت می کنیم و در کنار آن، باقیمانده ها را به ترتیب از آخر به اول، یادداشت می کنیم. عدد ۱۱۰۱۰۱ معادل عدد ۵۳ در مبنای ۲ است.

مثال ۲-۲

عدد ۲۹ را به مبنای ۲ تبدیل کنید.



بنابراین معادل عدد ۲۹ در مبنای دو، ۱۱۱۰۱ هست.

بیت و بایت

برای بیان کردن هر کمیتی از واحدی استفاده می کنیم. به طور مثال برای بیان کردن وزن از واحد گرم و کیلوگرم استفاده می‌کنیم و برای بیان کردن طول از متر و سانتی متر استفاده می کنیم.

در کامپیوتر نیز برای بیان کردن اندازه حافظه، از واحدهای بیت، بایت، کیلو بایت و مگا بایت استفاده می‌کنیم. هر بیت، فضایی هست که می تواند یک عدد ۰ یا ۱ را در خود جا دهد. به عبارت دیگر، محتوای یک بیت می‌تواند، صفر یا یک باشد.

از کنار هم قرار گرفتن ۸ بیت، یک بایت به وجود می‌آید. بنابراین یک بایت، می‌تواند اعداد باینری ۸ رقمی را در خود جای دهد. کوچکترین عددی که در یک بایت جا می‌شود ۰۰۰۰۰۰۰۰ است و بزرگترین عددی که در یک بایت جا می‌شود ۱۱۱۱۱۱۱۱ است. همان طوری که در مثال زیر خواهید دید، عدد ۰۰۰۰۰۰۰۰ معادل ۰ و عدد ۱۱۱۱۱۱۱۱ معادل ۲۵۵ است. بنابراین یک بایت می تواند حاوی اعداد بین ۰ تا ۲۵۵ باشد.

مثال ۲-۳

کوچکترین و بزرگترین عددی که می تواند در یک بایت ذخیره شود چیست؟

پاسخ:

کوچکترین عددی که در یک بایت می توان ذخیره کرد، عدد ۰۰۰۰۰۰۰۰ باینری است که اگر به مبنای ده تبدیل کنیم معادل عدد ۰ می شود.

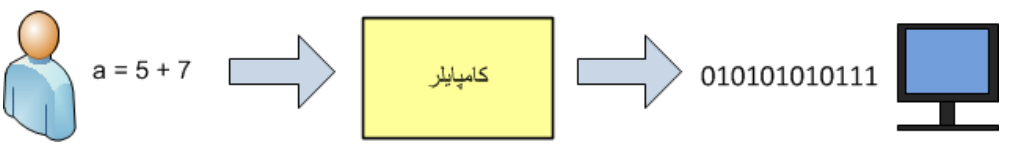
بزرگترین عددی که در یک بایت می توانیم ذخیره کنیم، زمانی است که همه بیتهای آن ۱ باشد. پس عدد ۱۱۱۱۱۱۱۱ باینری بزرگترین عدد است. عدد ۱۱۱۱۱۱۱۱ برابر است با:

$$۱۱۱۱۱۱۱۱ = 1*2^0 + 1*2^1 + 1*2^2 + 1*2^3 + 1*2^4 + 1*2^5 + 1*2^6 + 1*2^7 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$$

بخش ۲-۴: زبان های برنامه نویسی

همان طوری که قبلا گفته شد، زبان قابل فهم برای کامپیوترهای دیجیتال، شامل حروف ۰ و ۱ است و جز ۰ و ۱ چیزی را نمی فهمند. اما برای ما بسیار دشوار خواهد بود اگر که بخواهیم به زبان ماشین صحبت کنیم و مثلا به جای اینکه بگوییم ۵ را با ۷ جمع کن، لازم باشد بگوییم 010101010111. در عین حال که امکان اینکه اشتباه کنیم و یکی از یک ها را صفر و یا یکی از صفرها را یک بنویسیم زیاد است و بعدا نیز پیدا کردن چنین اشتباهی بسیار مشکل خواهد بود.

در ابتدای پیدایش کامپیوتر، برنامه ها را به زبان ماشین می نوشتند. اما پس از مدتی این ایده به ذهن انسان رسید که مترجمی بین انسان و کامپیوتر قرار گیرد تا بتوانیم به زبانی نزدیک به زبان انسان، درخواستهای خود را مطرح کنیم و این مترجم، این درخواستها را به زبان ماشین تبدیل نماید. بنابراین زبانهای برنامه نویسی به وجود آمدند تا ما بتوانیم به طور استاندارد و دقیق افکار و درخواستهای خود را بیان کنیم و برنامه هایی به نام کامپایلر ایجاد شدند که برنامه های ما را به زبان ماشین ترجمه می کنند.



شکل ۲-۸: کامپایلر دستورات برنامه نویس را به زبان ماشین تبدیل می کند

در فصلهای آینده با زبان سی آشنا می‌شویم و کامپایلر آردوینو، برنامه های ما را به زبان ماشین ترجمه خواهد کرد.

تمرین

۱. در مبنای ۱۰، هر رقم چند حالت می تواند داشته باشد؟

۲. در مبنای ۲، هر رقم چند حالت می تواند داشته باشد؟ نام ببرید.

۳. کامپیوترهای دیجیتال در چه مبنایی کار می کنند؟

٤. اعداد زیر را از مبنای ۲ به مبنای ۱۰ تبدیل کنید.

الف. ۱۱۰۱۰۱ ب. ۱۰۱۱۱۰۱ ج. ۱۱۱۱۰۱۱۱ د. ۱۰۰۰۰۰۱۱

۵. اعداد زیر را از مبنای ۱۰ به مبنای ۲ تبدیل کنید.

٦. هر بایت معادل چند بیت است؟

۷. بزرگترین عددی که می توانیم با ٤ بیت بیان کنیم چیست؟ معادل آن در مبنای ۱۰ چه عددی است؟

۸. کامپایلر چه کاری انجام می دهد؟

فصل ۳: آغاز برنامه نویسی با برد آردوینو (Arduino)

امروز می‌خواهیم استفاده از برد آردوینو را شروع کنیم. بردهای آردوینو در عین ساده بودن امکانات فراوانی دارند و با قیمتهای مناسب در سراسر جهان در دسترس هستند و به همین خاطر، مقبولیتِ فراوانی پیدا کرده اند و بسیاری از نوجوانان و دانشجویان و علاقه مندان به سخت افزار، از این بردها به عنوان سرگرمی و یا پروژه های کاری و یا دانشجویی استفاده می‌کنند. بعضی از وسایلی که می‌توانیم با استفاده از آردوینو بسازیم عبارتند از:

- رباتها: از بردهای آردوینو می توانیم در طراحی رباتهای متحرک مختلف از قبیل دنبال کننده خط، فوتبالیست و حل کننده ماز و نیز رباتهای صنعتی استفاده کنیم.
- سیستم های کنترل: می توانید سیستم های گرمایشی و سرمایشی و روشنایی و سایر وسایل منزل خود را به این برد متصل کنید و روشن و خاموش کنید.

بخش ۳-۱: آشنایی با برد آردوینو

عکس یک برد آردوینو را در شکل ۳-۱ می بینید.

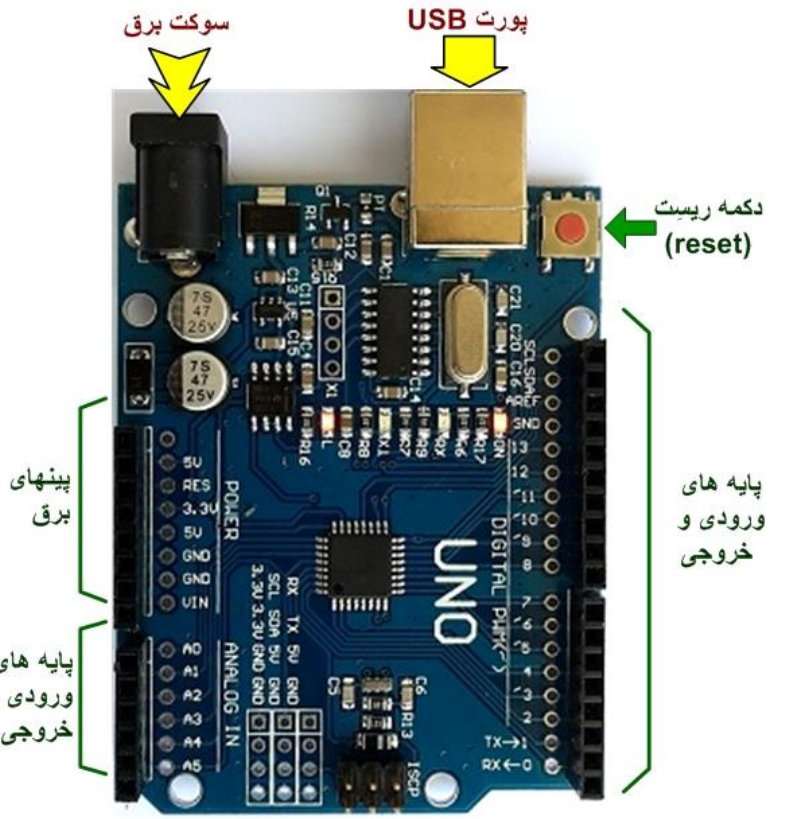
پایه های ورودی و خروجی

برد آردوینو تعدادی پایه ورودی و خروجی دارد که می توانیم به آنها وسایل برقی مختلف، از قبیل چراغ، کولر، موتور و غیره را متصل کنیم و از طریق برنامه ای که برای آردوینو می نویسیم وسایل برقی را روشن و خاموش و کنترل کنیم. همچنین می توانیم سنسورهای مختلف از قبیل سنسور دما و رطوبت و نور و برخورد ربات با مانع و موقعیت سنجی و غیره را متصل کنیم تا اطلاعاتی را از محیط اطراف دریافت کنیم.

پورت یو اس بی (USB)

برد آردوینو، یک پورت یو اس بی (USB) دارد که از طریق آن، به پورت یو اس بی کامپیوتر متصل می‌شود. از طریق پورت یو اس بی، ما بر روی برد آردوینو برنامه می‌ریزیم (اصطلاحاً آن را پروگرام (Program) می‌کنیم.) همچنین برق مورد نیاز جهت کار کردن برد آردوینو را فراهم می کنیم.

همان طوری که در فصلهای بعدی خواهیم دهید از پورت USB می توانیم جهت تبادل اطلاعات بین برد آردوینو و کامپیوتر نیز استفاده نماییم.



شکل ۳-۱: اجزای برد آردوینو

سوکت ورودی برق

برد آردوینو دارای سوکت ورودی برق می‌باشد. به طور معمول، برق برد آردوینو را از طریق سوکت USB، تعمین می کنیم و نیازی به اتصال آداپتور به برد آردوینو نیست. اما در پروژه هایی که ماژولهای خارجی به برد آردوینو متصل می کنیم و مصرف برق برد آردوینو زیاد می شود، می توانیم به جای پورت USB، سوکت برق آردوینو را به یک آداپتور ۹ ولت متصل کنیم.

پینهای برق

هنگامی که رباتهای متحرکی را می سازیم، تمایل نداریم که سیمی را جهت فراهم کردن برق سیستم، به ربات متصل کنیم. بلکه مایلیم از طریق باتری برق مورد نیاز جهت ربات را فراهم کنیم. در چنین حالتی، از طریق پورت USB و یا سوکت ورودی برق، برق مورد نیاز برد آردوینو را فراهم نمی کنیم. بلکه سر مثبت باتری ۵ ولت را به پایه VIN از برد آردوینو و سر منفی آن را به پایه GND از برد متصل می کنیم.

بخش ۳-۲: برنامه نویسی با آردوینو

برنامه نویسی برای کامپیوتر

هنگامی که برنامه هایی را می خواهیم برای کامپیوتر بنویسیم، ابتدا خودمان را در جای کامپیوتر می-گذاریم و به زبان فارسی بیان می کنیم که چه کارهایی می بایست انجام دهیم تا هدف مورد نظر تحقق پیدا کند. سپس با استفاده از دستورات برنامه نویسی همین کارها را از کامپیوتر می خواهیم که انجام دهد.

به طور مثال، فرض کنید، بخواهیم یک لامپ چشمک زن درست کنیم؛ به گونه ای که این لامپ ۱ ثانیه روشن و ۱ ثانیه خاموش باشد. اگر بخواهیم چنین کاری انجام شود:

- لامپ را روشن می کنیم
- یک ثانیه صبر می کنیم
- لامپ را خاموش می کنیم
- یک ثانیه صبر می کنیم
- دوباره کارهای فوق را تکرار می کنیم.

به عنوان مثالی دیگر، فرض کنید بخواهیم هر موقع دمای اتاق از ۲۹ درجه بیشتر شد کولر روشن شود. برای این کار:

- به دماسنج نگاه می کنیم تا ببینیم دمای اتاق، چند درجه است.
- اگر دمای اتاق، بیشتر از ۲۹ درجه بود، کولر را روشن می کنیم و در غیر این صورت کولر را خاموش می کنیم.

در ادامه خواهیم دید که چگونه می توانیم برنامه ها را برای کامپیوتر بنویسیم.

ساختار کلی برنامه های سخت افزاری

برنامه هایی که برای سخت افزار می نویسیم معمولا دارای دو قسمت هستند:

۱- قسمت مقداردهی اولیه و آماده سازی سیستم برای کار

۲- کار عادی برنامه

بعضی از وسایل سخت افزاری، قبل از اینکه بتوانند استفاده شوند لازم است که آماده استفاده شوند. به طور مثال، همین کامپیوتر شما هنگامی که روشن می‌شود، ابتدا بوت (boot) می شود و سخت افزارهای خود را برای استفاده شدن، آماده می کند و سپس به کار عادی خود می پردازد.

بنابراین برنامه هایی که برای آردوینو می نویسیم نیز دارای دو قسمت است:

۱- قسمت setup که وظیفه مقداردهی اولیه وسایل را در آن انجام می دهیم.

۲- قسمت loop که کار عادی برنامه در آن انجام می شود.

در زیر ساختار کلی یک برنامه آردوینو را می بینید:

```
void setup() {
// مجموعه دستوراتی که برای آماده به کار کردن سخت افزارها لازم است،
در اینجا قرار می گیرد //
}

void loop() {
//قسمت اصلی برنامه در اینجا قرار می گیرد//
}
```

به طور مثال، برنامه زیر هر یک ثانیه یکبار، پایه ۱۳ از میکروکنترلر که به آن چراغی متصل است را روشن و خاموش می کند:

برنامه ۳-۱: برنامه چشمک زن

```
// برنامه چشمک زن
// این برنامه هر ثانیه یکبار پایه ۱۳ برد آردوینو را روشن و خاموش می کند
// بنابراین اگر چراغی به پایه ۱۳ متصل کنیم، یک ثانیه روشن و یک ثانیه
// خاموش خواهد بود.
void setup() {
    pinMode(13, OUTPUT); //پایه 13 خروجی باشد
}

void loop() {
    digitalWrite(13, HIGH); //پایه 13 را روشن کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(13, LOW); //پایه 13 را خاموش کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
}
```

در زیر به توضیح برنامه فوق می پردازیم:

ساختار دستورات در زبان سی

همان طور که در فوق می بینید، در زبان سی، هر دستور، با یک ; ختم میشود. برای اینکه برنامه هایی که می نویسیم خواناتر باشد، هر دستور را در یک خط می‌نویسیم و هیچگاه بیش از یک دستور را در یک خط نمی نویسیم.

نوشتن توضیحات در زبان سی

در زبان سی، هر گاه بخواهیم برای خود توضیحی بنویسیم دو تا خط مورب به شکل // می گذاریم. علامت // به کامپایلر می گوید که هر آنچه که جلوی // نوشته شده است را برای خودمان یادداشت کرده‌ایم و کامپایلر نبایست آنها را به زبان ماشین تبدیل کند. به طور مثال در برنامه فوق، در جلوی هر دستور برای شما توضیح داده ام که آن دستور چه کاری را انجام میدهد. اگر ما علامت // را در جلوی توضیحات نگذاریم، کامپایلر به ما خطا خواهد داد و خواهد گفت که در زبان سی چنین دستوراتی وجود ندارد. زیرا این توضیحاتی که نوشته ایم برای کامپایلر بی معنی است. اما وجود علامت // به کامپایلر می‌فهماند که هر نوشته ای که پس از // است را نادیده انگارد. اگر چه ما، برای تسهیل در درک شما، توضیحات را به زبان فارسی نوشته ایم، ولی پیشنهاد می‌شود که از نوشتن توضیحات فارسی در محیط آردوینو و سایر محیطهای برنامه نویسی خودداری کنید.

بدنه setup و loop

بدنه setup با { آغاز می شود و با } خاتمه می پذیرد و جمیع دستوراتی که می خواهیم در setup قرار دهیم می بایست که بین این { و } قرار دهیم. بدنه loop نیز همین طور است و با } آغاز و به { پایان می یابد.

همان طوری که در فصل نه خواهیم دید، جمیع توابع با } آغاز و به { پایان می پذیرند و setup و loop نیز تابع هستند.

ترتیب اجرا شدن دستورات

هنگامی که برد آردوینو روشن شود، ابتدا دستوراتی که در داخل setup قرار دارند، به ترتیب اجرا می‌شوند. هنگامی که اجرای دستورات موجود در داخل setup، به پایان رسید، دستورات داخل loop به ترتیب، اجرا می شوند. لغت loop به معنی *حلقه* است و اجرا شدن دستورات داخل loop نیز تکرار

می‌شود. بدین معنی که پس از اینکه آخرین دستور موجود در loop اجرا شد، مجددا اجرای دستورات داخل loop را از ابتدا شروع می کند.

ورودی و یا خروجی بودن پایه های برد

همان طوری که در عکس ۲-۱ که در ابتدای این فصل قرار دارد می بینید، پایه های ورودی و خروجی که در سمت راست برد آردوینو قرار دارند با اعداد ۰ تا ۱۳ نامگذاری شده اند. هر یک از این پایه ها می‌توانند ورودی و یا خروجی باشند. بنابراین هنگامی که می خواهیم از هر یک از این پینها استفاده کنیم می‌بایست که در داخل setup، اعلام کنیم که آیا آن پایه ورودی است و یا خروجی. هنگامی که آردوینو از طریق پایه اطلاعاتی را از محیط اطراف دریافت می کند، اصطلاحا می گوییم که آن پایه، ورودی است. اما اگر برد آردوینو از طریق پایه خود، بخواهد یک وسیله برقی، مثلا موتور و یا لامپ، را روشن و خاموش کند، می گوییم که آن پایه خروجی است. ما در این برنامه می خواهیم با استفاده از پایه ۱۳، یک لامپ را روشن و خاموش کنیم. پس می بایست این پایه را در وضعیت خروجی (OUTPUT) قرار دهیم. با استفاده از دستور pinMode مشخص می کنیم که هر یک از پایه ها ورودی و یا خروجی باشد. در جلوی pinMode دو تا پارامتر ذکر می کنیم که اولین پارامتر، شماره پایه برد آردوینو است و دومین پارامتر اعلام می کند که ورودی (INPUT) و یا خروجی (OUTPUT) باشد. به طور مثال اگر بخواهیم بگوییم که پین ۵، ورودی باشد، دستور زیر را می نویسیم:

```
pinMode(5,INPUT);
```

در دستور فوق توجه دارید که بین پارامترها , می گذاریم.

روشن و خاموش کردن پایه ها

هنگامی که پایه ای به عنوان خروجی مشخص می شود، با استفاده از دستور digitalWrite می توانیم آن پایه را روشن و یا خاموش کنیم. در جلوی digitalWrite دو تا پارامتر ذکر می کنیم. اولین پارامتر، شماره پایه ای است که می خواهیم روشن و یا خاموش شود و دومین پارامتر اعلام می کند که پایه روشن (HIGH) و یا خاموش (LOW) شود.

هنگامی که پایه ای را روشن (HIGH) کنیم ولتاژ آن پایه ۵ ولت می شود و هنگامی که آن را خاموش (LOW) کنیم، ولتاژ آن ۰ ولت می شود. بنابراین با روشن و خاموش کردن پایه ها می توانیم وسایل برقی را روشن و خاموش کنیم.

ایجاد کردن تاخیر

برای اینکه به کامپیوتر بگوییم که مدتی صبر کند، از delay استفاده می کنیم. در جلوی delay، مدت زمانی که مایلیم کامپیوتر صبر کند را بر حسب میلی ثانیه ذکر می کنیم. هر میلی ثانیه، یک هزارم ثانیه است. بنابراین اگر ۱۰۰۰ میلی ثانیه صبر کنیم، به اندازه یک ثانیه صبر کرده ایم. زیرا

0.001 * 1000 = 1

به عبارت دیگر، برای اینکه ثانیه را به میلی ثانیه تبدیل کنیم کافی است که سه تا صفر به جلوی عدد اضافه کنیم. مثلا اگر بخواهیم که به اندازه دو ثانیه صبر کنیم می توانیم از دستور زیر استفاده کنیم:

```
delay(2000);
```

بی تاثیر بودن فاصله (space) و tab در برنامه

در برنامه فوق اگر دقت کنید، همه دستورات موجود در setup و loop را به اندازه یک tab، به جلو آمده ایم و یا در قبل از // فاصله گذاشته ایم. فاصله و tab در عملکرد برنامه تاثیری ندارد. اما برای زیبایی و خوانایی بیشتر برنامه، خوبست که دستوراتی که در داخل } و { قرار می دهیم را به اندازه یک tab به جلو بیاوریم.

وجود فاصله، فقط در زمانی اهمیت پیدا می کند که دو تا لغت مستقل را از هم جدا می کند. به طور مثال، کلمات void و setup دو لغت مستقل از هم هست و اگر بین آنها فاصله نگذاریم، یک لغت می‌شوند. همچنین در کلماتی مثل digitalWrite کل آن یک نام است و نمی توانیم آن را به صورت digital Write بنویسیم.

حساس به بزرگ و کوچک بودن زبان سی

در زبان سی، بزرگ و کوچک بودن حروف اهمیت دارد. مثلا digitalWrite را نمی توانیم به صورت DigitalWrite و یا digitalwrite بنویسیم و اگر این موضوع را در نظر نگیریم، کامپایلر به ما خطا خواهد داد.

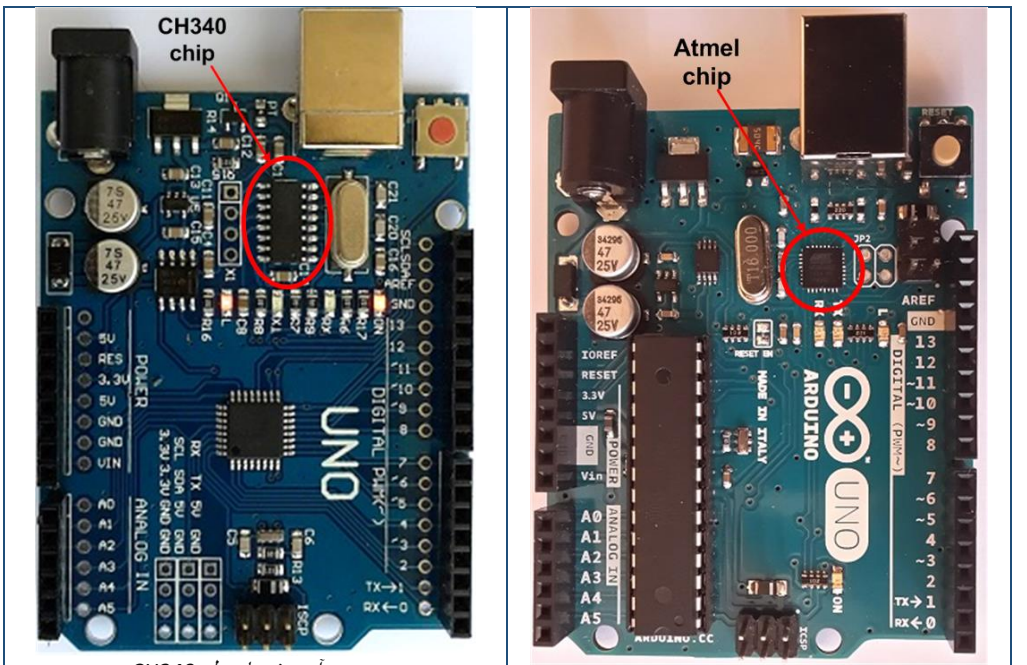
نصب نرم افزار آردوینو و نوشتن اولین برنامه بر روی برد آردوینو

نصب کردن آردوینو

برای نصب کردن محیط برنامه نویسی مربوط به آردوینو فایل arduino-windows.exe را اجرا کنید. اگر این فایل را ندارید، آن را از آدرس زیر دانلود و سپس اجرا کنید:

http://www.arduino.cc/en/Main/Software

همچنین اگر از بردهای آردوینو چینی استفاده می کنید، می بایست درایور مربوط به مبدل یو اس بی به سریال آن را نصب کنید. در عکس ۳-۲، دو تا برد آردوینو نمایش داده شده است. اگر بر روی برد آردوینو شما در پایین سوکت یو اس بی، یک آی سی سیاه رنگ مربعی وجود دارد (شکل الف) نیازی به نصب کردن درایور ندارید. اما اگر در پایین سوکت یو اس بی، یک آی سی سیاه رنگ مستطیلی وجود دارد (شکل ب) می بایست درایور CH341 را دانلود و نصب کنید. به منظور نصب کردن درایور، بر روی گوگل عبارت CH341 driver را جستجو کنید و درایور مربوط به آن را دانلود کنید. سپس فایل setup.exe را از داخل دایرکتوری CH341SER اجرا کنید تا درایور مربوطه نصب شود.



الف. برد آردوینو با مبدل Atmel

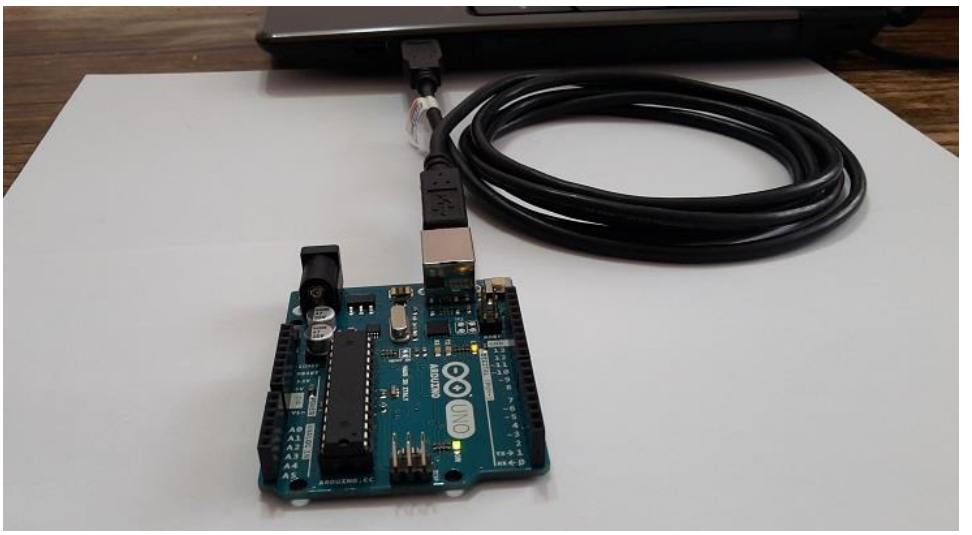
ب. برد آردوینو با مبدل CH340

شکل ۳-۲: عکس دو برد آردوینو اونو با مبدلهای USB به سریال متفاوت

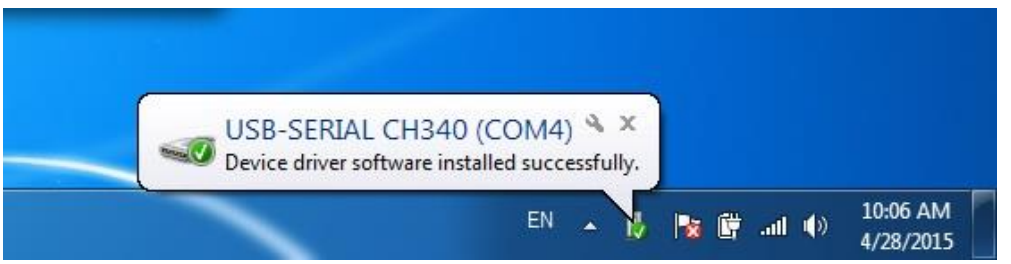
برد آردوینو را از طریق کابل یو اس بی به کامپیوتر خود متصل کنید. در شکل ۳-۳، طریقه اتصال برد آردوینو از طریق کابل یو سی بی، به پورت یو سی بی کامپیوتر، نشان داده شده است.

هنگامی که برای اولین بار، برد آردوینو را به کامپیوتر متصل می کنید، کامپیوتر شروع به نصب درایور آن می کند و در پایین صفحه پیغامی ظاهر می شود و بعد از چند ثانیه، اعلام می کند که درایور با موفقیت

نصب شده است و اسم پورتی که درایور بر روی آن نصب شده است را نیز اعلام می کند. به طور مثال در عکس۳-٤، اعلام کرده است که درایور بر روی COM4 نصب شده است.



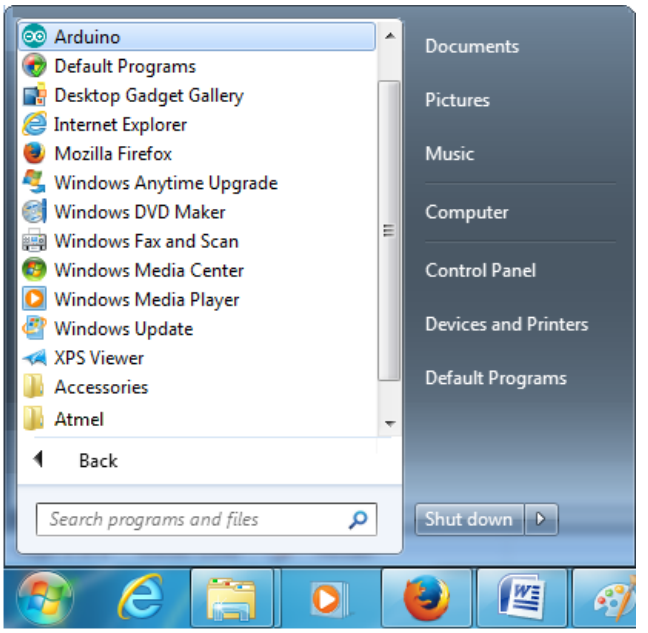
شکل ۳-۳: متصل کردن برد آردوینو از طریق کابل یو اس بی به کامپیوتر



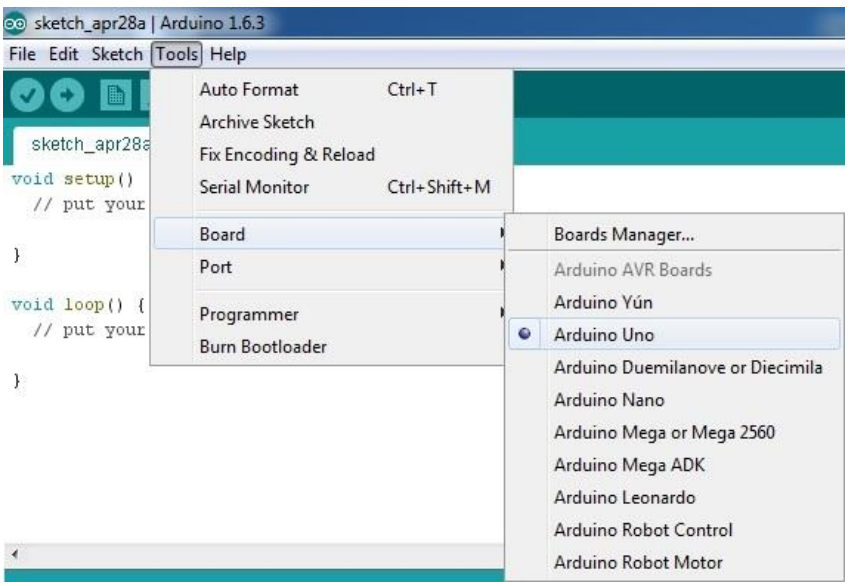
شکل ۳-۴: تشخیص و نصب شدن درایور مربوط به آردوینو

حال به داخل منوی Start بروید و بر روی All Programs کلیک کنید و برنامه Arduino را اجرا کنید. شکل ۳-۵ را ببینید.

در محیط Arduino به سراغ منوی Tools بروید و بر روی Board کلیک کنید و از لیستی که باز می‌شود، مدل برد آردوینو خود را انتخاب کنید. در شکل ۳-٦، Arduino Uno به عنوان مدل برد، انتخاب شده است. مدل برد آموزشی شما، با خط درشت بر روی برد شما چاپ شده است.

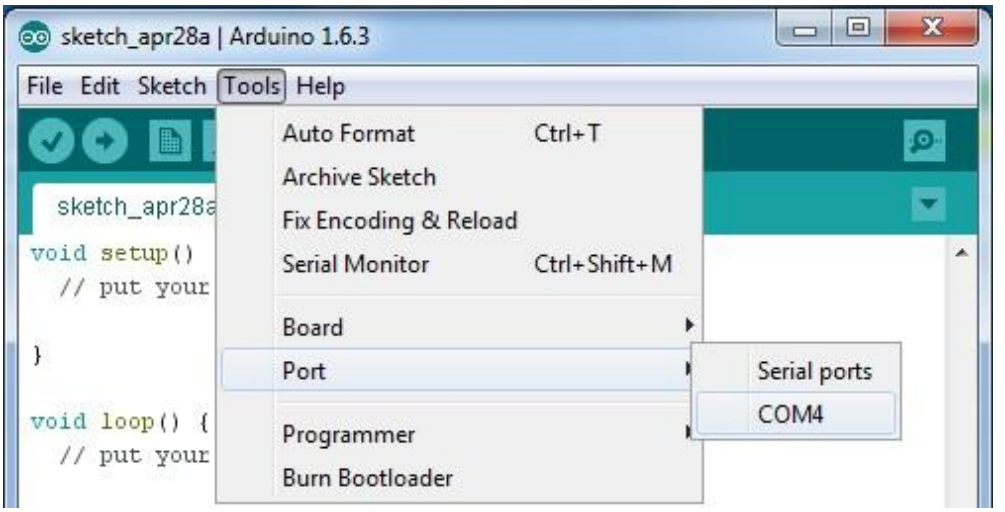


شکل ۳-۵: باز کردن نرم افزار آردوینو (Arduino) از داخل منوی start



شکل ۳- ۶: انتخاب کردن برد آردوینو

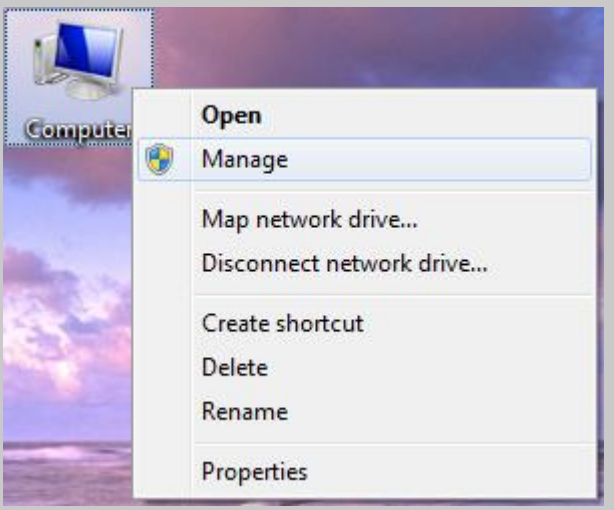
مجددا به منوی Tools بروید و بر روی Port کلیک کنید و پورتی که برد آردوینو به آن متصل است را انتخاب کنید. در شکل ۳-۷، ما COM4 را انتخاب کرده ایم. اما ممکن است در کامپیوتر شما به پورت دیگری متصل باشد.



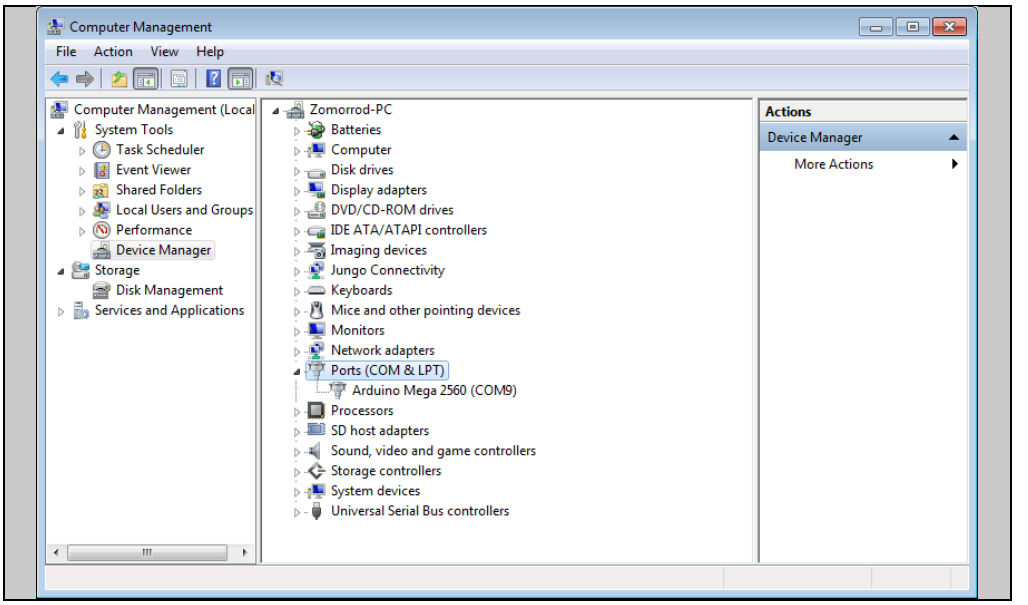
شکل ۳- ۷: انتخاب کردن پورتی که آردوینو به آن متصل شده است

طریقه تشخیص پورتی که آردوینو به آن متصل است

برای اینکه ببینید، برد آردوینو شما به چه پورت سریالی متصل شده است، با موس بر روی Computer، کلید راست کنید و Manage را انتخاب کنید.

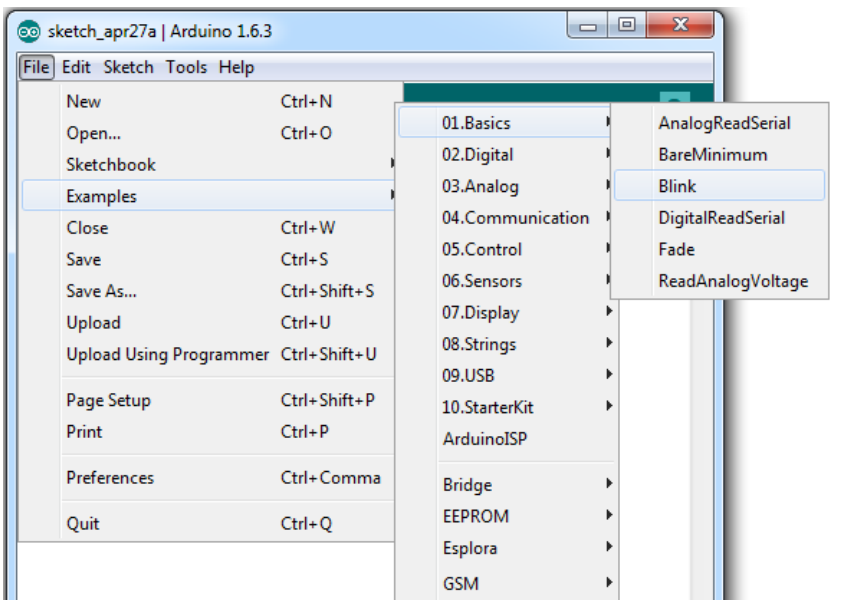


سپس در کادر سمت چپ بر روی Device Manager کلیک کنید و در کادر وسط بر روی Ports (COM & LPT) کلیک کنید. بر زیر این عنوان، اعلام شده است که آردوینو به چه پورتی متصل است. به طور مثال، تصویر زیر نشان می دهد که آردوینو به پورت COM9 متصل است.



باز کردن نمونه برنامه های موجود در محیط آردوینو

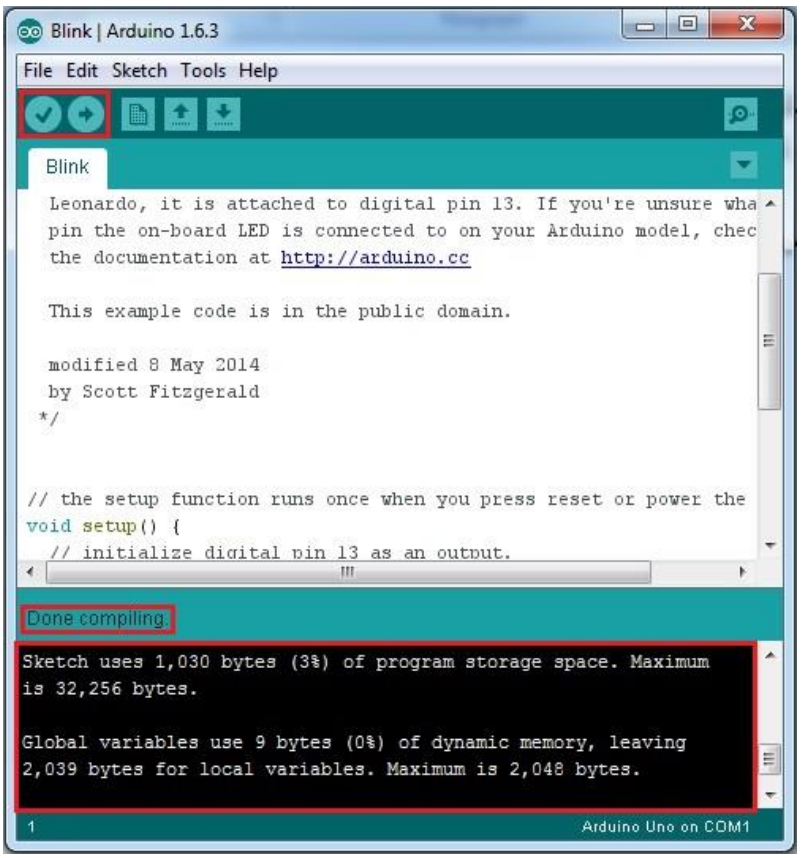
به منوی File مراجعه کنید و بر روی گزینه Example کلیک کنید و از داخل منویی که باز می شود 01.Basics و سپس Blink را انتخاب کنید. در پنجره جدیدی برنامه Blink باز می شود که مشابه همین برنامه چشمک زنی هست که در فوق نوشتیم.



شکل ۳ - ۸: باز کردن مثالهای آماده در محیط آردوینو

کامپایل کردن برنامه

بر روی دکمه  کلیک کنید تا برنامه شما به زبان ماشین ترجمه شود. اصطلاحا به عملِ تبدیل کردن برنامه به زبان ماشین، کامپایل کردن (Compile) گفته می شود. اگر در برنامه، خطایی وجود نداشته باشد، با موفقیت به زبان ماشین تبدیل می شود و پیغام Done Compiling در پایین برنامه ظاهر می شود و در پنجره ای که پایین برنامه است گزارش می دهد که چه مقدار از حافظه آی سی، مصرف شده است.



نوشتن برنامه بر روی برد آردوینو (پروگرام کردن برد آردوینو)

دکمه  را فشار دهید تا برنامه را بر روی برد آردوینو بریزد. اگر پروگرام کردن برد با موفقیت انجام شود، در پایین قسمت برنامه نویسی پیغام Done uploading ظاهر می‌شود و یک چراغ LED که با اسم L بر روی برد نامگذاری شده است و به پایه ۱۳ برد متصل است، شروع به روشن و خاموش شدن می کند.

نوشتن برنامه هایی دیگر با استفاده از پایه های خروجی

در این قسمت به منظور ممارست بیشتر با برنامه نویسی آردوینو چندین برنامه دیگر می نویسیم.

چهار تا چراغ، به پایه های ۱۰ تا ۱۳ متصل کرده ایم. می خواهیم ابتدا، چراغ متصل به پایه ۱۰ روشن شود و بعد از یک ثانیه، چراغ متصل به پایه ۱۱ نیز روشن شود و همین طور به ترتیب چراغهای ۱۲ و ۱۳ نیز روشن شوند. سپس به ترتیب چراغها خاموش شوند. یعنی اول چراغ ۱۳ خاموش شود. بعد از یک ثانیه چراغ ۱۲ خاموش شود و همین طور به ترتیب چراغهای ۱۱ و ۱۰ نیز خاموش شوند.

برنامه ۳ـ۲

```
// رقص نور
//
void setup() {
    pinMode(10, OUTPUT); //پایه 10 خروجی باشد
    pinMode(11, OUTPUT); //پایه 11 خروجی باشد
    pinMode(12, OUTPUT); //پایه 12 خروجی باشد
    pinMode(13, OUTPUT); //پایه 13 خروجی باشد
}

void loop() {
    digitalWrite(10, HIGH); //پایه 10 را روشن کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(11, HIGH); //پایه 11 را روشن کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(12, HIGH); //پایه 12 را روشن کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(13, HIGH); //پایه 13 را روشن کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(13, LOW); //پایه 13 را خاموش کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(12, LOW); //پایه 12 را خاموش کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(11, LOW); //پایه 11 را خاموش کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
    digitalWrite(10, LOW); //پایه 10 را خاموش کن
    delay(1000);     //به اندازه 1 ثانیه صبر کن
}
```

در برنامه فوق، در داخل **setup**، پایه های ۱۰ تا ۱۳ را خروجی می کنیم. سپس در داخل **loop**، ابتدا چراغ متصل به پایه ۱۰ را روشن می کنیم. سپس بعد از یک ثانیه، چراغ پایه ۱۱ را نیز روشن می کنیم. پس از یک ثانیه چراغ پایه ۱۲ را نیز روشن می کنیم و در نهایت، چراغ ۱۳ را روشن می کنیم. پس از یک ثانیه چراغ ۱۳ را خاموش می کنیم. سپس چراغ ۱۲ را نیز خاموش می کنیم. بعد چراغ ۱۱ را خاموش می‌کنیم

و در نهایت، چراغ ۱۰ را خاموش می کنیم. بعد، دستورات موجود در loop از ابتدا آغاز می شوند. یعنی ابتدا پایه ۱۰ روشن می شود و الی آخر.

چراغها و رقص نورهایی که در خیابانها می بینید، برنامه ای مشابه برنامه فوق را دارند و با تغییر دادن ترتیب روشن و خاموش شدن، شکلهای مختلفی از رقص نور را می توانید درست کنید. همچنین می توانید مقدار delay را کم کنید که سریع تر، روشن و خاموش شوند.

در برنامه ۳-۳ می خواهیم یک چراغ راهنمایی بسازیم. به گونه ای که ۱۰ ثانیه، چراغ سبز باشد، ۵ ثانیه زرد باشد و ۱۵ ثانیه قرمز باشد.

ابتدا به زبان خود بیان می کنیم که چه کارهایی لازم است انجام شود.

- چراغ سبز را روشن می کنیم. بعد از ۱۰ ثانیه، چراغ سبز را خاموش و چراغ زرد را روشن می‌کنیم.
- مدت ۵ ثانیه صبر می کنیم. سپس چراغ زرد را خاموش و چراغ قرمز را روشن می کنیم.
- مدت ۱۵ ثانیه صبر می کنیم و سپس چراغ قرمز را خاموش می کنیم و کارهای فوق را مجددا تکرار می کنیم.

برای پیاده سازی کردن برنامه فوق بر روی برد آردوینو، سه تا از پایه های ورودی/خروجی را برای این اتصال به این سه چراغ در نظر می گیریم. مثلا پایه ۱۱ را به عنوان سبز، پایه ۱۲ را به عنوان زرد و پایه ۱۳ را به عنوان قرمز در نظر می گیریم. در قسمت setup از برنامه ابتدا این سه پایه را خروجی می کنیم و الگوریتم فوق را در داخل loop می نویسیم:

برنامه ۳-۳

```
// چراغ راهنمایی
//
void setup() {
       pinMode(11, OUTPUT); //پایه ۱۱ که متصل به چراغ سبز است خروجی باشد
       pinMode(12, OUTPUT); //پایه ۱۲ که متصل به چراغ زرد است خروجی باشد
       pinMode(13, OUTPUT); //پایه ۱۳ که متصل به چراغ قرمز است خروجی باشد
}

void loop() {
       digitalWrite(11,HIGH); //پایه ۱۱ که متصل به چراغ سبز است را روشن کن
       delay(10000);          //به اندازه ۱۰ ثانیه صبر کن
```

```
        digitalWrite(11,LOW); //پایه ۱۱ که متصل به چراغ سبز است را خاموش کن
        digitalWrite(12,HIGH); //پایه ۱۲ که متصل به چراغ زرد است را روشن کن
        delay(5000);          //به اندازه ۵ ثانیه صبر کن
        digitalWrite(12,LOW); //پایه ۱۲ که متصل به چراغ زرد است را خاموش کن
        digitalWrite(13, HIGH); //پایه ۱۳ که متصل به چراغ قرمز است را روشن کن
        delay(15000);         //به اندازه ۱۵ ثانیه صبر کن
        digitalWrite(13, LOW); //پایه ۱۳ که متصل به چراغ قرمز است را خاموش کن
}
```

تمرین

۱. برنامه ای بنویسید که پایه ۱۳، به مدت ۲ ثانیه روشن باشد و سپس به مدت ۲ ثانیه خاموش باشد.

۲. برنامه ای بنویسید که پایه ۱۳ به مدت 0.5 ثانیه روشن باشد و سپس به مدت 0.5 ثانیه خاموش باشد.

۳. برنامه ای بنویسید که پایه ۱۳ به مدت 0.2 ثانیه روشن باشد و سپس به مدت 0.2 ثانیه خاموش باشد. سپس به مدت 0.2 ثانیه روشن باشد و سپس به مدت 2.5 ثانیه خاموش باشد.

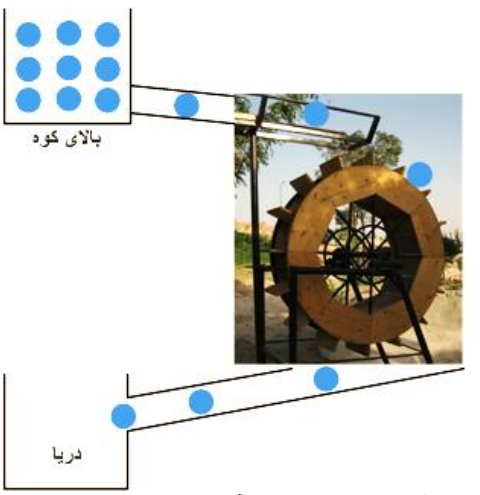
فصل ۴: آشنایی با مدار های الکتریکی

در این فصل با کلیات مدارهای الکتریکی آشنا می شویم و فرا می گیریم که تعدادی LED را به برد آردوینو متصل کنیم و آنها را روشن و خاموش کنیم.

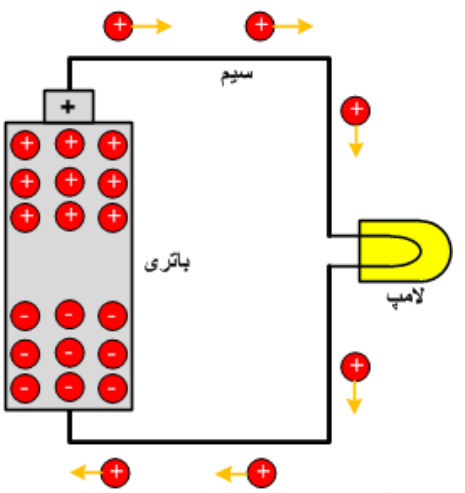
بخش ۴-۱: آشنایی با مفاهیم اولیه الکترونیک

مفهوم اختلاف پتانسیل (ولتاژ)

تعدادی تیله را در نظر بگیرید که بر روی میزی قرار گرفته اند. تیله ها بر سر جای خود ایستاده اند. حال اگر همین تیله ها را بر بالای یک سطح شیبدار قرار دهیم، به سمت پایین شروع به حرکت می کنند. در کل، اشیاء تمایل دارند که از جایی که ارتفاع بیشتری دارد به سمت جایی که ارتفاع کمتری دارد حرکت کنند. اگر آبی بالای کوهی باشد نیز به سمت پایین کوه جریان می یابد و اگر توربین آسیابی در بین راه باشد آن را می چرخاند.



شکل ۴-۱: حرکت قطرات آب از بالای کوه به سمت دریا

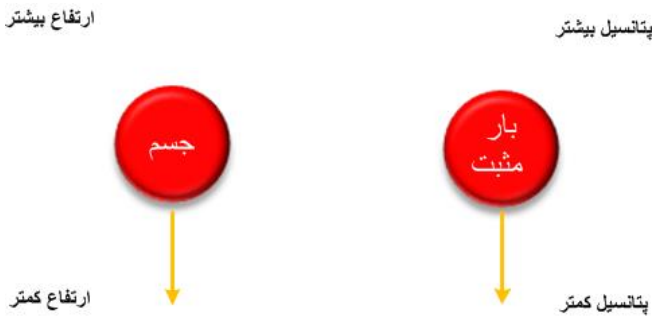


شکل ۴-۲: حرکت بارهای الکتریکی در مدار

در مدار نیز مفهومی به نام ولتاژ (پتانسیل) وجود دارد. همان طوری که اجسام تمایل دارند از محلی که دارای ارتفاع بیشتری است به محلی که ارتفاع کمتری دارد حرکت کنند، بارهای الکتریکی نیز تمایل دارند که از محلی که ولتاژ بیشتری دارند به محلی که ولتاژ کمتری دارد، حرکت کنند و اگر در طی مسیر خود وسیله‌ای الکتریکی وجود داشته باشد، آن را روشن می کنند. شکل ٤-٣ را ببینید.

در باتری ولتاژ سر مثبت باتری، بیشتر از سر منفی آن است و به همین خاطر است که بارهای الکتریکی تمایل دارند که از سر مثبت به سر منفی حرکت کنند. در شکل ٤-٢ یک مدار الکتریکی را می بینید که مشتمل بر یک باتری و یک لامپ و مقداری سیم است. شکل ٤-٢ را با شکل ٤-١ مقایسه کنید. در شکل ٤-

۲، بارهای الکتریکی از سر مثبت باتری به سمت سر منفی باتری حرکت می کنند و در هنگام عبور از لامپ موجب روشن شدن لامپ می شوند. در شکل٤-١ نیز قطرات آب از ارتفاع زیاد به ارتفاع کم می‌روند و هنگام عبور از توربین آسیاب، موجب چرخش آسیاب می شوند.



شکل ۴-۳: تشابه بین حرکت اجسام از ارتفاع بیشتر به کمتر با حرکت بارهای مثبت از ولتاژ (پتانسیل) بیشتر به کمتر

منابع تغذیه

علاوه بر باتری، وسایل دیگری نیز وجود دارند که در دو سر خروجی خود اختلاف ولتاژ ایجاد می‌کنند و از آنها می توانیم برای روشن کردن وسایل برقی استفاده کنیم. اصطلاحا به هر وسیله ای که بتواند برق مورد نیاز وسیله ای برقی را فراهم کند، منبع تغذیه می گوییم. باتری ها، آداپتورها و سلولهای خورشیدی، از جمله منابع تغذیه ای هستند که شما به وفور در اطراف خود می بینید. در زیر عکس چندین منبع تغذیه را می بینید.

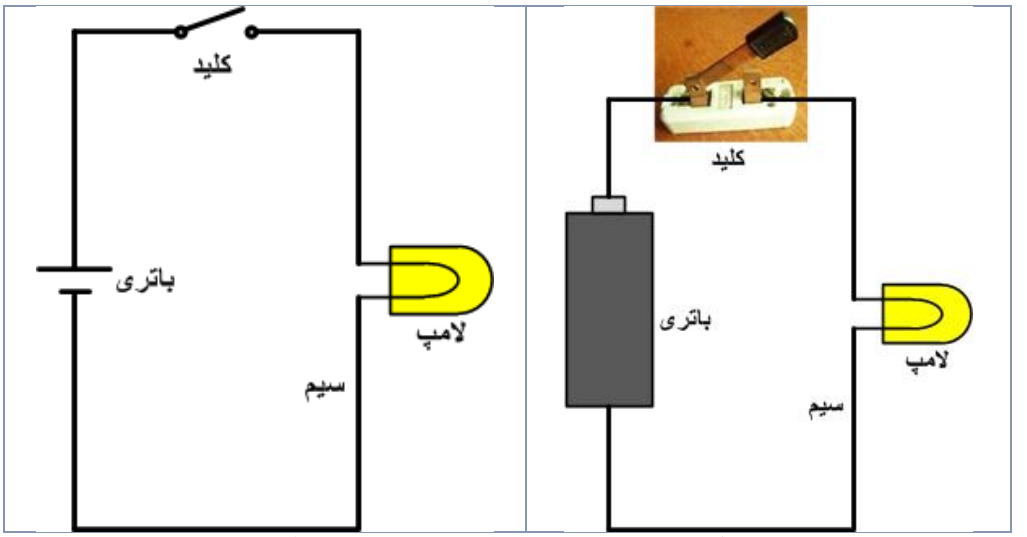
الف. سلول خورشیدی | *ب. آداپتور* | *ج. باتری قلمی*

شکل ۴-۴: نمونه هایی از منابع تغذیه

کلید

می توانیم به مدار شکل ٤-٢، یک کلید اضافه کنیم تا با استفاده از آن، لامپ را روشن و خاموش کنیم. شکل ٤-٥ را ببینید. برای اینکه وسایل الکتریکی روشن شوند، می بایست دو سر آنها به دو سر منبع تغذیه

وصل شوند. برای اینکه وسیله برقی خاموش شود کافی است که یک سر آن از منبع تغذیه جدا شود. با استفاده از کلید می توانیم هر موقع که مایل بودیم یک سر وسیله برقی را از منبع تغذیه جدا کنیم و وسیله برقی را خاموش کنیم.



شکل ۴-۵: عکس مدار یک چراغ و یک باتری به همراه کلید

ساختار کلید

کلید از یک تکه فلز رسانا تشکیل شده است. هنگامی که این فلز بین دو سر کلید قرار گیرد، بارهای الکتریکی از این فلز عبور می کنند و وسایل الکتریکی روشن می شوند. اما زمانی که قطعه رسانا نسبت به یکی از سرهای کلید جدا شود، بارهای الکتریکی را از خود عبور نمی دهد و وسایل برقی خاموش هستند. شکل فوق را ببینید. به قسمت رسانای کلید اصطلاحا پلاتین و یا کنتاکت می گویند.

انواع کلید

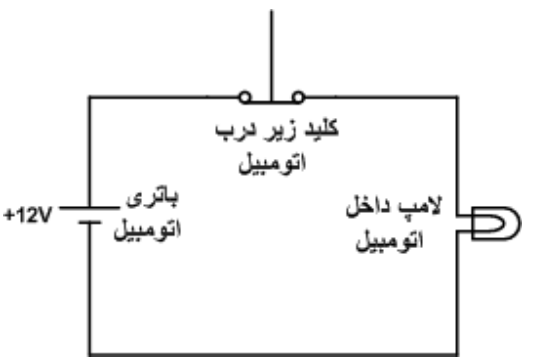
کلیدها را می توان به انواع زیر تقسیم کرد:

- **کلیدهای معمولا باز (Normally Open)**: این کلیدها در حالت عادی قطع هستند. اما هنگامی که به پلاتین آنها فشاری وارد کنیم وصل می شوند. به طور نمونه کلید زنگ منزل و یا کلید چراغ را در نظر بگیرید. پلاتین این کلیدها در حالت عادی قطع هستند. اما هنگامی که کلید را فشار می‌دهیم پلاتین آنها دو سر کلید را به هم وصل می کند و چراغ روشن می شود. برای نشان دادن کلیدهای معمولا باز از سمبلهای زیر استفاده می شود.

شکل ۴-۶: نمادهای رایج برای نشان دادن کلیدهای معمولا باز (Normally Open)

- **کلیدهای معمولا بسته (Normally Close)**: ساختار این کلیدها به گونه ای است که در حالت عادی پلاتین کلید، دو سر خود را به هم وصل کرده است و برای اینکه قطع شود باید فشاری وارد شود. به طور نمونه کلیدهای زیر درب یخچال و درب اتومبیل از این نوع هستند؛ تا مادامی که درب یخچال و یا اتومبیل باز است پلاتین این کلیدها بسته است و چراغ روشن است. هنگامی که درب بسته می شود، این کلیدها فشرده می شوند و پلاتین آنها قطع می شوند و چراغ خاموش می شود. برای نشان دادن کلیدهای معمولا بسته از نمادهای زیر استفاده می کنند. در زیر نقشه مربوط به چراغ اتومبیل را می بینید.

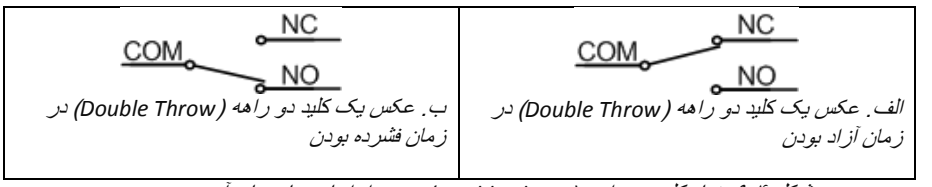
شکل ۴-۷: نمادهای رایج برای نشان دادن کلیدهای معمولا بسته (Normally Close)



شکل ۴-۸: نقشه مدار چراغ داخل اتومبیل

- **کلیدهای دو راهه (Double Throw)**: این کلیدها سه پایه دارند. هنگامی که کلید آزاد است، پلاتین دو تا از سرهای کلید را به هم متصل کرده است. زمانی که کلید فشرده می شود، پلاتین از یکی از سرهای کلید جدا می شود و به سر سوم متصل می شود. برای اینکه بتوانیم پایه‌های این نوع کلید را از هم تشخیص دهیم معمولا روی پایه ها را نامگذاری می کنند. پایه ای که دائما

به پلاتین متصل است را اصطلاحا مشترک (Common) می نامند. پایه ای که در زمان آزاد بودن کلید، پلاتین به آن متصل است را Normally Close (NC) و پایه ای که در زمان فشرده بودن کلید، پلاتین به آن متصل است را Normally Open (NO) می نامند. در زیر عکس یک کلید دو راهه را در لحظه آزاد بودن و فشرده بودن می بینید.



الف. عکس یک کلید دو راهه (Double Throw) در زمان آزاد بودن

ب. عکس یک کلید دو راهه (Double Throw) در زمان فشرده بودن

شکل ۴-۹: نماد کلید دو راهه (Double Throw) به همراه اسامی پایه های آن

انتخاب منبع تغذیه مناسب برای وسایل الکتریکی

برای اینکه وسایل الکتریکی روشن شوند لازم است که آنها را به برق متصل کنیم. هنگامی که می‌خواهیم وسایل برقی را به برق متصل کنیم، می بایست به مشخصات الکتریکی وسیله برقی توجه کنیم. در غیر این صورت ممکن است که وسیله برقی کار نکند و یا آنکه آسیب ببیند. در زیر مشخصاتی از وسایل الکتریکی که می بایست مورد توجه قرار گیرند، ذکر می شوند.

ولتاژ کاری وسیله الکتریکی

هنگامی که وسیله الکتریکی را به برق متصل می کنیم، می بایست ولتاژ وسیله برقی و ولتاژ منبع تغذیه یکسان باشند. اگر ولتاژ منبع تغذیه کمتر از ولتاژ وسیله برقی باشد، وسیله الکتریکی ممکن است روشن نشود و یا ضعیف تر از حالت عادی کار کند. مثلا اگر دو شاخه تلویزیونی که ۲۲۰ ولت است را به باتری ۱۲ ولت متصل کنیم، تلویزیون روشن نمی شود و اگر یک لامپ ۱۲ ولت را به منبع تغذیه ۵ ولت وصل کنیم، با نور کم روشن می شود. همچنین اگر یک موتور ٦ ولت را به منبع تغذیه ۵ ولت وصل کنیم، موتور می چرخد اما سرعت چرخش آن کمتر از زمانی است که موتور را به ٦ ولت متصل کنیم. در عین حال، قدرت موتور نیز کمتر خواهد بود.

اما اگر وسیله الکتریکی را به منبع تغذیه ای متصل کنیم که ولتاژ منبع تغذیه بیشتر از وسیله الکتریکی باشد، به احتمال زیاد وسیله برقی آسیب می بیند. مثلا اگر یک موتور ۵ ولت را به منبع تغذیه ۱۲ ولت متصل کنیم، موتور خواهد سوخت.

اگر به مثال چرخ آسیاب که در ابتدای فصل مطرح شده برگردیم، کم بودن ولتاژ منبع تغذیه مثل این است که آب قدرت کافی برای چرخاندن چرخ آسیاب نداشته باشد و زیاد بودن ولتاژ منبع تغذیه مثل این است که سیلی بیاید که در این صورت مسلما چرخ آسیاب را می شکند.

توجه: *ولتاژ منبع تغذیه می بایست با ولتاژ وسیله برقی برابر باشد.*

جریان مورد نیاز وسیله برقی

هنگامی که وسیله برقی را به منبع تغذیه متصل می کنیم، اگر جریان مورد نیاز وسیله برقی از جریانی که منبع تغذیه می تواند فراهم کند، بیشتر باشد، وسیله برقی ممکن است کار نکند و یا ممکن است به منبع تغذیه آسیب بزند. بنابراین جریانی که منبع تغذیه می تواند فراهم کند، می بایست از جریان مورد نیاز وسیله برقی بیشتر باشد و یا برابر هم باشند. برای اینکه این موضوع را بهتر درک کنید تصور کنید که شما یک توربین کوچک و یا یک فرفره دارید. اگر این فرفره را در جوی آبی که سر کوچه شماست و آب کمی از آن می گذرد قرار دهید، جریان آب باعث چرخش فرفره می شود. حال اگر فرفره را داخل رودخانه ای که آب زیادی از آن می‌گذرد قرار دهید نیز جریان آب موجب چرخش فرفره می شود. اما اگر توربین بزرگ آسیاب را بر روی جوی آب نصب کنیم، تعداد قطرات گذرا از جوی آب برای چرخش توربین آسیاب کافی نخواهد بود و توربین آسیاب نمی چرخد.

جریان با واحد آمپر بیان می شود که علامت آن A است. مثلا اگر روی یک وسیله برقی نوشته شده باشد 2A بدین معنی است که برای کار کردن به ۲ آمپر جریان نیاز دارد. همچنین اگر بر روی یک منبع تغذیه نوشته شده باشد 5A بدین معناست که حداکثر جریانی که می تواند فراهم کند ۵ آمپر است. البته گاهی جریان را بر حسب میلی آمپر نیز بیان می کنند. میلی، به معنای یک هزارم (0.001) است. بنابراین هر میلی آمپر، یک هزارم آمپر است که با نماد mA نمایش داده می شود. همانطور که میلی متر، یک هزارم متر است و برای بیان کردن فواصل کوچک استفاده می شود، از میلی آمپر نیز برای بیان کردن جریانهای کوچک استفاده می شود.

مستقیم یا متناوب بودن

در باتری همیشه یک سر آن مثبت و سر دیگر آن منفی است. اما در برخی از منابع تغذیه و برق شهر، دائما سر مثبت و منفی با هم عوض می شوند. به طور مثال، اگر یک پریز برق شهر را در نظر بگیریم، به مدت ۰/۰۱ ثانیه سوراخ سمت راستی مثبت است و سپس به مدت ۰/۰۱ ثانیه سوراخ سمت چپ مثبت است. درست انگار که یک نفر داشته باشد، سریعا سر مثبت و منفی برق را عوض کند.

هر گاه در یک منبع تغذیه سر مثبت و منفی همیشه ثابت باشد، اصطلاحا آن منبع تغذیه را دی سی (DC) می گویند و در فارسی آن را مستقیم می نامند. هنگامی که سر مثبت و منفی یک منبع تغذیه دائما با هم عوض شوند، آن منبع تغذیه را اِی سی (AC) می گویند و در فارسی آن منبع را متناوب می نامند.

هنگامی که یک وسیله الکتریکی با ولتاژ DC کار می کند، می بایست آن را حتما به منبع تغذیه DC وصل کنیم و هنگامی که وسیله الکتریکی AC است می بایست آن را به منبع تغذیه AC متصل کنیم.

در عکس زیر مشخصات یک آداپتور را می بینید. این آداپتور از طریق دو شاخه به پریز برق شهر متصل می شود و در خروجی خود برق مورد نیاز جهت شارژ شدن یک گوشی موبایل را فراهم می کند. در این عکس، مشخصات ورودی (input) و خروجی (output) این آداپتور مشخص شده است. این عکس نشان می دهد که ورودی آن (دو شاخه آداپتور) می بایست به ولتاژ ۱۰۰ ولت تا ۲۴۰ ولت متصل شود و ولتاژ آن می بایست AC باشد. (برق برخی از کشورها ۱۱۰ ولت است و در برخی کشورهای دیگر ۲۲۰ ولت است. بنابراین بعضی از آداپتورها به گونه ای ساخته می شوند که با برق ۱۱۰ ولت تا ۲۴۰ ولت کار کنند و در جمیع کشورها قابل استفاده باشند.) این آداپتور در خروجی خود ولتاژ ۵ ولت را ایجاد می کند و می‌تواند حداکثر تا ۲ آمپر جریان را فراهم کند.

شکل ۴-۱۰: عکس یک آداپتور ۵ ولت

برای جمع بندی مطالب فوق، مثال زیر را ببینید.

مثال ۴-۱

بر روی یک فن الکتریکی اعلام شده است که ولتاژ کاری آن ۵ ولت DC است و جریان آن 1.5 آمپر است. آیا می توانیم این فن را با آداپتوری که در شکل فوق دیدیم روشن کنیم؟

پاسخ:

ولتاژ کاری فن ۵ ولت است و ولتاژ خروجی آداپتور فوق نیز ۵ ولت است. بنابراین از لحاظه ولتاژ برای هم مناسب هستند.

جریان مورد نیاز این فن 1.5 آمپر است و حداکثر جریانی که آداپتور فوق الذکر می تواند فراهم کند ۲ آمپر است، پس آداپتور میتواند جریان مورد نیاز این فن را فراهم کند.

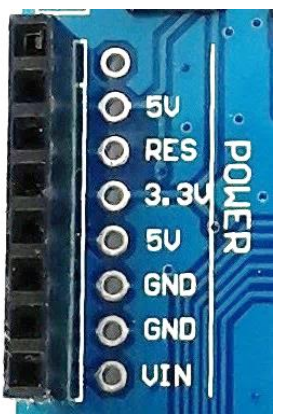
همچنین، هم فن و هم خروجی منبع تغذیه DC است. پس از این لحاظ نیاز مناسب یکدیگرند. بنابراین می توانیم این فن را با این آداپتور روشن کنیم.

بخش ۴-۲: آشنایی با پایه های برد آردوینو و روشن کردن یک ال ای دی (LED) با پایه های آردوینو

در این بخش می خواهیم با پایه هایی که روی برد آردوینو هستند بیشتر آشنا شویم و سپس با استفاده از آردوینو یک چراغ LED را روشن کنیم.

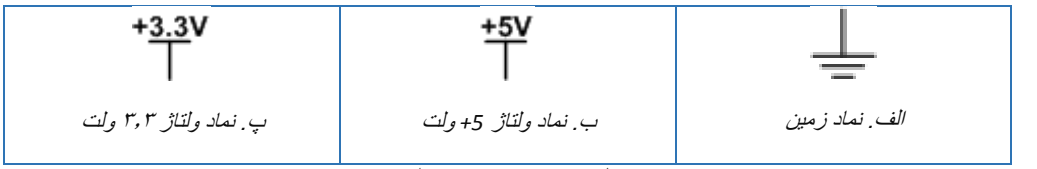
پایه های تغذیه

در سمت بالا سمت چپ برد آردوینو، تعدادی پایه وجود دارند که از آنها می توان به عنوان منبع تغذیه استفاده کرد. شکل ٤-١١ را ببینید. در زیر به توضیح این پایه ها می پردازیم.



شکل ۴-۱۱: پایه های تغذیه در آردوینو

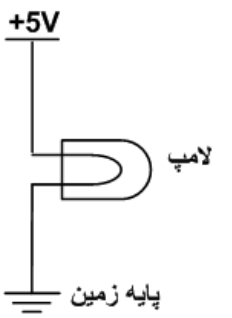
پایه GND: حروف GND مخفف ground (به معنی زمین) هستند. ما ممکن است که ارتفاع وسایل را نسبت به سطح زمین بسنجیم و بگوییم که فلان چیز ۲ متر بالاتر از سطح زمین است و بگوییم که فلان چیز ٤ متر زیر زمین است. همچنین می توانیم ارتفاع وسایلی که بالای سطح زمین هستند را با اعداد مثبت و وسایلی که زیر زمین هستند را با ارتفاع منفی بیان کنیم. مثلا بگوییم ارتفاع فلان چیز ٤- و ارتفاع فلان چیز ۲+ است. در مدارهای الکتریکی نیز، نقطه ای از مدار که ولتاژ آن ۰ ولت است را اصطلاحا زمین می‌گوییم و ولتاژ سایر نقاط را نسبت به زمین می سنجیم. مثلا اگر می گوییم که ولتاژ نقطه ای از مدار ۵+ است یعنی ولتاژ آن نقطه ۵ ولت از ولتاژ زمین بیشتر است.در مدارهای الکتریکی، برای نشان دادن زمین از نماد نشان داده شده در شکل ٤-۱۲-الف استفاده می‌شود. در مدارها، هرگاه در نقطه ای از مدار، نماد زمین وجود داشته باشد، بدین معنی است که آن نقطه با سیم به پایه زمین متصل شده است.



الف. نماد زمین | *ب. نماد ولتاژ 5+ ولت* | *پ. نماد ولتاژ ۳,۳ ولت*

شکل ۴-۱۲: نماد زمین و ولتاژ

پایه 5V: ولتاژ این پایه از ولتاژ پایه GND، ۵ ولت بیشتر است. بنابراین می توانیم از سر ۵+ و GND به عنوان یک منبع تغذیه ۵ ولت استفاده کنیم. در مدارهای الکتریکی برای اینکه نشان دهند که نقطه ای از مدار می بایست به ولتاژ ۵ ولت متصل شود، از نماد ٤-۱۲-ب استفاده می کنند. به طور مثال در مدار زیر، یک سر لامپ را به پایه 5V+ از برد آردوینو و سر دیگر آن را به پایه GND متصل کرده ایم.



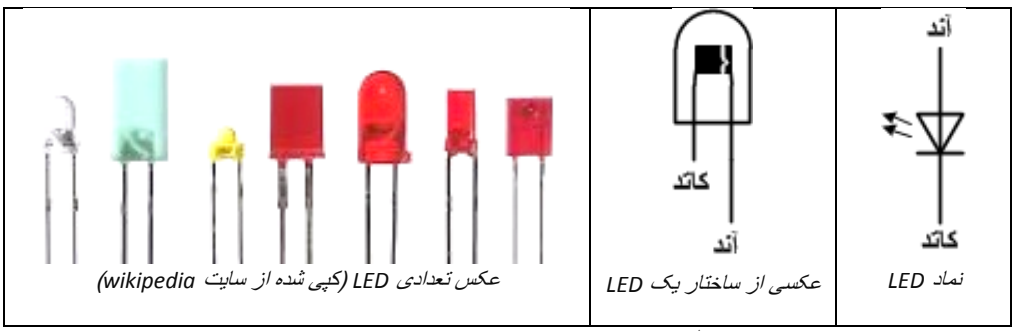
شکل ۴- ۱۳: متصل کردن یک لامپ به ولتاژ ۵ ولت

پایه 3.3V: ولتاژ این پایه نسبت به زمین (پایه GND)، 3.3 ولت است.

آشنایی با LED (Light Emitting Diode)

LED یک قطعه الکتریکی است که دو پایه دارد. در شکل ٤-١٤ عکس تعدادی LED را به همراه نماد الکتریکی آن می بینید.

پایه ای از LED که سیم آن بلندتر است آنُد (Anode) نام دارد و پایه ای که سیم آن کوتاه تر است را کاتُد (Cathode) می گویند. امروزه LEDها به عنوان چراغ و نمایشگر وضعیت در بسیاری از وسایل الکتریکی کاربرد دارند. ولتاژ کاری LEDها به رنگ LED بستگی دارد و بین 1.8 ولت تا 3.3 ولت است. در جدول ٤-١، ولتاژ کاری و جریان برخی از LEDها را می بینید.



نماد LED | عکسی از ساختار یک LED | عکس تعدادی LED (کپی شده از سایت wikipedia)

شکل ۴-۱۴: نماد و ساختار LED و عکس تعدادی LED

رنگ LED	ولتاژ کاری	جریان مورد نیاز
قرمز رنگ	1.8V	10mA
زرد رنگ	2.1V	10mA
سبز رنگ	2.2V	10mA
آبی رنگ	3.3V	20mA

جدول ۴-۱: ولتاژ کاری و جریان مورد نیاز در LEDهای مختلف

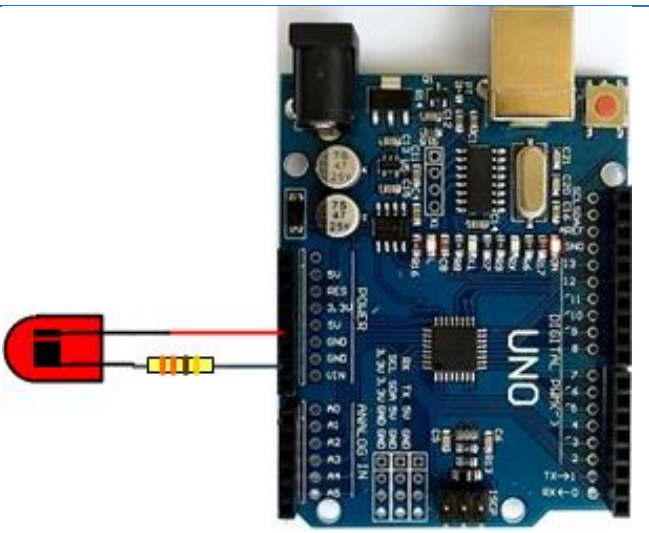
مقدار جریانی که LEDها برای روشن شدن نیاز دارند، در حد چندین میلی آمپر است و بنابراین پایه‌های آردوینو قادرند جریان مورد نیاز جهت روشن شدن LED را فراهم کنند. سر آنُد LED، سر مثبت آن است و سر کاتد آن، سر منفی می باشد. به عبارت دیگر، برای اینکه LED روشن شود، می بایست سر آند را به سر مثبت منبع تغذیه و سر منفی آن را به سر منفی منبع تغذیه متصل کنیم.

روشن کردن LED با استفاده از پایه ۵ ولت روی برد آردوینو

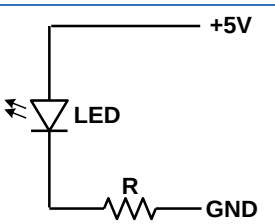
با توجه به اینکه ولتاژ کاری LEDها بین 1.8 ولت تا 3.3 ولت است و ولتاژ پایه های آردوینو ۵ ولت است، اگر بخواهیم LED را به پایه های آردوینو متصل کنیم، یک مقاومت بین پایه LED و پایه آردوینو قرار

می دهیم. در این صورت، ولتاژ مورد نیاز برای روشن شدن LED، بر روی LED می افتد و مابقی ولتاژ بر روی مقاومت می افتد. شکل ٤-١٥ را ببینید.

در جدول شکل ٤-١٥ مقاومت مورد نیاز جهت اتصال به LEDهای مختلف ذکر شده‌اند. البته در صورت در دسترس نبودن مقاومتهای مختلف، می‌توانید برای جمیع LEDهای قرمز و سبز و زرد از مقاومتهای ٢٧٠ یا ٣٣٠ اهم استفاده کنید. توجه داشته باشید که هیچگاه پایه های LED را مستقیما به پایه‌های آردوینو متصل نکنید. زیرا ولتاژ کاری LEDها بین 1.8 تا 3.3 ولت است و ولتاژ خروجی پایه های آردوینو ٥ ولت است و اگر مستقیما LED را به پایه های آردوینو متصل کنید، هم LED و هم پایه های آردوینو آسیب می‌بینند.



الف. عکس اتصال LED و مقاومت به برد آردوینو



ب. شماتیک اتصال LED و مقاومت به پایه +5V و GND

رنگ LED	اندازه مقاومت
قرمز رنگ	330 اهم
زرد رنگ	270 یا 330 اهم
سبز رنگ	270 اهم
آبی رنگ	91 اهم

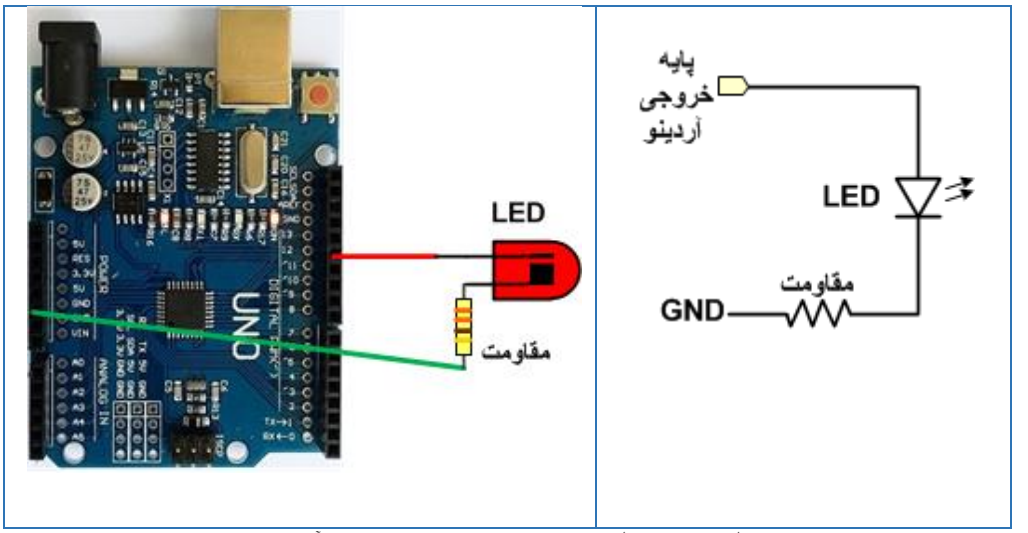
ج. میزان مقاومتهای مناسب به ازای هر LED

شکل ۴-۱۵:اتصال LED و مقاومت به پایه 5+ و GND از برد آردوینو

روشن و خاموش کردن LED با استفاده از پایه های آردوینو

هنگامی که پایه های آردوینو را HIGH کنیم، ولتاژ آن پایه ٥ ولت می شود و هنگامی که پایه را LOW کنیم ولتاژ پایه 0 ولت می‌شود. بنابراین می توانیم LED را به صورت نمایش داده شده در شکل ٤-١٦ به

پایه های ورودی/خروجی برد آردوینو متصل کنیم. در این صورت، زمانی که پایه آردوینو LOW باشد، ولتاژ دو سر LED، صفر ولت است و LED خاموش است. هنگامی که پایه HIGH شود، ولتاژی بر روی LED می افتد و روشن می شود.



شکل ۴-۱۶: اتصال LED و مقاومت به پایه های خروجی آردوینو

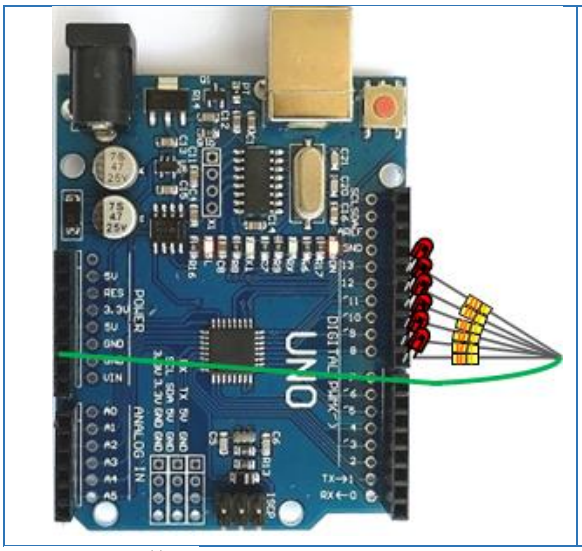
روی برد آردوینو، از قبل به پایه ۱۳، یک LED قرمز به همراه مقاومت متصل شده است و ما در فصلهای قبل آن را روشن و خاموش کرده ایم. اکنون مایلیم که به بقیه پایه های برد آردوینو نیز مقاومت و LED متصل کنیم و رقص نور و سایر وسایل تزیینی و کاربردی بسازیم. در مثالهای زیر نمونه هایی از آن را می‌بینید.

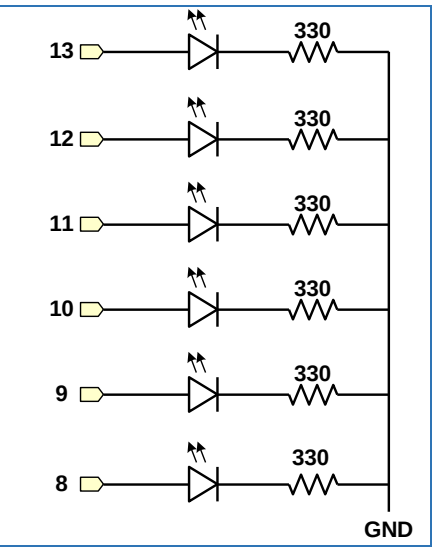
مثال ۴-۲

می خواهیم ۶ تا LED قرمز رنگ به پایه های ۸ تا ۱۳ متصل کنیم و سپس برنامه ای بنویسیم که ابتدا پایه ۸ روشن شود. بعد از ۱ ثانیه پایه ۹ روشن شود و به همین ترتیب بعد از یک ثانیه پایه های ۱۰ و ۱۱ و ۱۲ و ۱۳ نیز روشن شوند. زمانی که همه LED ها روشن شدند، بعد از یک ثانیه LED پایه ۸ خاموش شود. بعد از یک ثانیه LED پایه ۹ خاموش شود و به ترتیب سایر LED ها نیز خاموش شود و زمانی که همه LED ها خاموش شدند، برنامه از ابتدا شروع به روشن کردن LEDها کند.

پاسخ:

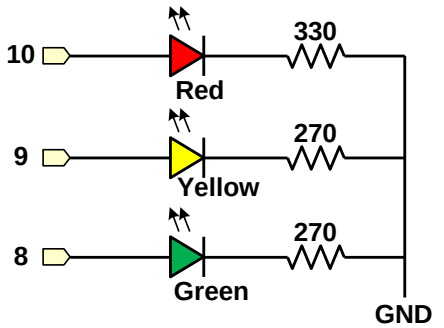
با توجه به اینکه می خواهیم از LED قرمز رنگ استفاده کنیم، از مقاومتهای 330 اهم استفاده می کنیم.
بنابراین مدار LED ها به صورت زیر می شود.






```
void setup() {
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}
void loop() {
  digitalWrite(8,HIGH);
  delay(1000);
  digitalWrite(9,HIGH);
  delay(1000);
  digitalWrite(10,HIGH);
  delay(1000);
  digitalWrite(11,HIGH);
  delay(1000);
  digitalWrite(12,HIGH);
  delay(1000);
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(8,LOW);
  delay(1000);
  digitalWrite(9,LOW);
  delay(1000);
  digitalWrite(10,LOW);
  delay(1000);
  digitalWrite(11,LOW);
  delay(1000);
  digitalWrite(12,LOW);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```

مثال ۴-۳



می خواهیم با استفاده از آردوینو یک چراغ راهنمایی بسازیم. به گونه ای که مدت ۱۰ ثانیه چراغ سبز روشن باشد، ۴ ثانیه زرد روشن باشد و ۱۴ ثانیه قرمز روشن باشد. LEDهای سبز، زرد و قرمز را به ترتیب به پایه-های ۸ و ۹ و ۱۰ از برد آردوینو متصل کنید و برنامه مربوطه را بنویسید.

پاسخ:

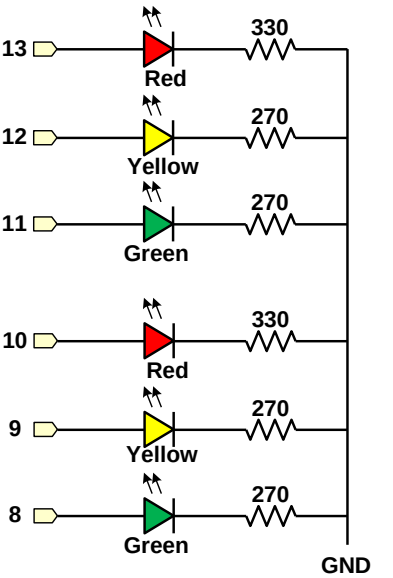
```
void setup() {
    pinMode(8,OUTPUT); //8 as output
    pinMode(9,OUTPUT); //9 as output
    pinMode(10,OUTPUT);//10 as output
}

void loop() {
    digitalWrite(8,HIGH); //Green = on
    delay(10000); // wait 10 seconds
    digitalWrite(8,LOW); //Green = off
    digitalWrite(9,HIGH); //Yellow = on
    delay(4000); // wait 4 seconds
    digitalWrite(9,LOW); //Yellow = off
    digitalWrite(10,HIGH); //Red = on
    delay(14000); // wait 14 seconds
    digitalWrite(10,LOW); //Red = off
}
```

مثال ۴-۴

می خواهیم با استفاده از آردوینو برای یک چهار راه چراغ راهنمایی بسازیم. در این چهار راه، ۲ چراغ راهنمایی داریم که زمانی که چراغ راهنمایی اول سبز یا زرد است چراغ راهنمایی دوم قرمز است و برعکس. با استفاده از ۶ عدد LED این چراغها را پیاده سازی کنید. زمان بندی های چراغها را همانند مثال قبل بگیرید.

پاسخ:

```
void setup() {
    pinMode(8,OUTPUT); //8 as output
    pinMode(9,OUTPUT); //9 as output
    pinMode(10,OUTPUT);//10 as output

    pinMode(11,OUTPUT);//11 as output
    pinMode(12,OUTPUT);//12 as output
    pinMode(13,OUTPUT);//13 as output
}

void loop() {
    digitalWrite(13,HIGH); //Red2 = on
    digitalWrite(8,HIGH); //Green1 = on
    delay(10000); // wait 10 seconds
    digitalWrite(8,LOW); //Green1 = off
    digitalWrite(9,HIGH); //Yellow1 = on
    delay(4000); // wait 4 seconds
    digitalWrite(9,LOW); //Yellow1 = off
    digitalWrite(10,HIGH); //Red1 = on
    digitalWrite(13,LOW); //Red2 = off
    digitalWrite(11,HIGH); //Green2 = on
    delay(10000); // wait 10 seconds
    digitalWrite(11,LOW); //Green2 = off
    digitalWrite(12,HIGH); //Yellow2 = on
    delay(4000); // wait 4 seconds
    digitalWrite(12,LOW); //Yellow2 = off
    digitalWrite(10,LOW); //Red1 = off
}
```



بخش ۴-۳: استفاده کردن از پایه های A0 تا A5

بر روی بردهای آردوینو تعدادی از پایه ها با نام A0، A1 و A5 نامیده شده اند. این پایه ها را می توانیم همانند سایر پایه های ورودی و خروجی مورد استفاده قرار دهیم. به طور مثال با استفاده از دستور زیر می توانیم، پایه A1 را به عنوان خروجی مقداردهی کنیم:

```
pinMode(A1,OUTPUT);
```

همچنین برای خاموش (LOW) کردن پایه A2 می توانیم دستور زیر را بنویسیم:

```
digitalWrite(A2,LOW);
```

در مثال زیر، به طور نمونه برنامه ای نوشته ایم که پایه A5 چشمک می زند.

مثال ۴-۵

برنامه ای بنویسید که پایه A5، هر ثانیه یکبار روشن و خاموش شود.

پاسخ:

```
void setup() {
  pinMode(A5,OUTPUT);
}
void loop() {
  digitalWrite(A5,HIGH);
  delay(1000);
  digitalWrite(A5,LOW);
  delay(1000);
}
```

تمرین

۱. سه تا LED به پایه های ۸ و ۹ و ۱۰ متصل کنید. سپس برنامه ای بنویسید که ابتدا پایه ۸ روشن شود و بعد از ۰,۵ ثانیه پایه ۸ خاموش شود و پایه ۹ روشن شود و پس از ۰,۵ پایه ۹ خاموش شود و پایه ۱۰ روشن شود. سپس بعد از نیمه ثانیه جمیع مراحل فوق از ابتدا اجرا شوند.

۲. می خواهیم ٦ تا LED قرمز رنگ به پایه های ۸ تا ۱۳ متصل کنیم و سپس برنامه ای بنویسیم که ابتدا پایه ۸ روشن شود. بعد از نیم ثانیه پایه ۹ روشن شود و به همین ترتیب بعد از نیم ثانیه پایه های ۱۰ و ۱۱ و ۱۲ و ۱۳ نیز روشن شوند. زمانی که همه LED ها روشن شدند، بعد از نیم ثانیه LED پایه ۱۳ خاموش شود. بعد از نیم ثانیه LED پایه ۱۲ خاموش شود و به ترتیب سایر LED ها نیز خاموش شود و زمانی که همه LED ها خاموش شدند، برنامه از ابتدا شروع به روشن کردن LEDها کند.

۳. اگر دقت کرده باشید، در چراغهای راهنمایی، هنگامی که چراغ یک سمت قرمز می شود، بلافاصله چراغ سمت دیگر سبز نمی شود. بلکه برای اینکه مطمئن باشند که اتومبیلهای قبلی از چهار راه عبور کرده اند و چهار راه خالی شده است، پس از اینکه چراغ یک سمت قرمز شد، مدت یک ثانیه چراغ سمت دیگر نیز قرمز باقی می ماند و سپس چراغ آن سبز می شود. برنامه مثال ٤-٤ را به گونه ای تغییر دهید که زمانی که چراغ یک سمت قرمز می شود، یک ثانیه بعد چراغ سمت دیگر سبز شود.

فصل ۵: ورودی ها، متغیرها و دستورات شرطی

بخش ۵-۱: آشنایی با متغیرها

در فصل یک، دیدیم که یکی از اجزای کامپیوتر، حافظه است که درون آن می تواند اطلاعاتی را نگهداری کند. در زبان های برنامه نویسی می توانیم متغیّر تعریف کنیم، بدین معنی که قسمتهایی از حافظه را نامگذاری کنیم تا در طی برنامه اطلاعاتی را درون آن ذخیره کنیم و بعداً از اطلاعات موجود در آنها استفاده کنیم. ساختار تعریف کردن متغیر در زبان سی به صورت زیر است:

جنس متغیر نام متغیر ;

مثلا دستور زیر، درون حافظه، یک متغیر به نام num برای ذخیره سازی عدد (integer) تعریف می‌کند:

```
int num;
```

همچنین دستور زیر یک متغیز به نام c برای ذخیره کردن حرف (character) تعریف می کند:

```
char c;
```

جنس متغیر، بیان می کند که چه نوع اطلاعاتی درون متغیر ذخیره می شود و فضایی که به متغیر اختصاص داده می شود چه اندازه باشد. در جدول زیر انواع متغیرها و اندازه آنها بر حسب بایت بیان می‌شود:

نوع اطلاعات	اندازه	جنس متغیر
حروف	۱ بایت	**char**
عدد	۱ بایت	**byte**
عدد	۲ بایت	**int**
عدد	۴ بایت	**long**
عدد اعشاری	۴ بایت	**float**
نتیجه گزاره های منطقی	۱ بیت	**bool**

جدول ۵-۱: متغیرها و اندازه فضایی که در حافظه می گیرند

همان طوری که در جدول فوق می بینید، برای ذخیره کردن عدد، جنسهای مختلفی مثل byte و int و long وجود دارد. با توجه به اینکه فضای اختصاص یافته به آنها متفاوت است، اندازه اعدادی که می‌توانند درون خود نگه دارند فرق می کند. مثلا متغیری که از جنس byte تعریف شود، فقط می تواند اعداد ۰ تا ۲۵۵ را در خود جا دهد. در جدول ۵-۲، اعدادی که هر یک از متغیرها می توانند در خود جا دهند را می‌بینید.

جنس متغیر	دامنه اعدادی که می تواند نگهدارد
byte	۰ تا ۲۵۵
int	-۳۲۷۶۸ تا +۳۲۷۶۷
long	-۲۱۴۷۴۸۳۶۴۸ تا ۲۱۴۷۴۸۳۶۴۷

جدول ۵-۲: دامنه اعدادی که در هر متغیر جا می شود

بنابراین هنگامی که دستور زیر را می نویسیم، کامپایلر ۲ بایت از حافظه را برایمان کنار می گذارد. شکل زیر را ببینید:

int num;

با استفاده از علامت = می توانیم مقدار دلخواهی را به داخل متغیرها بریزیم. مثلا دستور زیر، متغیر num را با مقدار ۵، مقداردهی می کند:

num = 5;

همچنین به راحتی می توانیم بر روی متغیرها عملیات ریاضی چهار گانه از قبیل جمع و تفریق و ضرب و تقسیم را انجام دهیم. جدول زیر عملیات ریاضی را به همراه مثال نشان می دهد

عمل ریاضی	علامت	مثال	نتیجه اجرا شدن مثال
جمع	+	a = 2 + 5;	عدد ۵ را با ۲ جمع می زند و حاصل جواب که ۷ است را در a می ریزد.
تفریق	-	a= b – 5;	5 را از مقدار متغیر b کم می کند و حاصل جواب را در متغیرa می ریزد.
ضرب	*	n = 2 * 9;	۲ و ۹ را در هم ضرب می کند و حاصل جواب که ۱۸ است را درون n می ریزد
تقسیم	/	n = 9 / 2;	۹ را بر ۲ تقسیم می کند و خارج قسمت تقسیم که ۴ است را درون متغیر n می ریزد.
باقیمانده تقسیم	%	n = 9 % 2;	هنگامی که ۹ را بر ۲ تقسیم کنیم، باقی مانده تقسیم ۱ می شود که مقدار ۱ را به درون n می ریزد.

جدول ۵-۳: عملوندهای ریاضی و انجام عملیات ریاضی

به طور مثال، برای اینکه مقدار ax+2 را هنگامی که x برابر ٤ و a برابر ٣ است بخواهیم به دست آوریم و به درون y بریزیم، میتوانیم برنامه زیر را بنویسیم

```
int x = 4;
int a = 3;
int y;
y = a * x + 2;
```

توضیح اینکه، ما می توانیم در زمانی که متغیر را تعریف می کنیم مقداری را به آن اختصاص دهیم و یا اختصاص ندهیم. به عنوان نمونه، در مثال فوق، ما به x و a در زمان تعریف کردنشان مقداری را اختصاص داده ایم. اما به y مقداری را اختصاص نداده ایم، بلکه بعداً به درون آن مقداری را ریخته ایم. اگر مایل بودیم، به متغیر y نیز می توانستیم در موقع تعریف کردن، مقداری را اختصاص بدهیم.

مثال ۵-۱

برنامه ای بنویسید که متغیرهای a و b و c را از جنس int تعریف کند و آنها را به ترتیب با مقادیر ۶ و ۷ و ۸ مقدار دهی کند. سپس حاصل عبارت b*b – 4*a*c را حساب کند و در متغیری از جنس int به نام delta بریزد.

پاسخ:

```
int a = 6;
int b = 7;
int c = 5;
int delta = b*b - 4*a*c;
```

قوانین نامگذاری متغیرها در زبان سی

برای نامگذاری کردن متغیرها فقط میتوانیم از حروف انگلیسی و اعداد استفاده کنیم. همچنین اولین حرف اسم باید عدد نباشد. مثلا نمی توانیم اسم متغیری را 3th بگذاریم. اما گذاشتن اسامی مثل num3 و یا a123 و یا Arc32 مجاز است، چرا که در اسامی آنها فقط از حروف و عدد استفاده شده است و اولین حرف اسم نیز، عدد نیست.

متغیرهای unsigned

کلمه unsigned در انگلیسی به معنی بدون علامت هست. همانطوری که در جدول زیر می بینید متغیرهایی که از جنس unsigned تعریف شوند، فقط اعداد صفر و مثبت را در خود نگه می دارند و در عوض، اعداد مثبت بزرگتری را در مقایسه با متغیرهای معمولی جای می دهند. به طور مثال، متغیر int

حداکثر تا عدد 32767+ را میتواند در خود جای دهد، در حالی که unsigned int قادر است که تا اعدادی به بزرگی ٦٥٥٣٥ را نیز در خود جای دهد.

جنس متغیر	اندازه	دامنه اعدادی که می تواند نگهدارد
unsigned int	۲ بایت	از ۰ تا ۶۵۵۳۵
int	۲ بایت	از ۳۲۷۶۸- تا ۳۲۷۶۷+
unsigned long	۴ بایت	از ۰ تا ۴۲۹۴۹۶۷۲۹۵
long	۴ بایت	از ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷+

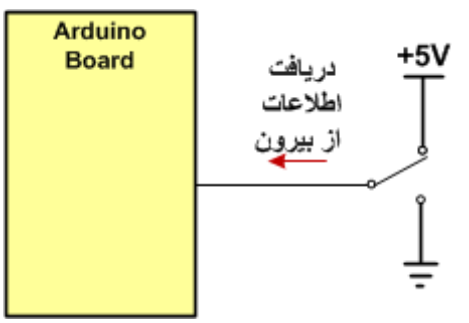
جدول ۵-۴ : متغیرهای unsigned در مقایسه با متغیرهای علامتدار (signed)

بخش ۵-۲ : خواندن وضعیت ورودیها

در فصل قبل دیدیم که با استفاده از *pinMode* می توانیم مشخص کنیم که هر یک از پایه های برد آردوینو، ورودی و یا خروجی باشد. به طور مثال، دستور زیر سبب می شود که پایه ٦ ورودی شود:

```
pinMode(6, INPUT);
```

هنگامی که پایه ای در وضعیت ورودی قرار داشته باشد با استفاده از دستور digitalRead می توانیم اطلاعاتی را از بیرون دریافت کنیم. به عبارت دیگر، به پایه ای که ورودی شده است می توانیم وسیله ای مثل کلید و یا آی سی و یا سنسور و یا مداری الکتریکی متصل کنیم که آن وسیله، پایه برد آردوینو را به ۱ و یا ۰ متصل کند و ما با استفاده از دستور digitalRead چک کنیم که آیا پایه به ۱ متصل شده است (HIGH شده است) و یا به ۰ متصل شده است (LOW شده است)



شکل ۵-۱

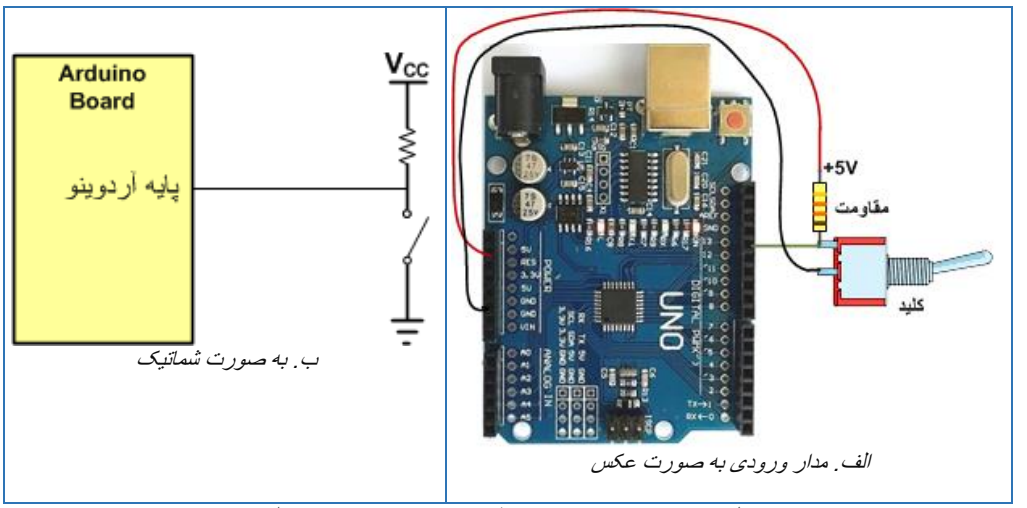
در جلوی دستور digitalRead، شماره پایه ای که می خواهیم وضعیت آن چک شود را ذکر می کنیم. مثلا دستور digitalRead(9) وضعیت پایه ۹ را می خواند. همچنین دستور زیر، وضعیت پایه ٦ را می‌خواند و به داخل متغیر a می ریزد:

```
a = digitalRead(6);
```

مقاومت بالا کش (Pull-up)

در این قسمت می خواهیم طریقه اعمال کردن ۰ و ۱ منطقی (۰ ولت و ۵ ولت) به پایه های آردوینو را فرا بگیریم.

در نگاه اول به نظر می رسد که اعمال کردن ۰ ولت و ۵ ولت به پایه های آردوینو کار خیلی ساده ای باشد؛ کافیست که پایه آردوینو را از طریق سیم به ۰ ولت یا ۵ ولت متصل کنیم تا ۰ و ۱ منطقی به پایه آردوینو اعمال شود. اما نکته ای که وجود دارد زمانی است که سیم آزاد است. زمانی که سیم را از ۵ ولت جدا می کنیم تا به ۰ ولت متصل کنیم یا زمانی که سیم را از ۰ ولت جدا می کنیم تا به ۵ ولت متصل کنیم، در این زمان پایه آردوینو آزاد است و در این موقع آردوینو که مقدار پایه ورودی را می خواند، مقدار پایه را ممکن است که هر مقداری بخواند. (در عمل وقتی که پایه ای آزاد باشد مقدار آن دائما بین ۰ و ۱ منطقی تغییر می کند.) بنابراین در مواقعی که می خواهند ۰ و ۱ منطقی را به پایه ای اعمال کنند مرسوم است که مداری به شکل زیر ببندند.



شکل ۵-۲: ایجاد کردن ۰ و ۱ منطقی با استفاده از مقاومت و کلید

در مدار فوق، هنگامی که کلید بسته باشد، پایه آردوینو مستقیما به زمین متصل می شود و پایه آردوینو ۰ منطقی (low) می شود. همچنین هنگامی که کلید باز باشد، پایه آردوینو از طریق مقاومت به ولتاژ ۵ ولت متصل می شود و پایه آردوینو ۱ منطقی (high) را دریافت می کند.

مدار فوق ساختار ساده ای دارد، اما اگر قرار باشد هر موقع که می خواهیم به پایه های آردوینو ورودی اعمال کنیم، مدار فوق را ببندیم کار وقتگیری خواهد بود. خوشبختانه پایه های آردوینو مجهز به مقاومت

بالاکش داخلی هستند. هرگاه بخواهیم علاوه بر قرار دادن پایه آردوینو در وضعیت ورودی، مقاومت بالاکش را نیز فعال کنیم، کافی است که دستور زیر را بنویسیم:

```
pinMode(شماره پایه,INPUT_PULLUP);
```

به طور مثال برای اینکه پایه ٤ را در وضعیت ورودی قرار دهیم و مقاومت بالاکش آن را فعال کنیم می بایست دستور زیر را بنویسیم:

```
pinMode(4,INPUT_PULLUP);
```

هنگامی که مقاومت بالاکش پایه آردوینو را فعال کنیم، پایه آردوینو از طریق یک مقاومت به ولتاژ ۵ ولت متصل می شود. بنابراین اگر پایه آردوینو آزاد باشد، آن پایه از طریق مقاومت بالاکش داخلی به ۵ ولت وصل می شود و ۱ منطقی (high) خوانده می شود. همچنین اگر پایه را به ۵ ولت متصل کنیم، همان ۱ منطقی خوانده می شود و هنگامی که پایه را از طریق سیم به زمین متصل کنیم، ولتاژ پایه ۰ ولت می‌شود و ۰ منطقی (low) خوانده می شود.

بنابراین هرگاه بخواهیم از پایه آردوینو به عنوان ورودی دیجیتال استفاده کنیم، بهتر است که آن را به صورت INPUT_PULLUP مقداردهی اولیه کنیم. در این حالت هر گاه بخواهیم به پایه ۱ منطقی را اعمال کنیم، آن را به حال خود رها می کنیم و هر گاه بخواهیم ۰ منطقی را اعمال کنیم، پایه مربوطه را از طریق سیم به زمین متصل می کنیم.

بخش ۵-۳: دستورات شرطی

در قسمت قبل دیدیم که چگونه می توانیم وضعیت پایه ای را بخوانیم و ببینیم که پایه ای آیا روشن (HIGH) و یا خاموش (LOW) است. اما صرف دانستن، هیچ فایده ای ندارد. بلکه باید دانشی که داریم را به کار ببندیم. ما معمولا بر حسب وضعیت ورودیها تصمیم می گیریم که برنامه چه کاری را انجام دهد.

به طور مثال، ممکن است کلیدی را به پایه ٦ از برد آردوینو و چراغی را به پایه ۱۳ از برد آردوینو متصل کنیم و بخواهیم هر موقع که کلید فشرده شده بود، چراغ هر یک ثانیه یکبار روشن و خاموش شود. اما اگر کلید، رها بود، چراغ خاموش بماند.

نخست بیایید الگوریم کاری که انتظار داریم برنامه انجام دهد را به زبان خود بیان کنیم.

در گام نخست می بایست که پایه ٦ را ورودی و پایه ۱۳ را خروجی کنیم.

سپس وضعیت پایه ٦ را می‌خوانیم. اگر در وضعیت LOW بود، پایه ١٣ را روشن و خاموش می‌کنیم (پایه ١٣ را روشن می کنیم و یک ثانیه صبر می کنیم. سپس پایه ١٣ را خاموش می کنیم و یک ثانیه صبر می کنیم) اما اگر در وضعیت HIGH بود، کاری انجام نمی دهیم.

ما در الگوریتم فوق گفتیم "اگر در وضعیت LOW بود"، فلان شود و گر نه فلان شود. به عبارت دیگر، روشن و خاموش شدن چراغ را مشروط به LOW بودن کلید کردیم. لغت معادل اگر در زبان انگلیسی if می باشد و در زبان سی برای اینکه شرطی را بنویسیم از ساختار زیر پیروی می کنیم:

```
if(فلان شرط برقرار بود)
{
    فلان کار را بکن
}
```

در دستور فوق، اگر شرط ذکر شده برقرار بود، دستورات داخل { و } اجرا می شوند. اما اگر شرط برقرار نبود، دستورات داخل { و } اجرا نمی شوند.

دستور if در زبان برنامه نویسی خیلی کاربرد دارد. مثلا ممکن است بگوییم که "اگر دمای اتاق بالای ٢٧ درجه بود کولر را روشن کن" یا "اگر کاربر دکمه ٢ را فشار داد، فلان کار را بکن." برای بیان کردن شرط ها معمولا از مقایسه استفاده می کنیم. در زیر نمونه هایی از مقایسه به همراه معادل زبان سی آنها، نشان داده شده اند. در این مثالها، از متغیرهای a و b استفاده کرده ایم. اما در عمل بجای آنها می توانیم از هر متغیر و یا عددی استفاده کنیم:

مثال	معادل سی
اگر a از b بزرگتر بود	if(a > b)
اگر a کوچکتر از b بود	if(a < b)
اگر a بزرگتر یا مساوی با b بود	if(a >= b)
اگر a کوچکتر یا مساوی با b بود	if(a <= b)
اگر a برابر با b بود	if(a == b)
اگر a با b مساوی نبود	if(a != b)

جدول ٥-٥: مقایسه کردن مقادیر دو متغیر

به طور مثال برای اینکه بگوییم "اگر پایه ٦، پایین (LOW) بود، چشمک بزن" می توانیم دستورات زیر را بنویسیم

```
if(digitalRead(6) == LOW)
{
    digitalWrite(13, HIGH);
    delay(1000);
```

```
    digitalWrite(13, LOW);
    delay(1000);
}
```

بنابراین متن برنامه ای که موقع فشرده بودن کلید، پایه ۱۳ چشمک بزند، به صورت زیر است:

مثال ۵-۲: برنامه ای که موقع پایین بودن پایه ۶، چراغ متصل به پایه ۱۳ روشن و خاموش شود

```
void setup() {
    pinMode(6, INPUT_PULLUP); //پایه ۶ را ورودی کن
    pinMode(13, OUTPUT); //پایه ۱۳ را خروجی کن
}

void loop() {
    int input = digitalRead(6);
    if(input == LOW)
    {
        digitalWrite(13, HIGH); //پایه ۱۳ را روشن کن
        delay(1000); //یک ثانیه صبر کن
        digitalWrite(13, LOW); //پایه ۱۳ را خاموش کن
        delay(1000);
    }
}
```

در برنامه فوق، در داخل setup، پایه ٦ را ورودی و پایه ۱۳ را خروجی کرده ایم و در داخل loop، وضعیت کلید متصل به پایه ٦ را چک می کنیم و اگر پایین (LOW) بود، پایه ۱۳ چشمک می زند. اما اگر کلید پایین (LOW) نبود کاری صورت نمی گیرد.

توجه داشته باشید که برای مقایسه کردن از علامت == و برای ریختن مقداری درون متغیرها از علامت = استفاده می کنیم. افرادی که برنامه نویسی به زبان سی را شروع می کنند، گاهی اوقات اشتباهاً این علامتها را به جای هم استفاده می کنند و در نتیجه، برنامه به درستی کار نمی کند.

به عنوان مثالی دیگر برای دستور if، تصور کنید که متغیر temp حاوی دمای اتاق باشد. اگر بخواهیم بگوییم که "هر موقع دما بیشتر از ۲۷ درجه بود، کولر روشن شود" می توانیم دستورات زیر را بنویسیم:

```
if(temp > 27) //اگر دما بیشتر از 27 درجه بود
{
    digitalWrite(13, HIGH); //پایه 13 که متصل به کولر است را روشن کن
}
```

در مثال فوق، اگر دما بیشتر از ۲۷ درجه بود، دستور روشن کردن پایه ۱۳، اجرا می شود. اما اگر دما بیشتر از ۲۷ درجه نبود، دستور روشن کردن پایه ۱۳، اجرا نمی شود.

در پاره ای اوقات، ما مایلیم که اگر شرطی برقرار بود، فلان اتفاق بیافتد و اگر شرط برقرار نبود، اتفاق دیگری بیافتد. مثلا در همین مثال آخر، ممکن است مایلیم باشیم که "اگر دما بیشتر از ۲۷ درجه بود، کولر روشن شود و در غیر این صورت، کولر خاموش شود." در زبان سی else معادل لغت "در غیر این صورت" است. ساختار دستور if و else به صورت زیر است

```
if(شرط مورد نظر) //اگر دما بیشتر از 27 درجه بود
{
    فلان کار را بکن //این دستور اگر شرط برقرار باشد اجرا می شود
}
else
{
    فلان کار را بکن //این دستور اگر شرط برقرار نباشد اجرا می شود
}
```

مثلا برای اینکه بگوییم که "اگر دما بیشتر از ۲۷ درجه بود، کولر روشن شود و در غیر این صورت، کولر خاموش شود" دستورات زیر را می نویسیم:

```
if(temp > 27) //اگر دما بیشتر از 27 درجه بود
{
    digitalWrite(13, HIGH); //پایه 13 که متصل به کولر است را روشن کن
}
else
{
    digitalWrite(13, LOW); //پایه 13 که متصل به کولر است را خاموش کن
}
```

مثال ۵-۳: برابری

۲ تا کلید به پایه های ۶ و ۷ متصل کرده ایم. برنامه ای بنویسید که اگر دو کلید در وضعیت یکسانی بودند (هر دو فشرده بودند یا هر دو رها بودند) چراغی که به پایه ۱۳ متصل است روشن باشد و در غیر این صورت، چراغ خاموش باشد.

پاسخ:

در این برنامه، در قسمت setup، پایه های ۶ و ۷ را ورودی و پایه ۱۳ را خروجی می کنیم. سپس در قسمت loop، وضعیت پایه های ۶ و ۷ را می خوانیم. اگر وضعیت یکسانی داشتند، چراغ را روشن می کنیم و در غیر این صورت، چراغ را خاموش می کنیم.

```
void setup() {
    pinMode(6, INPUT_PULLUP); //پایه 6 را ورودی کن
    pinMode(7, INPUT_PULLUP); //پایه 7 را ورودی کن
    pinMode(13, OUTPUT);    //پایه 13 را خروجی کن
}

void loop() {
  int a = digitalRead(6);   //وضعیت پایه 6 را بخوان
  int b = digitalRead(7);   //وضعیت پایه 7 را بخوان
  if(a == b)     //اگر وضعیت پایه های 6 و 7 یکسان بود
  {
      digitalWrite(13, HIGH);      //پایه 13 را روشن کن
  }
  else
  {
      digitalWrite(13, LOW);       //پایه 13 را خاموش کن
  }
}
```

بخش ۵-۴: بیان کردن "یا" و "و"

تصور کنید که به پایه های ۵ و ٦ کلیدهایی متصل کرده باشیم. می خواهیم هر موقع که یکی از این کلیدها پایین بود، چراغی روشن شود. به عبارت دیگر، اگر کلید پایه ٦ پایین بود یا کلید پایه ۷ پایین بود چراغ روشن شود. در زبان سی، برای اینکه "یا" را بیان کنیم از علامت || استفاده می کنیم (کلید شیفت را پایین نگه دارید و کلید \ را دو بار بزنید). مثلاً در دستورات زیر، اگر a برابر با LOW بود یا b برابر با LOW بود، پایه ۱۳ را HIGH می کند:

```
  if((a == LOW)||(b == LOW))
  {
      digitalWrite(13, HIGH);      //پایه 13 را روشن کن
  }
```

در دستورات فوق، الزامی به گذاشتن پرانتز در دو طرف شرطها، وجود ندارد و می توانیم به صورت زیر نیز شرطها را بنویسیم اما برای خوانایی بیشتر توصیه می شود که حتما پرانتز بگذارید و مانند فوق بنویسید:

```
  if(a == LOW||b == LOW)
  {
      digitalWrite(13, HIGH);      //پایه 13 را روشن کن
  }
```

بنابراین شکل کامل برنامه ای که "اگر کلید متصل به پایه ٦ یا ٧ پایین بود، چراغ پایه ١٣ را روشن کن" به صورت زیر می شود:

مثال ۵-۴: یا (OR)

```
void setup() {
    pinMode(6,INPUT_PULLUP);
    pinMode(7,INPUT_PULLUP);
    pinMode(13,OUTPUT);
}

void loop() {               // the following code runs repeatedly:
    int input1 = digitalRead(6);  //وضعیت پایه ۶ را بخوان
    int input2 = digitalRead(7);  //وضعیت پایه ۷ را بخوان

    if((input1 == LOW)||(input2 == LOW)) //اگر پایه ۶ یا ۷ پایین بود
    {
        digitalWrite(13,HIGH);   //پایه ۱۳ را روشن کن
    }
    else
    {
        digitalWrite(13,LOW);    //پایه ۱۳ را خاموش کن
    }
}
```

طریقه تایپ کردن علامت |

علامت | در بالای دکمه \ قرار دارد. بنابراین برای اینکه علامت | را تایپ کنید، دکمه Shift را پایین نگه دارید و سپس دکمه \ را فشار دهید.

در زیر، به منظور ممارست بیشتر شما با دستورات شرطی و علامت ||، مثالهای بیشتری ذکر می‌شود.

مثال ۵-۵

برنامه ای بنویسید که اگر a برابر با ۲ بود یا b کوچکتر از ۵ بود، پایه ۱۳ را روشن کند.

پاسخ:

```
    if((a == 2)||(b < 5))
    {
        digitalWrite(13,HIGH);
    }
```

مثال ۵-۶

برنامه ای بنویسید که اگر a برابر با ۵ یا ۱۰ بود در متغیر b مقدار یک را بریزد.

پاسخ:

```
if((a == 5)||(a == 10))
{
    b = 1;
}
```

مثال ۵-۷

برنامه ای بنویسید که اگر a بر ۳ بخش پذیر بود، یکی مقدار b را بیشتر کند.

پاسخ:

برای اینکه a بر ۳ بخش پذیر باشد، می بایست باقی مانده تقسیم a بر ۳ برابر با صفر شود.

```
if(a % 3 == 0)
{
    b = b + 1;      //مقدار متغیر بی را با ۱ جمع کن و حاصل را در بی بریز
}
```

همچنین ممکن است مایل باشیم که هر گاه دو شرط، برقرار بود اتفاقی بیافتد. مثلاً دو تا کلید به پایه های ٦ و ۷ متصل می کنیم و مایلیم که هر موقع هر دو تا کلید بالا بودند، چراغ روشن شود. در زبان سی، برای اینکه کلمه "و" را بیان کنیم از علامت && استفاده می کنیم:

```
if((فلان شرط برقرار بود)&&(فلان شرط برقرار بود))
{
 فلان کار را بکن
}
```

به طور نمونه، مثالهای زیر را ببینید.

مثال ۵-۸

برنامه ای بنویسید که اگر متغیر a برابر ۵ و b برابر ۷ بود، متغیر c را برابر ۹ قرار دهد.

پاسخ:

```
    if((a == 5)&&(b == 7))
    {
        c = 9;
    }
```

مثال ۵-۹: دزدگیر ساده

می خواهیم یک دزدگیر ساده بسازیم، به گونه ای که اگر کلید دزدگیر فعال بود و درب منزل باز شد آژیر به صدا در بیاید.

پاسخ:

برای این کار، به زیر درب منزل کلیدی وصل می کنیم که موقعی که درب بسته بود کلید فشرده باشد و هر موقع درب باز شد، کلید رها شود. کلید فعال کننده دزدگیر را به پایه ۴ برد آردوینو و کلید زیر درب را به پایه ۵ از برد آردوینو متصل می کنیم. اگر دزدگیر فعال بود و درب باز بود، پایه ۱۲ را روشن می کنیم.

```
void setup() {
    pinMode(4,INPUT_PULLUP); //پایه متصل به کلید فعال کننده را ورودی کن
    pinMode(5,INPUT_PULLUP); //پایه متصل به کلید زیر درب را ورودی کن
    pinMode(12,OUTPUT); //پایه متصل به آژیر را خروجی کن
}

void loop() {
    int alarm = digitalRead(4);
    int door = digitalRead(5);

    if((alarm == LOW)&&(door == HIGH)) //اگر دزدگیر فعال بود و در باز شد
    {
        digitalWrite(12,HIGH); //آژیر به صدا درآید
    }
    else
    {
        digitalWrite(12,LOW); //آژیر خاموش باشد
    }
}
```

مثال ۵-۱۰: دزدگیر

می خواهیم دزدگیری که در مثال قبل ساختیم، را گسترش دهیم به گونه ای که اگر پنجره اتاق پذیرایی باز شد نیز آژیر بزند.

پاسخ:

برای این کار، به زیر پنجره اتاق پذیرایی نیز کلیدی وصل می کنیم که موقعی که پنجره بسته بود کلید فشرده باشد و هر موقع پنجره باز شد، کلید رها شود. کلید زیر پنجره را به پایه ۶ از برد آردوینو متصل می کنیم. اگر موقعی که دزدگیر فعال بود، درب باز شود یا پنجره باز شود، پایه ۱۲ را روشن می کنیم.

```
void setup() {
    پایه متصل به کلید فعال کننده را ورودی کن//
    pinMode(4,INPUT_PULLUP);
    pinMode(5,INPUT_PULLUP); //پایه متصل به کلید زیر درب را ورودی کن
    پایه متصل به کلید زیر پنجره را ورودی کن//
    pinMode(6,INPUT_PULLUP);
    pinMode(12,OUTPUT); //پایه متصل به آژیر را خروجی کن
}

void loop() {
    int alarm = digitalRead(4);
    int door = digitalRead(5);
    int window = digitalRead(6);

    if((alarm == LOW)&&((door == HIGH)||(window == HIGH)))
    اگر دزدگیر فعال بود و در یا پنجره باز شد//
    {
        digitalWrite(12,HIGH); //آژیر به صدا درآید
    }
    else
    {
        digitalWrite(12,LOW); //آژیر خاموش باشد
    }
}
```

مثال ۵-۱۱: آبیاری خودکار

فرض کنید که از قبل درون متغیر h، زمان ساعت وجود دارد. برنامه ای بنویسید که از ساعت ۸ تا ساعت ۱۱، خروجی ۱۳ روشن شود تا آبیاری باغچه منزل شما انجام گیرد.

پاسخ:

برای اینکه ساعت بین ۸ تا ۱۱ باشد، باید مقدار ساعت از ۸ بیشتر باشد و از ۱۱ کمتر باشد. بنابراین برنامه به صورت زیر می‌شود:

```
void setup() {
    pinMode(13,OUTPUT); //پایه متصل به شیر آب را خروجی کن
}

void loop() {
    if((h >= 8)&&(h < 11))
    {
        digitalWrite(13,HIGH); //آبیاری انجام شود
```

```
    }
    else
    {
        digitalWrite(13,LOW); //آبیاری خاموش باشد
    }
}
```

تمرین

۱. هر یک از عبارات ریاضی زیر را با استفاده از دستورات برنامه نویسی، بازنویسی کنید.

الف) $3a+2$ ب) $(2a-3b)/c$ پ) $3-(2a\div b)$

ت) $3-2\times b$ ث) $a^2\times b$

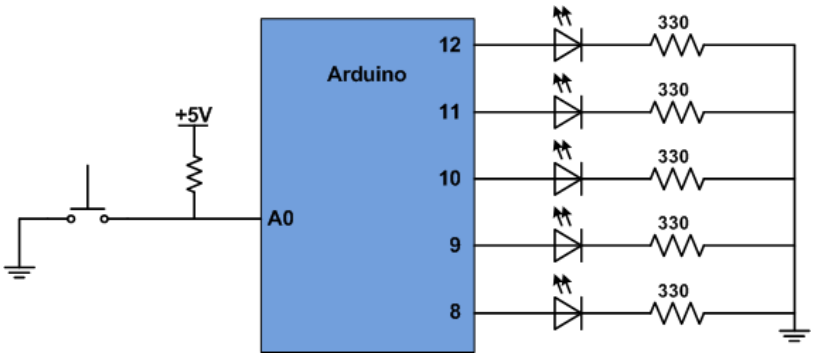
۲. مجموعه دستوراتی بنویسید که اگر مقدار متغیر a بیشتر از b بود، مقدار متغیر b را ۳ تا زیاد کند.

۳. برنامه ای بنویسید که اگر پایه ٤، پایین بود، پایه ۱۲، روشن شود و پایه ۱۳ خاموش شود. اما اگر پایه ٤ بالا بود، پایه ۱۲ خاموش و پایه ۱۳ روشن شود.

٤. برنامه ای بنویسید که اگر پایه ٤ یا پایه ٥ پایین بود، پایه ۱۳ را روشن کند. اما اگر هر دو پایه بالا بود، پایه ۱۳ را خاموش کند.

٥. برنامه ای بنویسید که قدرمطلق مقدار a را درون متغیر t بریزد. یعنی اگر مقدار a بزرگتر مساوی با صفر بود مقدار a را درون متغیر t بریزد و اگر مقدار a کوچکتر از صفر بود، مقدار $-1\times a$ را درون متغیر t بریزد.

٦. می خواهیم با استفاده از ٥ تا چراغ، برای اتومبیل خود چراغ ترمز بسازیم. بدین صورت که هرگاه پدال ترمز فشار داده شد، ابتدا یکی از چراغها روشن شود. سپس بعد از یک ثانیه ۲ تا چراغ دیگر نیز روشن شوند و بعد از ۲ ثانیه همه چراغها روشن شوند. برای این کار ٥ تا LED به ٥ تا از پایه های برد آردوینو متصل کنید و یک کلید نیز به پایه A0 متصل کنید. (شکل زیر) سپس برنامه ای بنویسید که وضعیت پایه A0 را چک می کند و هر گاه کلید متصل به پایه A0 فشرده شد، به ترتیبی که ذکر شد LEDها را روشن کند و بعد از اینکه چراغها روشن شدند، وضعیت پایه A0 را چک کند و هر گاه کلید رها شد، چراغها خاموش شوند.


Arduino
12
11
10
9
8
A0
+5V
330
330
330
330
330

فصل ۶: حلقه های تکرار

بخش ۶-۱: آشنایی با while

فرض کنید بخواهیم برنامه ای بنویسیم که سه بار چراغ متصل به پایه ۱۳ روشن و خاموش شود و سپس به مدت ۱۰ ثانیه خاموش باشد و مجدداً این اتفاق تکرار شود. با استفاده از آنچه که تاکنون فرا گرفته‌ایم، برنامه را به صورت زیر می نویسیم:

```
void setup() {
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);

    delay(10000);
}
```

برنامه فوق کاملا درست کار می کند. اما برای اینکه سه بار پایه ۱۳ چشمک بزند، مجبور شده ایم که دستورات روشن و خاموش شدن را سه بار تکرار کنیم. حال اگر بخواهیم ۱۰ بار چشمک بزند لازم است که ۱۰ بار این مجموعه کد را تایپ کنیم که البته روش بهینه ای نیست. بلکه در چنین مواردی ما از حلقه while استفاده کنیم. با استفاده از حلقه while می توانیم بگوییم که تا مادامی که شرایطی برقرار است کار مشخصی تکرار شود. ساختار دستور while به صورت زیر است:

```
while(شرطی که مایلیم چک شود)
{
    کاری که مایلیم تکرار شود
}
```

تا مادامی که شرط ذکر شده در جلوی while برقرار باشد، دستوراتی که بین } و { نوشته ایم، اجرا می‌شوند. به طور نمونه، در برنامه زیر، تا مادامی که n کوچکتر از 3 باشد، b را یکی کم می کند و n را یکی زیاد می کند:

```
int n = 0;

while(n < 3)
{
    b = b * 2;
    n = n + 1;
}
```

در زیر به مراحل اجرا شدن برنامه فوق می پردازیم:

- در برنامه فوق، ابتدا متغیر n را با ۰ مقداردهی می کنیم.
- سپس شرط دستور while را چک می کند یعنی آیا n کوچکتر از ۳ هست یا نه. با توجه به اینکه n برابر ۰ است، پس n کوچکتر از ۳ است و شرط برقرار است. بنابراین دستورات داخل while اجرا می شوند. یعنی متغیر b در ۲ ضرب میشود و همچنین مقدار n یکی زیاد می‌شود. پس مقدار متغیر n، ۱ می شود.
- پس از اتمام اجرا شدن دستورات داخل while، مجدداً شرط جلوی while چک می شود. با توجه به اینکه مقدار n برابر با ۱ است، پس n کوچکتر از ۳ است و شرط درست است. بنابراین دستورات داخل while مجددا اجرا می شوند. یعنی متغیر b در ۲ ضرب می شود و به مقدار n یکی افزوده می شود. پس مقدار n، ۲ می شود.
- مجددا شرط جلوی while چک می شود. با توجه به اینکه n برابر ۲ است و ۲ کوچکتر از ۳ است، پس شرط جلوی while درست است و مجددا دستورات داخل while اجرا می شوند. بنابراین متغیر b در ۲ ضرب می شود و به مقدار n یکی افزوده می‌شود. در نتیجه مقدار n، ۳ می‌شود.
- مجدداً شرط جلوی while چک می شود. با توجه به اینکه n برابر با ۳ است، پس n کوچکتر از ۳ نیست و شرط جلوی while درست نیست. پس دستورات داخل while اجرا نمی شود.

همان طوری که می بینید، در برنامه فوق دستورات داخل حلقه while سه مرتبه تکرار شده اند و اگر مایلیم باشیم با تغییر دادن شرط جلوی while، باعث شویم که دستورات داخل حلقه while به هر تعداد دفعاتی که مایلیم اجرا شوند.

در زیر با استفاده از while، برنامه ای نوشته ایم که ۱۰ بار پایه ۱۳ روشن و خاموش شود.

مثال ۶-۱: چشمک زدن در حلقه تکرار

```
void setup() {
    pinMode(13, OUTPUT);
}

void loop() {
    int count = 0;

    while(count < 10)
    {
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);
        count = count + 1;
    }

    delay(10000);
}
```

در برنامه فوق، ابتدا متغیر count را برابر صفر قرار می دهیم و از آن برای شمارش تعداد دفعاتی که چشمک زده ایم استفاده می‌کنیم.

هنگامی که برای اولین بار، به حلقه while می رسیم، مقدار count، صفر است. پس مقدار count کوچکتر از ۱۰ است و وارد حلقه while می شود و چشمک می زند و متغیر count نیز ۱ می شود. پس از اتمام حلقه while، مجدداً شرط count < 10 چک می شود و با توجه به اینکه count برابر ۱ است. پس شرط درست است و مجدداً وارد حلقه می شویم و دستورات داخل while اجرا می شود. این سناریوی فوق دائما تکرار می شود تا اینکه ۱۰ بار چشمک بزنیم و مقدار count به ۱۰ برسد. هنگامی که مقدار count به ۱۰ برسد، شرط count < 10 درست نخواهد بود و وارد حلقه while نخواهد شد. بلکه دستوری که بعد از حلقه while قرار گرفته است، یعنی delay، اجرا خواهد شد.

در زیر مثالهای دیگری با استفاده از while می بینید.

مثال ۶-۲: چشمک زن

برنامه ای بنویسید که ۴ بار پایه ۱۳ چشمک بزند، سپس ۶ بار پایه ۱۲ چشمک بزند.

پاسخ:

```
void setup() {
    pinMode(12, OUTPUT);
    pinMode(13, OUTPUT);
}

void loop() {
    int count;
```

```
    //=============
    //13 چشمک زدن پایه
    //=============
    count = 0;

    while(count < 4)
    {
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);

        count = count + 1;
    }

    //=============
    //12 چشمک زدن پایه
    //=============
    count = 0;

    while(count < 6)
    {
        digitalWrite(12, HIGH);
        delay(1000);
        digitalWrite(12, LOW);
        delay(1000);

        count = count + 1;
    }
}
```

مثال ۶-۳: مجموع اعداد ۰ تا ۲۰

با استفاده از while برنامه ای بنویسید که حاصل جمع اعداد بین ۰ تا ۲۰ را محاسبه کند.

پاسخ:

```
    int sum = 0;  //حاوی مقدار حاصل جمع اعداد خواهد بود
    int n = 0;  //حاوی اعداد 0 تا 20 خواهد شد

    while(n <= 20)
    {
        sum = sum + n; //به حاصل جمع قبلی، مقدار جدید را اضافه کن
        n = n + 1;  //مقدار آن را یکی زیاد کن
    }
```

در برنامه فوق، ابتدا n برابر با ۰ است. پس هنگامی که به while می رسیم، شرط n <= 20 برقرار است و وارد حلقه می شویم. درون حلقه، مقدار n را که 0 است به sum اضافه می کنیم. پس sum برابر 0 می‌شود. سپس به مقدار n، یکی می افزاییم و بنابراین مقدار n برابر با ۱ می شود. سپس شرط جلوی while مجددا چک می شود و چون عدد ۱ از ۲۰ کوچکتر است پس مجدداً وارد حلقه میشود و دستورات داخل حلقه اجرا می شود. آخرین باری که وارد حلقه می شود، زمانی است که مقدار n برابر ۲۰ است. در این دفعه، عدد ۲۰ به مقدار متغیر sum افزوده می شود و مقدار n، یکی زیاد می شود و برابر ۲۱ می شود. این بار که شرط جلوی while چک می شود، با توجه به اینکه ۲۱ بزرگتر از ۲۰ است، پس شرط جلوی while درست نیست و وارد حلقه نمی شود. پس همان طوری که دیدیم، این برنامه، مجموع اعداد بین ۰ تا ۲۰ را محاسبه می کند.

مثال ۶-۴: مجموع اعداد فرد بین ۱ تا ۱۹

با استفاده از while برنامه ای بنویسید که حاصل جمع اعداد بین ۱ تا ۱۹ را محاسبه کند.

پاسخ:

```
int sum = 0; //حاوی مقدار حاصل جمع اعداد خواهد بود
int n = 1;  //حاوی اعداد 1 تا 19 خواهد شد

while(n <= 19)
{
    sum = sum + n;
    n = n + 2;  //مقدار آن را دو تا زیاد کن
}
```

در برنامه فوق، ابتدا n برابر با ۱ است. پس هنگامی که به while می رسیم، شرط n <= 19 برقرار است و وارد حلقه می شویم. درون حلقه، مقدار n را که ۱ است به sum اضافه می کنیم. پس sum برابر ۱ می‌شود. سپس به n، ۲ تا می افزاییم و مقدار داخل متغیر n برابر ۳ می شود. سپس شرط جلوی while مجددا چک می شود و چون عدد ۳ از ۱۹ کوچکتر است پس مجدداً وارد حلقه میشود و دستورات داخل حلقه اجرا می شود. آخرین باری که وارد حلقه می شود، زمانی است که مقدار n برابر ۱۹ است. در این دفعه، عدد ۱۹ به مقدار متغیر sum افزوده می شود و مقدار n، دو تا زیاد می شود و برابر ۲۱ می شود. این بار که شرط جلوی while چک می شود، با توجه به اینکه ۲۱ بزرگتر از ۱۹ است، پس شرط جلوی while درست نیست و وارد حلقه نمی شود. پس همان طوری که دیدیم، این برنامه، مجموع اعداد فرد بین ۱ تا ۱۹ را محاسبه می‌کند.

مثال ۶-۵: رقص نور

به پایه های ۱۰ تا ۱۳، چراغهایی متصل کرده ایم. میخواهیم ابتدا چراغ متصل به پایه ۱۰ روشن شود. بعد از یک ثانیه پایه ۱۱ نیز روشن شود و همین طور به ترتیب سایر چراغها روشن شوند. هنگامی که همه چراغها روشن شدند، چراغ ۱۰ خاموش شود، بعد از یک ثانیه چراغ ۱۱ نیز خاموش شود و به همین ترتیب سایر چراغها نیز خاموش شود.

پاسخ:

مشابه این برنامه را در فصل ۳ نوشته ایم. اما این بار با استفاده از while می نویسیم.

```
void setup() {
    int num = 10;
    while(num <= 13)
    {
        pinMode(num, OUTPUT);
        num = num + 1;
    }
}

void loop() {
    int num;

    num = 10;
    while(num <= 13)
    {
        digitalWrite(num, HIGH);
        delay(1000);
        num = num + 1;
    }

    num = 10;
    while(num <= 13)
    {
        digitalWrite(num, LOW);
        delay(1000);
        num = num + 1;
    }
}
```

مثال ۶-۶: رقص نور

برنامه مثال قبل را به گونه ای تغییر دهید که موقع خاموش شدن ابتدا پایه ۱۳ خاموش شود و سپس ۱۲ خاموش شود و به همین ترتیب تا چراغ ۱۰ خاموش شود.

پاسخ:

```
void setup()
{
    int num = 10;
    while(num <= 13)
    {
        pinMode(num, OUTPUT);
        num = num + 1;
    }
```

```
}

void loop()
{
    int num;

    num = 10;
    while(num <= 13)
    {
        digitalWrite(num, HIGH);
        delay(1000);
        num = num + 1;
    }

    num = 13;
    while(num >= 10)
    {
        digitalWrite(num, LOW);
        delay(1000);
        num = num - 1;
    }
}
```

مثال ۶-۷

۴ تا کلید، به پایه های ۴ تا ۷ متصل کرده ایم. برنامه ای بنویسید که بشمارد که چند تا از این کلیدها فشرده شده اند. سپس به تعداد کلیدهایی که فشرده هستند، پایه ۱۳ چشمک بزند و سپس ۵ ثانیه صبر کند و مجدداً تعداد کلیدهای فشرده شده را بشمارد و مراحل فوق تکرار شود.

پاسخ:

```
void setup()
{
    int num = 4;
    while(num <= 7)
    {
        pinMode(num, INPUT);
        num = num + 1;
    }

    pinMode(13, OUTPUT);
}

void loop()
{
    int num;
    int count;

    count = 0;
    num = 4;
```

```
    while(num <= 7)
    {
        int input = digitalRead(num);
        if(input == LOW)
        {
            count = count + 1;
        }
        num = num + 1;
    }

    int i = 0;

    while(i < count)
    {
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);
        i = i + 1;
    }
    delay(5000);
}
```

مقایسه دستور while **با دستور** if

دستور **while** همانند **if**، شرط جلوی خود را چک می کند و اگر درست بود، دستورات داخل حلقه **while** اجرا می شود. اما تفاوتشان این است که پس از اینکه دستورات بین } و { انجام شد، در دستور **while** مجدداً شرط جلوی **while** چک می شود و تا مادامی که شرط برقرار بود دستورات داخل حلقه **while** اجرا می شوند. در حالی که در **if**، فقط یک بار شرط چک می شود و هیچ گونه تکراری در کار نیست.

بخش ۶-۲: دستور for

دستور **for** در حقیقت، شکل فشرده شده ای از دستور **while** است. در مواردی که از **while** برای تکرار استفاده می کنیم:

۱. ابتدا متغیری را با عددی مقداردهی اولیه می کنیم.
۲. سپس با **while** شرطی را چک می کنیم و
۳. در انتهای **while**، مقدار متغیری را زیاد می کنیم.

با استفاده از دستور for می توانیم این سه قسمت را به صورت فشرده، به شکل زیر بنویسیم

```
for(عمل ریاضی بر روی متغیر ;شرط ;مقداردهی اولیه)
{
}
```

به طور مثال، برنامه زیر حاصل جمع اعداد ۰ تا ۲۰ را با استفاده از for محاسبه می کند.

مثال ۶-۸: مجموع اعداد ۰ تا ۲۰

با استفاده از for برنامه ای بنویسید که حاصل جمع اعداد بین ۰ تا ۲۰ را محاسبه کند.

پاسخ:

```
int sum = 0;  //حاوی مقدار حاصل جمع اعداد خواهد بود
int n;  //حاوی اعداد 0 تا 20 خواهد شد

for(n = 0; n <= 20; n = n + 1)
{
    sum = sum + n; //به حاصل جمع قبلی، مقدار جدید را اضافه کن
}
```

در زیر نمونه های دیگری از استفاده از دستور for را می بینید.

مثال ۶-۹: مجموع اعداد 12 تا 32

با استفاده از for برنامه ای بنویسید که حاصل جمع اعداد بین 12 تا 32 را محاسبه کند.

پاسخ:

```
int sum = 0;  //حاوی مقدار حاصل جمع اعداد خواهد بود
int n;  //حاوی اعداد 12 تا 32 خواهد شد

for(n = 12; n <= 32; n = n + 1)
{
    sum = sum + n; //به حاصل جمع قبلی، مقدار جدید را اضافه کن
}
```

مثال ۶-۱۰: مجموع اعداد زوج بین ۰ تا ۲۰

با استفاده از for برنامه ای بنویسید که حاصل جمع اعداد زوج بین ۰ تا ۲۰ را محاسبه کند.

پاسخ:

```
int sum = 0;  //حاوی مقدار حاصل جمع اعداد خواهد بود
int n;  //حاوی اعداد 0 تا 20 خواهد شد

for(n = 0; n <= 20; n = n + 2)
{
    sum = sum + n; //به حاصل جمع قبلی، مقدار جدید را اضافه کن
}
```

تمرین

۱. با استفاده از دستور while برنامه ای بنویسید که ۵ مرتبه پایه ۱۳، را هر ۰,۵ ثانیه روشن و خاموش کند و سپس به مدت ۵ ثانیه خاموش باشد.

۲. برنامه مثال ٦-۱ را به گونه ای تغییر دهید که وضعیت پایه ٦ را چک کند. اگر این پایه پایین بود، پایه ۱۳ چهار مرتبه روشن و خاموش شود. اما اگر پایه ٦ بالا بود، پایه ۱۳ هفت مرتبه روشن و خاموش شود.

۳. با استفاده از while و if برنامه ای بنویسید که اگر پایه ٤، بالا بود، پایه ۱۳ خاموش باشد. اما اگر پایه ٤، پایین بود، تا مادامی که پایه ٤ پایین بود، پایه ۱۳ نیم ثانیه روشن و نیم ثانیه خاموش باشد و هنگامی که پایه ٤، دیگر پایین نبود، پایه ۱۳، به مدت ۲ ثانیه به طور ممتد روشن باشد و سپس خاموش شود.

٤. با استفاده از for برنامه ای بنویسید که پایه ۱۳ را شش مرتبه هر نیم ثانیه روشن و خاموش کند و سپس به مدت ۷ ثانیه خاموش شود

۵. با استفاده از for برنامه ای بنویسید که مجموع اعداد 3, 6, 9, 12, 15, 18, 21 را محاسبه کند و جواب را درون متغیر a بریزد.

٦. با استفاده از for برنامه ای بنویسید که مجموع اعداد 2, 5, 8, 11, 14, 17, 20 را محاسبه کند.

۷. با استفاده از for برنامه ای بنویسید که مجموع اعداد مضرب ۵ کوچکتر از ۱۰۰ را محاسبه کند.

فصل ۷: حروف و جملات و برقراری ارتباط از طریق سریال

بخش ۷-۱: حروف و جملات

همان طوری که در فصل ۲ مطرح شد، الفبای زبان ماشین فقط ۲ حرف دارد: ۰ و ۱ که با استفاده از همین ۰ و ۱ جمیع آنچه که مایلیم را بایست بیان کنیم. برای اینکه حروف انگلیسی و اعداد و سمبلها با استفاده از ۰ و ۱ بیان شوند، به هر یک از حروف، یک عدد باینری ۸ رقمی اختصاص داده اند که اصطلاحاً به این عددها، کد اسکی (ASCII) گفته می شود. در جدول ۷-۱ کد اسکی مربوط به حروف در مبنای دو و ده نمایش داده شده است. مثلاً کد اسکی ٦٥ معادل حرف A است و کد ٥٠، معادل عدد ۲ می باشد.

کد مبنای دو		
000	00000000	
001	00000001	☺
002	00000010	☻
003	00000011	♥
004	00000100	♦
005	00000101	♣
006	00000110	♠
007	00000111	•
008	00001000	◘
009	00001001	○
010	00001010	◙
011	00001011	♂
012	00001100	♀
013	00001101	♪
014	00001110	♫
015	00001111	☼
016	00010000	►
017	00010001	◄
018	00010010	↕
019	00010011	‼
020	00010100	¶
021	00010101	§
022	00010110	▬
023	00010111	↨
024	00011000	↑
025	00011001	↓
026	00011010	→
027	00011011	←
028	00011100	∟
029	00011101	↔
030	00011110	▲
031	00011111	▼

کد مبنای دو		
032	00100000	
033	00100001	!
034	00100010	"
035	00100011	#
036	00100100	$
037	00100101	%
038	00100110	&
039	00100111	'
040	00101000	(
041	00101001	)
042	00101010	*
043	00101011	+
044	00101100	,
045	00101101	-
046	00101110	.
047	00101111	/
048	00110000	0
049	00110001	1
050	00110010	2
051	00110011	3
052	00110100	4
053	00110101	5
054	00110110	6
055	00110111	7
056	00111000	8
057	00111001	9
058	00111010	:
059	00111011	;
060	00111100	<
061	00111101	=
062	00111110	>
063	00111111	?

کد مبنای دو		
064	01000000	@
065	01000001	A
066	01000010	B
067	01000011	C
068	01000100	D
069	01000101	E
070	01000110	F
071	01000111	G
072	01001000	H
073	01001001	I
074	01001010	J
075	01001011	K
076	01001100	L
077	01001101	M
078	01001110	N
079	01001111	O
080	01010000	P
081	01010001	Q
082	01010010	R
083	01010011	S
084	01010100	T
085	01010101	U
086	01010110	V
087	01010111	W
088	01011000	X
089	01011001	Y
090	01011010	Z
091	01011011	[
092	01011100	\
093	01011101	]
094	01011110	^
095	01011111	_

کد مبنای دو		
096	01100000	`
097	01100001	a
098	01100010	b
099	01100011	c
100	01100100	d
101	01100101	e
102	01100110	f
103	01100111	g
104	01101000	h
105	01101001	i
106	01101010	j
107	01101011	k
108	01101100	l
109	01101101	m
110	01101110	n
111	01101111	o
112	01110000	p
113	01110001	q
114	01110010	r
115	01110011	s
116	01110100	t
117	01110101	u
118	01110110	v
119	01110111	w
120	01111000	x
121	01111001	y
122	01111010	z
123	01111011	{
124	01111100	!
125	01111101	}
126	01111110	~
127	01111111	⌂

جدول ۷- ۱: کدهای اسکی

کد مبنای دو		
128	10000000	Ç
129	10000001	ü
130	10000010	é
131	10000011	â
132	10000100	ä
133	10000101	à
134	10000110	å
135	10000111	ç
136	10001000	ê
137	10001001	ë
138	10001010	è
139	10001011	ï
140	10001100	î
141	10001101	ì
142	10001110	Ä
143	10001111	Å
144	10010000	É
145	10010001	æ
146	10010010	Æ
147	10010011	ô
148	10010100	ö
149	10010101	ò
150	10010110	û
151	10010111	ù
152	10011000	ÿ
153	10011001	Ö
154	10011010	Ü
155	10011011	¢
156	10011100	£
157	10011101	¥
158	10011110	₧
159	10011111	ƒ

کد مبنای دو		
160	10100000	á
161	10100001	í
162	10100010	ó
163	10100011	ú
164	10100100	ñ
165	10100101	Ñ
166	10100110	ª
167	10100111	º
168	10101000	¿
169	10101001	⌐
170	10101010	¬
171	10101011	½
172	10101100	¼
173	10101101	¡
174	10101110	«
175	10101111	»
176	10110000	░
177	10110001	▒
178	10110010	▓
179	10110011	│
180	10110100	┤
181	10110101	╡
182	10110110	╢
183	10110111	╖
184	10111000	╕
185	10111001	╣
186	10111010	║
187	10111011	╗
188	10111100	╝
189	10111101	╜
190	10111110	╛
191	10111111	┐

کد مبنای دو		
192	11000000	└
193	11000001	┴
194	11000010	┬
195	11000011	├
196	11000100	─
197	11000101	┼
198	11000110	╞
199	11000111	╟
200	11001000	╚
201	11001001	╔
202	11001010	╩
203	11001011	╦
204	11001100	╠
205	11001101	═
206	11001110	╬
207	11001111	╧
208	11010000	╨
209	11010001	╤
210	11010010	╥
211	11010011	╙
212	11010100	╘
213	11010101	╒
214	11010110	╓
215	11010111	╫
216	11011000	╪
217	11011001	┘
218	11011010	┌
219	11011011	█
220	11011100	▄
221	11011101	▌
222	11011110	▐
223	11011111	▀

کد مبنای دو		
224	11100000	α
225	11100001	ß
226	11100010	Γ
227	11100011	π
228	11100100	Σ
229	11100101	σ
230	11100110	µ
231	11100111	τ
232	11101000	Φ
233	11101001	Θ
234	11101010	Ω
235	11101011	δ
236	11101100	∞
237	11101101	ø
238	11101110	∈
239	11101111	∩
240	11110000	≡
241	11110001	±
242	11110010	≥
243	11110011	≤
244	11110100	⌠
245	11110101	⌡
246	11110110	÷
247	11110111	≈
248	11111000	≈
249	11111001	∙
250	11111010	·
251	11111011	√
252	11111100	ⁿ
253	11111101	²
254	11111110	■
255	11111111	□

ادامه جدول ۷-۱

بنابراین هر حرف با استفاده از ۸ بیت بیان می شود. به همین خاطر است که متغیرهای char که برای نگهداری کردن از حروف می باشند ۸ بیت (معادل یک بایت) فضا از حافظه را اشغال می کنند. بنابراین هر متغیر char می تواند در خود حاوی یکی از این ۲۵٦ کد اسکی باشد.

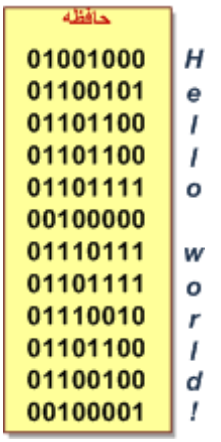
در زبان سی برای اینکه یک حرف را بیان کنیم، آن را بین دو تا علامت ' قرار می دهیم. به طور مثال، اگر بخواهیم حرف A را داخل متغیر c بریزیم می توانیم دستور زیر را بنویسیم:

```
char c = 'a';
```

طریقه تایپ کردن علامت '

علامتهای " و ' بر روی یک دکمه قرار گرفته اند. اگر دکمه Shift را پایین نگه دارید علامت " تایپ می‌شود و اگر دکمه Shift رها باشد علامت ' تایپ می‌شود.

هر متن و نوشته ای از کنار هم قرار گرفتن حروف (کاراکترها) به وجود می آیند. بنابراین عبارتی مثل Hello World! که از ۱۲ حرف تشکیل شده است، ۱۲ بایت از حافظه را به خود اختصاص می دهد. توجه دارید که فاصله (space) هم یک حرف است و برای ذخیره کردن آن نیز یک بایت لازم است و به همین خاطر نوشته Hello world! از ۱۲ حرف تشکیل شده است. در عکس زیر، عددی که به ازای Hello world! در حافظه ذخیره می شود را می بینید.



شکل ۷- ۱: نگهداری پیغام Hello world! در حافظه

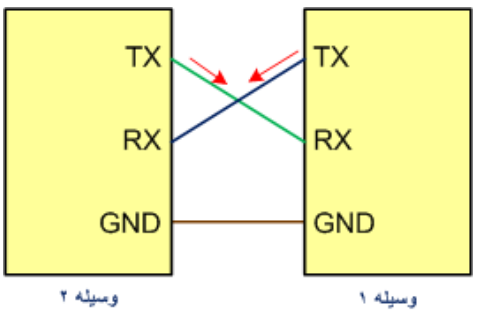
همان طوری که در جدول ۷-۱ می بینید کد اسکی معادل H، ۰۱۰۰۱۰۰۰ است. بنابراین همان طوری که در شکل ۷-۱ نشان داده شده است، در حافظه در خانه ای که مایلیم H ذخیره شود، در حقیقت عدد باینری 01001000 ذخیره می شود. سایر حروف نیز به همین طریق ذخیره شده اند.

بخش ۷-۲: آشنایی با سریال

یکی از روشهای رایج برای مرتبط کردن برد آردوینو با وسایل دیگر، استفاده کردن از سریال است. با استفاده از سریال می توانیم برد آردوینو را وسایلی مثل کامپیوتر و بردهای آردوینو دیگر و نیز ماژولهای مختلف مثل GSM (که از طریق آن می توانیم SMS بفرستیم و به اینترنت وصل شویم.) و کارتخوانهای RFID (به کارتهای مترو و حضور و غیاب اصطلاحاً RFID می گویند.) مرتبط کنیم.

برای اینکه دو تا وسیله از طریق سریال به هم متصل شوند می بایست سه تا پایه از آنها را به هم متصل کنیم. این سه پایه عبارتند از:

- زمین (GND): فرض کنید کسی اکنون از شما بپرسد که "برد آردوینو شما در چه ارتفاعی قرار دارد؟" شما می توانید ارتفاع برد آردوینو را نسبت به زمین و یا نسبت به میز و یا سطح آبهای آزاد بسنجید و در نتیجه جوابهای مختلفی به مخاطب بدهید. ولتاژ (پتانسیل) در مدارهای الکتریکی نیز همین گونه است و می بایست نسبت به نقطه معینی بیان شود. بنابراین هنگامی که می خواهیم دو وسیله با هم مرتبط شوند می بایست پایه های زمین آنها را به هم متصل کنیم تا ولتاژها را نسبت به نقطه همسانی بسنجند.
- پایه ارسال (TX): وسایل از طریق پایه TX خود، دیتا را به بیرون می فرستند.
- پایه دریافت (RX): وسایل از طریق پایه RX خود، دیتا را دریافت می کنند. بنابراین می بایست پایه TXD از یک وسیله را به پایه RX از وسیله دیگر متصل کنیم تا اطلاعاتی که توسط پایه TX از یک وسیله ارسال می شود به پایه RX از وسیله دیگر برسد و وسیله دوم اطلاعات را دریافت کند.



شکل ۷-۲: اتصال دو وسیله از طریق سریال

سرعت ارسال و دریافت اطلاعات

وسایل می توانند با سرعتهای مختلفی اطلاعات را ارسال و دریافت کنند. بنابراین هنگامی که می خواهیم دو وسیله از طریق سریال با هم مرتبط شوند، لازم است که سرعت سریال آنها را به عدد یکسانی تنظیم کنیم. در موقع برنامه نویسی برای آردوینو، برای اینکه سرعت ارتباط سریال را مشخص کنیم از دستور زیر استفاده می کنیم:

```
Serial.begin(سرعت تبادل اطلاعات);
```

یکی از سرعتهای رایج برای ارتباط سریال سرعت 9600 بیت بر ثانیه است. با استفاده از دستور زیر می توانیم سرعت ارتباط را به ۹۶۰۰ تنظیم کنیم:

Serial.begin(9600);

به طور معمول، سرعت ارتباط سریال، بر روی عددی مشخصی تنظیم می شود و در طول اجرای برنامه با همان سرعت تبادل اطلاعات صورت می گیرند و بنابراین دستور تنظیم سرعت را فقط یکبار در داخل setup صدا می زنیم.

بخش ۷-۳: ارسال اطلاعات از طریق سریال

آردوینو برای ارتباط سریال دستور ساده و قدرتمندی را فراهم کرده است که با استفاده از آن می توانیم مقدار متغیرها و نیز متن ها و پیغامهایی را به کامپیوتر و یا وسایل دیگر ارسال کنیم. ساختار دستور ارسال سریال به صورت زیر است

Serial.println(متغیر یا پیغامی که میخواهیم ارسال شود);

به طور مثال، اگر متغیری به نام number داشته باشیم و بخواهیم، مقدار آن را از طریق سریال برای کامپیوتر بفرستیم می توانیم دستور زیر را بنویسیم:

Serial.println(number);

در زبان سی، برای اینکه پیغامها و متنهایی را در بین برنامه بنویسیم، آن ها را در بین دو علامت نقل قول می گذاریم. علامت نقل قول به شکل " است. مثلا اگر بخواهیم پیغام Hello World را با استفاده از دستور println برای کامپیوتر ارسال کنیم، می بایست که آن را در بین علامت نقل قول به شکل زیر قرار دهیم:

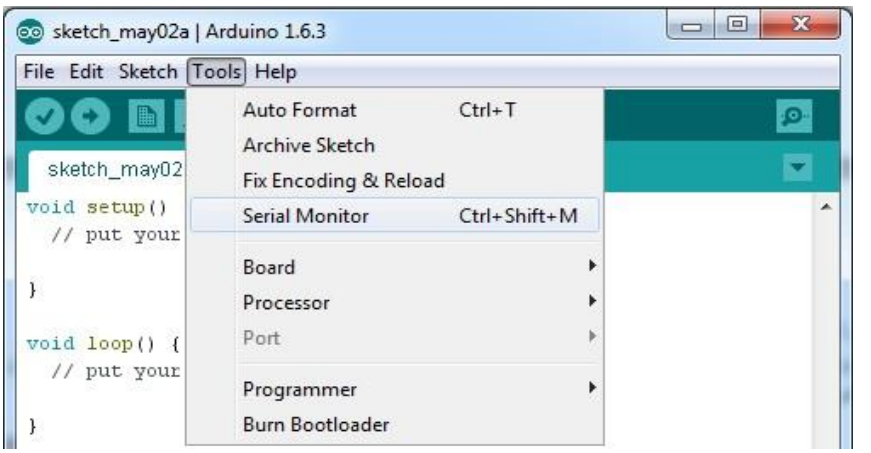
Serial.println(“Hello World”);

به طور نمونه، برنامه زیر پیغام !Hello World را هر ۲ ثانیه یکبار به کامپیوتر ارسال می کند.

مثال ۷-۱: ارسال پیغام !Hello World از طریق سریال به کامپیوتر

```
void setup() {
    Serial.begin(9600);  //سرعت ارتباط سریال را بر روی 9600 تنظیم کن
}

// the loop routine runs over and over again forever:
void loop() {
    Serial.println("Hello World!");  //پیغام مورد نظر را به کامپیوتر بفرست
    delay(2000);  //2 ثانیه صبر کن
}
```

در محیط برنامه نویسی آردوینو، برای اینکه بتوانیم پیغامهایی که از طریق سریال از برد آردوینو دریافت می شود را ببینیم می بایست به منوی Tools برویم و بر گزینه Serial Monitor کلیک کنیم.

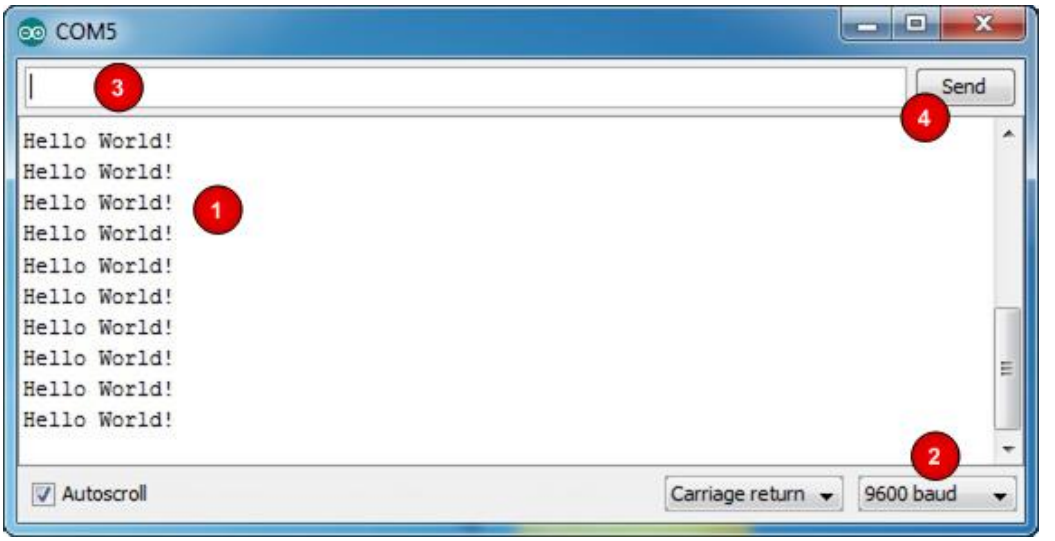


شکل ۷-۳: گزینه Serial Monitor از منوی Tools

امکانات پنجره Serial Monitor

شکل ۷–٤ را ببینید. در پنجره Serial Monitor:

- پیغامهایی که برد آردوینو ارسال می کند در فضایی که با عدد ۱ مشخص شده است، نمایش داده می شود.
- برای اینکه سرعتی که کامپیوتر اطلاعات را دریافت میکند تنظیم کنیم می بایست بر روی لیست پایین افتادنی که با ۲ مشخص شده است کلیک کنیم. به طور پیش فرض، سرعت تبادل اطلاعات بر روی گزینه ۹٦۰۰ است و با توجه به اینکه ما نیز در برنامه سرعت تبادل اطلاعات را به ۹٦۰۰ تنظیم کرده ایم، نیازی به تنظیم کردن سرعت تبادل اطلاعات در این پنجره نیست.
- هر گاه بخواهیم از کامپیوتر، اطلاعاتی را برای برد آردوینو بفرستیم، می توانیم اطلاعات مورد نظر را در محلی که با عدد ۳ مشخص شده است تایپ کنیم و سپس دکمه Send را بزنیم.



شکل ۷-۴: Serial Monitor

در زیر مثالهای دیگری را با استفاده از سریال می بینید.

مثال ۷-۲: ارسال اعداد ۰ تا ۷

برنامه ای بنویسید که اعداد ۰ تا ۷ را به کامپیوتر ارسال کند و سپس ۵ ثانیه صبر کند و مجدداً این کار را تکرار کند.

پاسخ:

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int n;
    for(n = 1; n <= 7; n = n + 1)
    {
        Serial.println(n);
    }
    delay(5000);
}
```

مثال ۷-۳: ارسال وضعیت پایه ۴

برنامه ای بنویسید که وضعیت پایه ۴ را چک کند. اگر پایه ۴، پایین (LOW) بود، پیغام Pin 4 is Low! و در غیر این صورت پیغام Pin 4 is High! را نمایش دهد.

پاسخ:

```
void setup() {
    pinMode(4,INPUT);
    Serial.begin(9600);
}

void loop() {
    if(digitalRead(4) == LOW)
    Serial.println("Pin 4 is Low!");
    else
    Serial.println("Pin 4 is High!");

    delay(200);
}
```

مثال ۷-۴: ثانیه شمار

متغیر n را با عدد ۰ مقداردهی اولیه کنید. سپس هر ثانیه یکبار مقدار آن را زیاد کنید و مقدار آن را ارسال کنید. هنگامی که مقدار n به ۵۹ رسید، شمارش را از ۰ شروع کنید.

پاسخ:

برای اینکه n از ۰ تا ۵۹ بشمارد از یک حلقه for استفاده می کنیم و در داخل حلقه مقدار n را با استفاده از سریال، ارسال می کنیم. برای اینکه هر ثانیه یکبار، مقدار متغیر n زیاد شود، در داخل حلقه for یک delay یک ثانیه ای می گذاریم تا پس از هر بار ارسال، یک ثانیه صبر کند و سپس مقدار n را زیاد کند.

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int n;
    for(n = 0; n <= 59; n = n + 1)
    {
        Serial.println(n);
        delay(1000);
    }
}
```

دستورات println **و** print

هنگامی که از دستور println استفاده میکنیم، پس از اینکه پیغامی که اعلام کردیم، ارسال شد، به سر خط می رود و بنابراین پیغام بعدی که از طریق سریال می کنیم، سر سطر بعدی نمایش داده می شود. به طور نمونه، در مثال ۷-۱ که پیغام Hello World! را به کامپیوتر ارسال کردیم، پس از ارسال شدن پیغام Hello World! به سر سطر بعدی رفت و پیغام بعدی را در خط جداگانه ای نمایش داد. گاهی اوقات مایل هستیم که چندین پیغام را در یک خط بنویسیم و سپس به سر سطر برویم. اگر از دستور print استفاده کنیم، پس از ارسال پیغام، به سر سطر نمی رود، بلکه پیغام بعدی را نیز در ادامه همین پیغام بعدی نمایش می دهد. به طور نمونه، مجموعه دستورات زیر، پیغامی به شکل "a is 5" را نمایش می دهد.

```
int a = 5;
Serial.print("a is ");
Serial.println(a);
```

در مجموعه دستورات فوق، ابتدا پیغام " a is" را به کامپیوتر می فرستیم و چون از دستور print استفاده کرده ایم، به سر سطر نمی‌رود، بلکه پیغام بعدی را نیز در ادامه همین پیغام می نویسد. در دستور بعد، ما مقدار a را به کامپیوتر ارسال کرده ایم که ۵ است. پس در ادامه پیغام قبلی عدد ۵ ظاهر می شود. با توجه به اینکه از دستور println استفاده کرده ایم، پس از نمایش دادن عدد ۵ به سر سطر می رود و پیغامهای بعدی را در خط جدیدی نمایش می دهد.

چگونگی رفتن سر خط (برای علاقه مندان)

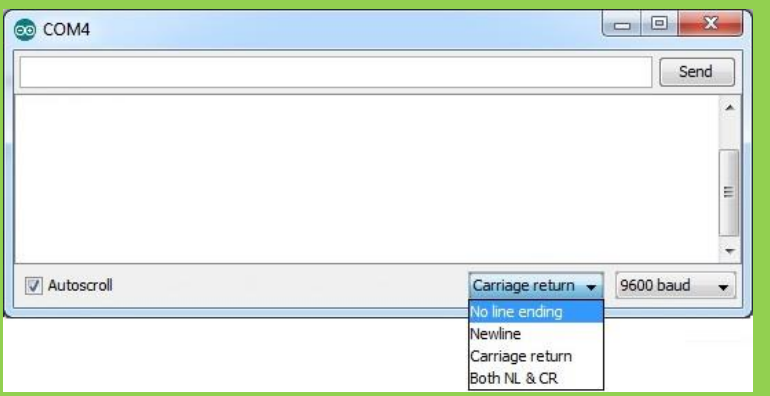
اگر به جدول ۷-۱ مراجعه کنید کاراکتری که دارای کد اسکی 10 است، را در حالت عادی در نوشته‌ها استفاده نمی کنیم. در تبادل اطلاعات، کد ۱۰ به معنی رفتن به سر خط است. در زبان سی، با فشار دادن کلید مربوط به حروف الفبا می توانیم حروف انگلیسی (مثلا a و یا b) را تایپ کنیم. اما کلیدی برای تایپ کردن کاراکتر با کد اسکی ۱۰ نداریم. بنابراین در زبان سی، برای اینکه دستور سر خط را بیان کنیم \n را تایپ می کنیم و کامپایلر در زمان کامپایل آن را به کاراکتر با کد اسکی ۱۰ تبدیل می کند. مثلا وجود \n در متن دستور زیر باعث می شود که پس از نمایش ۱۲۳ به سر خط بعد برود و ۴۵۶ را در خط بعدی نمایش دهد و ۷۸۹ را در خط بعد نمایش دهد:

```
Serial.print("123\n456\n789\n");
```

در برخی محیط ها، برای بیان کردن سر سطر فقط کد اسکی ۱۰ را بیان می کنند، در حالی که در برخی محیط های دیگر، کد اسکی ۱۰ را همراه با کد اسکی ۱۳ برای بیان کردن رفتن به سر سطر استفاده می کنند. کد اسکی ۱۳ در زبان سی، با استفاده از \r بیان می‌شود. دستور println در حقیقت با اضافه کردن \r\n به انتهای نوشته‌هایی که می نویسیم، باعث رفتن به سر سطر می‌شود. بنابراین دو دستور زیر، معادل یکدیگرند و ما برای رفتن به سر سطر معمولا از همان println استفاده می‌کنیم.

```
Serial.print("Hello World!\r\n");
Serial.println("Hello World!");
```

در هنگام ارسال سریال از کامپیوتر به برد آردوینو، در پنجره Serial Monitor (شکل ۷-۴)، ما می-توانیم انتخاب کنیم که پس از نوشته هایی که تایپ می کنیم آیا مایلیم که کاراکتر سر سطر اضافه شود یا نه. برای این کار در پایین صفحه، یک لیست پایین افتادنی در کنار سرعت ارسال وجود دارد. منظور از هر یک از گزینه ها به قرار زیر است:



No line ending: هیچ کاراکتری اضافه نکن و هر آنچه که تایپ کرده ایم را ارسال کن
Newline: کاراکتر \n را اضافه کن.
Carriage Return: کاراکتر \r را اضافه کن.
Both NL & CR: هر دو کاراکتر \n و \r را اضافه کن.

مثال ۷-۵: ارسال اعداد ۰ تا ۷

برنامه ای بنویسید که اعداد ۰ تا ۷ را به کامپیوتر ارسال کند و البته جمیع اعداد ۰ تا ۷ را در یک خط نمایش دهد و برای اینکه اعداد از هم مجزا شود، بین آنها کاما بگذارید و بعد از عدد ۷، به سر سطر برود.

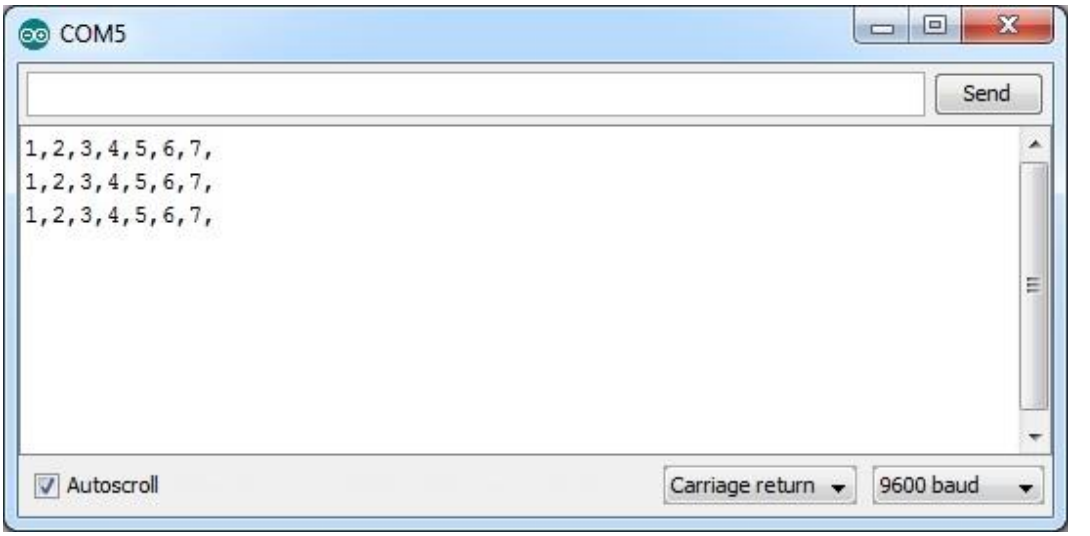
پاسخ:

این برنامه، مشابه مثال ۷-۲ است با این تفاوت که شکل نمایش دادن پیغامها، قدری تغییر کرده است.

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int n;
    for(n = 1; n <= 7; n = n + 1)
    {
        Serial.print(n);
        Serial.print(",");
    }

    Serial.println("");
    delay(5000);
}
```

در برنامه فوق، دستور Serial.print(n) مقدار n را ارسال می کند و دستور Serial.print(",") باعث می‌شود که بعد از هر عددی یک کاما نمایش داده شود. هنگامی که حلقه for به پایان رسید و از حلقه بیرون

آمد، دستور Serial.println("") سبب می شود که به سر سطر برود. در عکس زیر، شکل پیغامی که توسط برنامه فوق، برای کامپیوتر ارسال می شود را می بینید:



مثال ۷-۶: ارسال اعداد زوج

برنامه ای بنویسید که اعداد زوج بین ۲ تا ۹۰ را به کامپیوتر ارسال کند و البته جمیع اعداد را در یک خط نمایش دهد و برای اینکه اعداد از هم مجزا شود، بین آنها کاما بگذارید.

پاسخ:

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int n;
    for(n = 2; n <= 90; n = n + 2)
    {
        Serial.print(n);
        Serial.print(",");
    }

    Serial.println("");
    delay(5000);
}
```

مثال ۷-۷: ساعت (برای علاقه مندان)

برنامه ای بنویسید که اعلام کند از زمان روشن شدن برد آردوینو، چند ساعت و چند دقیقه و چند ثانیه میگذرد

پاسخ:

برای اینکه هر ثانیه یکبار مقدار n را زیاد کنیم، تاخیری به اندازه یک ثانیه در داخل loop می گذاریم و پس از ۱ ثانیه مقدار آن را زیاد می کنیم. یعنی به صورت زیر:

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    int hour;
    int minute;
    int second;
    for(hour = 0; hour < 23; hour = hour + 1)
    {
        for(minute = 0; minute < 59; minute = minute + 1)
        {
            for(second = 0; second < 60; second = second + 1)
            {
                Serial.print(hour);
                Serial.print(":");
                Serial.print(minute);
                Serial.print(":");
                Serial.println(second);
                delay(1000);
            }
        }
    }
}
```

در برنامه فوق، از سه تا حلقه تو در تو استفاده شده است. در این برنامه، ابتدا داخلی ترین حلقه، که حلقه مربوط second است از ۰ تا ۵۹ می شمارد. هنگامی که این حلقه پایان پذیرفت، با توجه به اینکه داخل حلقه مربوط به minute هستیم، دستوراتی که پس از حلقه مربوط به ثانیه وجود دارند اجرا میشوند و چون دستوری بعد از حلقه ثانیه وجود ندارد پس آخرین دستور مربوط به حلقه minute، که همان دستور minute=minute+1 است، اجرا می شود، پس مقدار minute یکی زیاد می شود و چون شرط minute<59 برقرار است، مجددا وارد حلقه مربوط به minute میشود و دستورات داخل حلقه minute که همین حلقه مربوط به second است اجرا می شود و به عبارت دیگر، یک بار دیگر ثانیه از ۰ تا ۵۹ می‌شمارد.

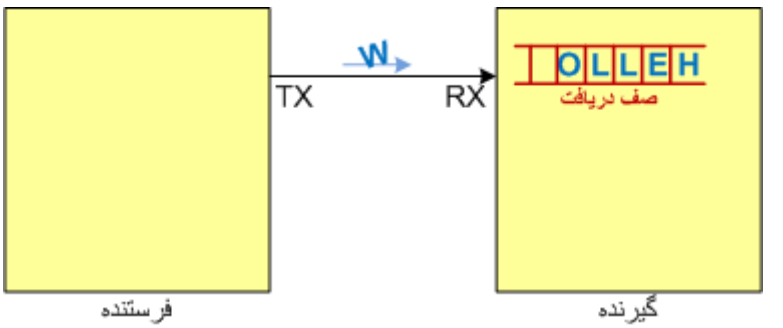
هنگامی که دقیقه نیز ۵۹ شمارش کرد، حلقه minute نیز به پایان می رسد و در حلقه hour، دستور بعد از حلقه minute اجرا می شود و چون دستور خاصی بعد از حلقه minute نیست پس آخرین دستور موجود در حلقه hour که همانا hour = hour + 1 است اجرا می شود و ساعت یکی زیاد می شود.

بخش ۷-۴: دریافت از طریق سریال

در بخش قبل دیدیم که چطور می‌توانیم پیغامهایی را از برد آردوینو به کامپیوتر ارسال کنیم. در این قسمت می‌خواهیم، پیغامهایی را از کامپیوتر به برد آردوینو ارسال کنیم. همان طوری که در توضیح شکل۷-۳ گفته شد، با استفاده از Serial Monitor می توانیم اطلاعاتی را برای برد آردوینو ارسال کنیم.

صف دریافت

یک صف نانوایی را در نظر بگیرید. مشتریان، هنگامی که به نانوایی می رسند به ترتیب پشت سر آخرین نفر می ایستند و شاطر نانوایی از سر صف به ترتیب به نیاز مشتریان رسیدگی می کند. برای دریافت سریال نیز صفی وجود دارد، که هنگامی که اطلاعات دریافت می شوند به ترتیب در داخل صف قرار می گیرند. ما با استفاده از دستور Serial.available() می توانیم چک کنیم که آیا هیچ حرفی در داخل صف دریافت وجود دارد یا نه و از طریق دستور Serial.read() می توانیم یک حرف از سر صف دریافت سریال را دریافت کنیم.



شکل ۷-۵: صف دریافت

اگر صف دریافت وجود نداشت، ما مجبور بودیم که دائما چشم انتظار دریافت اطلاعات جدید باشیم و اگر در زمان دریافت اطلاعات، آماده دریافت نبودیم، آنها را از دست می دادیم. اما وجود صف، باعث می شود که حروف هنگامی که دریافت می شوند، در صف پشت سر هم بشینند و برنامه هر موقع فرصت آزاد داشت به سراغ صف برود و اطلاعات دریافت شده را رسیدگی کنند.

به طور مثال، برنامه های زیر را ببینید.

مثال ۷-۸

برنامه ای بنویسید که هر گاه حرفی از طریق سریال دریافت شده بود، همان را برای کامپیوتر پس بفرستد.

پاسخ:

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    char c;
    if(Serial.available())
    {
        c = Serial.read();
        Serial.print(c);
    }
}
```

مثال ۷-۹

برنامه زیر حرفی را که از طریق کامپیوتر ارسال شده است را بررسی می کند. اگر حرف دریافت شده، حرف A یا a بود، چراغ متصل به پایه ۱۳ را روشن می کند و اگر حرف b یا B بود، چراغ متصل به پایه ۱۳ را خاموش می کند.

```
void setup()
{
    Serial.begin(9600);
    pinMode(13,OUTPUT);
}

void loop()
{
    char c;
    if(Serial.available())
    {
        c = Serial.read();

        if((c == 'a')||(c == 'A'))
        {
            digitalWrite(13,HIGH);
        }
        else
        {
            if((c == 'b')||(c == 'B'))
            {
                digitalWrite(13,LOW);
            }
        }
    }
}
```

همان طوری که می بینید با استفاده از سریال ما می توانیم برنامه هایی بنویسیم که از طریق کامپیوتر، وسایل مختلف را کنترل کنیم! به طور مثال، اجزای یک ربات را به برد آردوینو متصل کنیم و از طریق کامپیوتر به برد آردوینو فرمان دهیم که ربات را در جهتهای مختلف حرکت کند.

صف، در کامپیوتر استفاده های فراوانی دارد. به طور مثال، برای کی بورد نیز صفی وجود دارد، که وقتی شما تایپ می کنید، حروف ابتدا در داخل این صف قرار می گیرند و سپس به دست برنامه هایی که بر روی کامپیوتر اجرا می شوند می رسد.

تمرین

۱. برنامه ای بنویسید که از طریق سریال با سرعت ۹۶۰۰ بیت بر ثانیه، نام شما را به کامپیوتر ارسال کند.

۲. برنامه ای بنویسید که مضربهای ٤ که کوچکتر از ۳۰ هستند را به کامپیوتر ارسال کند.

۳. برنامه ای بنویسید که اعداد فرد بین ۱۰ تا ۲۰ را هر ثانیه یکبار به کامپیوتر ارسال کند. (یعنی اعداد فرد بین ۱۰ تا ۲۰ را به کامپیوتر ارسال کند و سپس یک ثانیه صبر کند و مجددا بعد از یک ثانیه اعداد فرد بین ۱۰ تا ۲۰ را ارسال کند.)

٤. دو کلید به پایه ۵ و ٦ از آردوینو متصل کنید. سپس برنامه ای بنویسید که اگر کلید پایه ۵ فشرده بود، پیغام LOW را به کامپیوتر بفرستد و اگر کلید پایه ۵ رها بود، پیغام HIGH را ارسال کند. همچنین کلید متصل به پایه ٦ سرعت ارسال پیغام به کامپیوتر را مشخص کند؛ اگر کلید پایه متصل به پایه ٦ فشرده بود، هر نیم ثانیه یکبار پیغام به کامپیوتر ارسال شود و اگر این کلید رها بود، هر ۲ ثانیه یکبار به کامپیوتر پیغام ارسال شود.

۵. برنامه ای بنویسید که کاراکتری را از کامپیوتر دریافت کند. سپس ٤ تا به مقدار آن اضافه کند و آن را به کامپیوتر ارسال کند. مثلا اگر کاراکتر a را از کامپیوتر دریافت کرد، کاراکتر e (که ٤ تا جلوتر از a هست) را به کامپیوتر ارسال کند.

٦. برنامه ای بنویسید که دو حرف از طریق سریال از کامپیوتر دریافت کند و هر یک از آنها که کد اسکی بزرگتری دارد را به کامپیوتر پس بفرستد. مثلا اگر حرف b و d را از طریق سریال به آردوینو بفرستیم، کاراکتر d را به کامپیوتر پس بفرستد.

۷. می خواهیم برنامه ای بنویسیم که چراغ پایه ۱۳ روشن و خاموش شود. اما سرعت چشمک زدن آن از طریق سریال قابل کنترل باشد. اگر کاراکتر S را از طریق سریال بفرستیم هر ۲ ثانیه یکبار روشن و خاموش شود. اگر کاراکتر M را از طریق سریال بفرستیم هر ثانیه یکبار روشن و خاموش شود. اگر کاراکتر F را بفرستیم هر 0.5 یکبار روشن و خاموش شود. (راهنمایی: یک متغیر int به نام t تعریف کنید. پس از هر بار چشمک زدن وضعیت بافر سریال را چک کنید و اگر کاراکتری درون آن بود، آن را دریافت کنید. اگر کاراکتر دریافتی S بود، متغیر t را با مقدار ۲۰۰۰ مقداردهی کنید. اگر کاراکتر دریافتی M بود، متغیر t را با مقدار ۱۰۰۰ مقداردهی کنید و اگر F بود، متغیر t را با ۵۰۰ مقداردهی کنید. در طی برنامه برای میزان تاخیر بین روشن شدن و خاموش شدن چراغ مقدار t را استفاده کنید. همچنین فراموش نکنید که متغیر t را با عدد ۵۰۰ یا ۱۰۰۰ یا ۲۰۰۰ مقداردهی اولیه کنید، در غیر این صورت در لحظه آغاز با سرعت خیلی زیادی شروع به روشن و خاموش شدن می کند به گونه ای که چشمک زدن آن قابل دیدن نیست.)

۸. می خواهیم با استفاده از برد آردوینو یک رقص نور ایجاد کنیم به گونه ای که از طریق سریال بتوانیم جهت حرکت رقص نور و سرعت آن را تغییر دهیم. شش تا LED به پایه های ۸ تا ۱۳ از برد آردوینو متصل کنید. سپس برنامه ای بنویسید که: هرگاه کاراکتر R (حرف اول از کلمه Right) از سریال دریافت شد، ابتدا پایه ۸ روشن شود، سپس پایه ۹ روشن شود و به همین ترتیب تا پایه ۱۳ روشن شود و سپس ابتدا پایه ۸ خاموش شود و بعد پایه ۹ خاموش شود و به همین ترتیب تا پایه ۱۳ خاموش شود. اما اگر کاراکتر L (حرف اول از کلمه Left) دریافت شد، ابتدا پایه ۱۳ روشن شود، سپس پایه ۱۲ روشن شود و به همین ترتیب از راست به چپ چراغها روشن شوند و سپس به همین ترتیب از راست به چپ چراغها خاموش شوند. اگر حرف F (حرف اول از کلمه Fast) از کامپیوتر دریافت شد، تاخیر بین روشن شدن هر چراغ با چراغ بعدی 0.3 ثانیه باشد و اگر حرف S (حرف اول از کلمه Slow) از کامپیوتر دریافت شد، تاخیر بین روشن شدن هر چراغ تا چراغ بعدی یک ثانیه باشد. (راهنمایی: دو متغیر int به نامهای direction و t تعریف کنید. پس از هر بار رقص نور وضعیت بافر سریال را چک کنید و اگر کاراکتری درون آن بود، آن را دریافت کنید. اگر کاراکتر دریافتی S یا F بود، متغیر t را با مقادیر ۳۰۰ یا ۱۰۰۰ مقداردهی

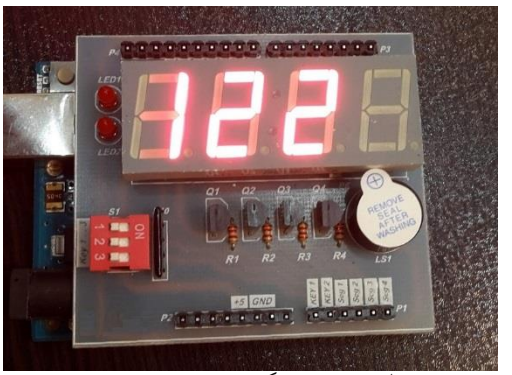
کنید و در طی برنامه برای میزان تاخیر بین روشن شدن و خاموش شدن چراغها مقدار t را استفاده کنید. همچنین هرگاه کاراکتر دریافتی R بود، متغیر direction را با صفر و هرگاه کاراکتر دریافتی L بود، متغیر direction را با ۱ مقداردهی کنید و در طول برنامه بر حسب اینکه مقدار متغیر direction، یک یا صفر است، از راست به چپ یا از چپ به راست چراغها را روشن کنید.)

فصل ۸: LCD کاراکتری

بخش ۸-۱: آشنایی با ال سی دی کاراکتری

بسیاری از سیستم های کنترلی که در اطراف ما وجود دارند، اطلاعاتی را به ما نمایش می دهند. برای بیان کردن اطلاعات، از روشهای مختلفی استفاده شده است از جمله وسایلی که برای نمایش اطلاعات استفاده می شوند عبارتند از LED، سون سگمنت و ال سی دی (LCD). در این فصل طریقه استفاده از LCD کاراکتری را می بینید.

شکل ۸-۲: ال سی دی (LCD)



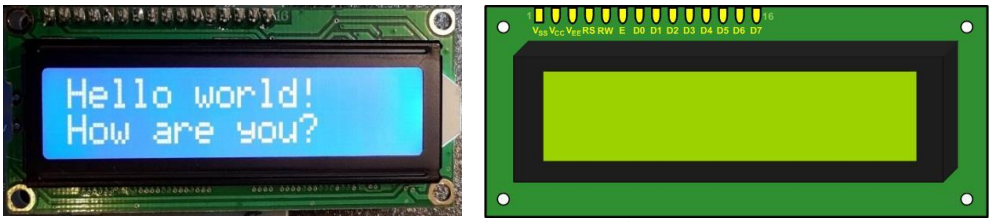
شکل ۸-۱:سون سگمنت (7-segment)

ال سی دی کاراکتری دارای خانه های هم شکل و هم اندازه ای است که در هر یک از خانه ها می‌توانیم هر کدام از کاراکترهای اسکی که مایلیم را نمایش دهیم. در شکل زیر یک ال سی دی کاراکتری را می‌بینید که دارای دو سطر است و در هر سطر، ۱۶ خانه وجود دارند. در سطر بالایی پیغام Hello world! و در سطر پایینی پیغام How are you? را نمایش داده ایم.

شکل ۸-۳: عکسی نزدیک از ال سی دی کاراکتری که در آن ۱۶ خانه و نقاط داخل خانه ها قابل دیدن است

پایه های ال سی دی کاراکتری

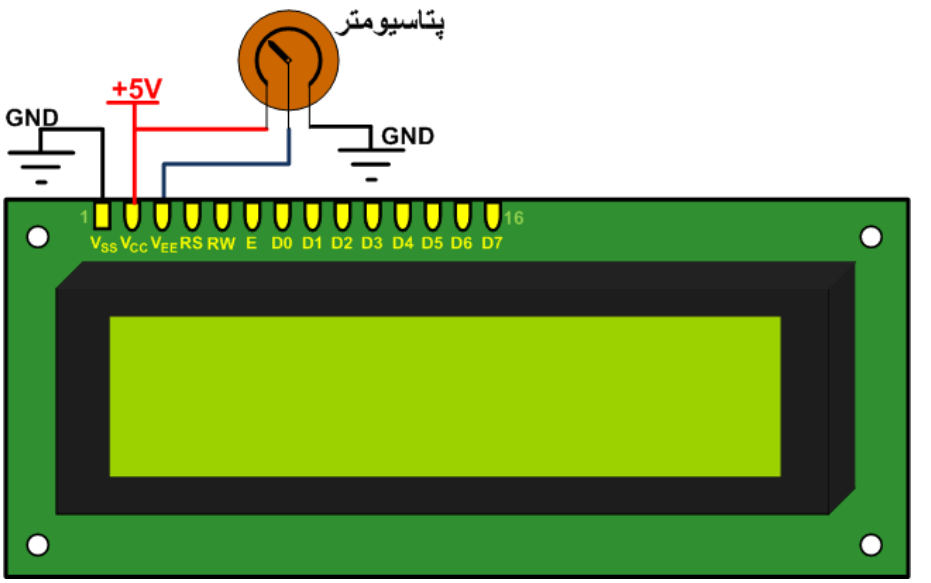
در عکس زیر، پایه های ال سی دی کاراکتری به همراه کاربردی که هر یک از آنها دارند نمایش داده شده است. در زیر به توضیح کاربرد هر یک از پایه ها می پردازیم.



شکل ۸-۴: پایه های ال سی دی (LCD)

پایه های ولتاژ

پایه های VCC و VSS: ال سی دی برای اینکه کار کند، به برق نیاز دارد. پایه های VCC و VSS برق را برای ال سی دی فراهم می کنند. ال سی دی های کاراکتری معمولا با ولتاژ ۵ ولت کار می کنند. پایه VCC می بایست به ولتاژ 5V+ و پایه VSS به ۰ ولت متصل شود. ولتاژ ۵ ولت مورد نیاز برای ال سی دی را می توانیم از پایه های برق برد آردوینو فراهم کنیم. شکل زیر را ببینید.



شکل ۸-۵: فراهم کردن ولتاژهای پایه های ولتاژ در ال سی دی

پایه VEE: با ولتاژی که به پایه VEE از ال سی دی متصل می کنیم کنتراست (contrast) صفحه نمایش را مشخص می کنیم. ولتاژی که به این پایه متصل می کنیم، می تواند بین ۰ تا ۵ ولت باشد. هر چه ولتاژ بیشتر باشد، نوشته های صفحه نمایش پر رنگ تر نمایش داده می شوند و البته اگر ولتاژ کنتراست بیش از حد زیاد باشد، همه صفحه نمایش سیاه دیده می شود. برای فراهم کردن ولتاژ کنتراست، می توانیم با استفاده از یک پتانسیومتر، مداری به شکل فوق ببندیم.

پایه های دیجیتال

پایه های دیجیتال ال سی دی که توضیح آنها در زیر آمده است، را می بایست به پایه های ورودی/خروجی برد آردوینو متصل کنیم. یعنی همان پایه هایی که سمت راست برد آردوینو قرار گرفته‌اند.

در زیر توضیحاتی کلی در مورد نقش پایه های ال سی دی (LCD) ذکر شده است. می توانید از خواندن توضیحات زیر صرف نظر کنید و مستقیما به سراغ برنامه نویسی برای آردوینو بروید.

پایه RS و پایه های D0 تا D7

در محیط word هنگامی که مطلبی را تایپ می کنید، یک آیکون مکان نما (Cursor) وجود دارد که به شما نقطه ای که در آن تایپ می شود را نشان می دهد. با استفاده از مکان نما، به نقطه ای که می‌خواهید در آن تایپ کنید، می روید و سپس شروع به تایپ کردن می کنید. هنگام نمایش دادن بر روی ال‌سی‌دی‌ها و صفحه های نمایش نیز، یک مکان نما وجود دارد که ما ابتدا آن را به نقطه ای که می‌خواهیم در آن بنویسیم می بریم و سپس پیغامی که می‌خواهیم نمایش داده شود را ارسال می‌کنیم.

ما دو چیز ممکن است برای ال سی دی ارسال کنیم: دستور و دیتا. با استفاده از دستورات، می‌توانیم به ال سی دی بگوییم که محتوای همه خانه های صفحه نمایش را پاک کند، یا ال سی دی خاموش شود یا مکان نما بر روی خانه مشخصی از ال سی دی قرار گیرد. هنگامی که دیتا بفرستیم، در خانه‌ای از ال سی دی که مکان نما روی آن است، قرار می گیرد و مکان نما یک خانه، جلو می‌رود.

از طریق پایه RS مشخص می کنیم که آنچه که داریم به ال سی دی می فرستیم، دیتا است یا اینکه دستور است و از طریق پایه های D0 تا D7 دستور و یا دیتا را ارسال می کنیم.

پایه R/W (Read/Write)

از طریق پایه RW مشخص می کنیم که می خواهیم به ال سی دی اطلاعات بفرستیم یا اینکه از آن اطلاعاتی دریافت کنیم. به طور معمول، ما برای LCD اطلاعات می‌فرستیم و دریافت اطلاعات از LCD خارج از دامنه کاری ماست. بنابراین پایه R/W را مستقیما به زمین متصل می‌کنیم تا مقدار صفر منطقی به این پایه داده شود و LCD در حالت دریافت اطلاعات (Write) قرار گیرد.

پایه E (Enable)

هنگامی که اطلاعات را بر روی پایه های ال سی دی قرار می‌دهیم، پایه Enable را یکبار روشن و خاموش می کنیم. هنگامی که پایه E از ۱ به ۰ تغییر کند (از ۵ ولت به ۰ ولت تغییر کند) ال سی دی متوجه

می‌شود که اطلاعات جدیدی بر روی پایه‌های D0 تا D7 قرار گرفته است. بنابراین اطلاعات روی این پایه‌ها را بر می‌دارد.

توضیحاتی برای علاقه مندان

اگر به پایه RS، یک منطقی را بدهیم، LCD اطلاعاتی که از طریق پایه های D0 تا D7 دریافت می کند را به عنوان دیتا در نظر میگیرد و اگر پایه RS، را صفر کنیم، LCD اطلاعاتی که از پایه های D0 تا D7 دریافت کند را به عنوان دستور در نظر می گیرد.
برای ارسال دیتا به LCD، کد اسکی کاراکتری که مایلیم بر روی ال سی دی نمایش داده شود را بر روی پایه های D0 تا D7 قرار می دهیم.
هنگامی که می خواهیم دستور بفرستیم، عدد باینری معادل دستور را بر روی پایه های D0 تا D7 می-گذاریم. هنگامی که دستور به دست LCD می رسد، LCD کار مربوطه را انجام می دهد. در جدول زیر لیست برخی از دستورات را می بینید:

عدد	دستور
۱	پاک کردن نوشته های صفحه نمایش
۸	خاموش کردن صفحه نمایش (LCD)
۱۲	صفحه نمایش روشن شود و مکان نما، حذف (خاموش) شود.
۱۴	صفحه نمایش روشن شود و مکان نما، به حالت چشمک زن روشن شود.
۱۲۸	مکان نما، به ابتدای خط اول برود.
۱۹۲	مکان نما به ابتدای خط دوم برود.

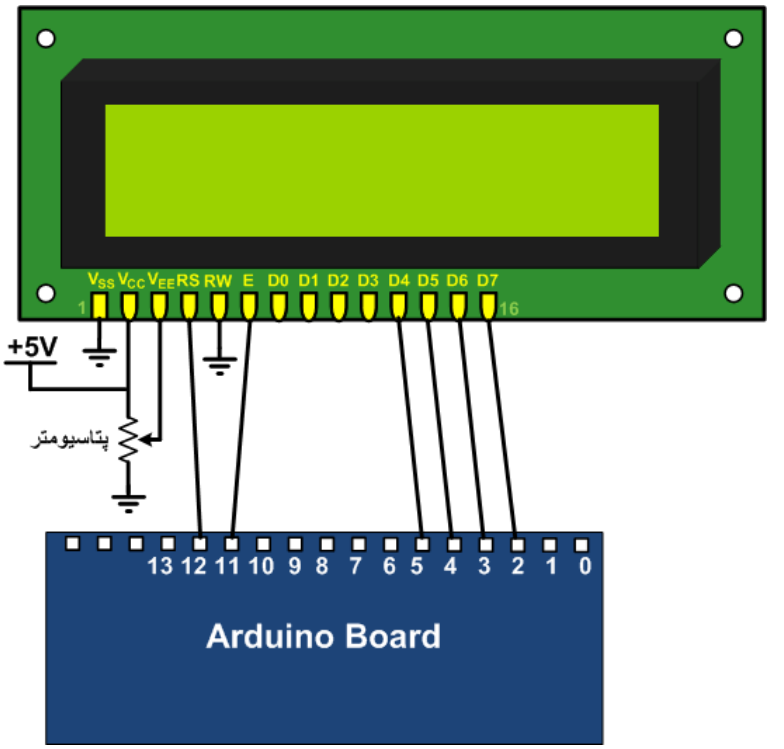
ارتباط چهار بیتی و ارتباط هشت بیتی

به دو طریق می توان با ال سی دی، ارتباط برقرار کرد. در روش نخست، جمیع پایه های D0 تا D7 برای برقراری ارتباط استفاده می شود و در روش دوم، فقط از پایه های D4 تا D7 استفاده می شود. برای اینکه سیم های کمتری بکشیم و نیز تعداد کمتری از پایه های ورودی/خروجی برد آردوینو استفاده شود، از روش دوم برای مرتبط کردن ال سی دی و آردوینو استفاده می شود. شکل ۸-٦ را ببینید.

بخش ۸-۲: برنامه نویسی برای ال سی دی کاراکتری

بکار گیری کتابخانه LCD

بشر امروز، گنجینه ای از دانش و علوم را از گذشتگان به ارث برده است و آنچه که اکنون ما داریم ثمره کوشش و تحقیقات گذشتگان است. ما نیز بر این گنجینه گذشتگان می افزاییم و آن را در اختیار آیندگان قرار می دهیم. به طور مثال، چرخ را گذشتگان اختراع کرده اند و اکنون دانش استفاده کردن از چرخ، در اختیار ماست. بنابراین با استفاده از چرخ می توانیم وسایل جدیدی را بسازیم. به قولی "گذشتگان کِشتند و ما خوردیم. ما نیز می کاریم تا آیندگان درو کنند."



شکل ۸-۶: اتصال پایه های دیجیتال ال سی دی به برد آردوینو

در دنیای برنامه نویسی نیز، این ایده مطرح است که لازم نباشد هر برنامه نویس به طور جداگانه همه برنامه‌ها را از آغاز بنویسد. بلکه یکبار هر برنامه ای نوشته شود و همه بشر از آن بهره مند شوند. مثلا یکبار برنامه مربوط به استفاده کردن از LCD، نوشته شود و همه بتوانند با استفاده از LCD وسایل جدیدی بسازند.

اصطلاحا به برنامه ای که از قبل نوشته شده باشد و قابل استفاده در برنامه های دیگر باشد کتابخانه گفته می شود. به طور نمونه، برای اینکه محیط برنامه نویسی آردوینو، به وجود آید افراد مختلفی در سراسر جهان، کتابخانه هایی را جهت کار کردن با وسایل مختلف از قبیل LCD، ماژول اینترنت، ماژول SMS و سنسور شتاب و سنسور GPS و غیره نوشته اند و ما می توانیم با استفاده از کتابخانه ها، وسایل جدیدی را بسازیم.

هنگامی که می خواهیم از برنامه هایی که از قبل آماده است در برنامه خود استفاده کنیم، در ابتدای برنامه خود از دستوری به نام #include استفاده می کنیم. به طور مثال، برای استفاده کردن از برنامه هایی که از قبل برای کار کردن با LCD فراهم شده اند از دستور زیر استفاده می کنیم:

```
#include <LiquidCrystal.h>
```

معرفی پایه های LCD در برنامه

کتابخانه ها به صورت فایلهایی فراهم شده اند. به طور نمونه، در مثال فوق LiquidCrystal.h یک فایل است که امکان استفاده از LCD را فراهم می کند. پایه های دیجیتال LCD را می توانیم به هر یک از پایه های برد آردوینو که مایلیم متصل کنیم. پس لازم است که پایه هایی از آردوینو که به آنها پایه های LCD متصل شده است را معرفی کنیم. برای اینکار از دستور زیر استفاده می کنیم:

```
LiquidCrystal   lcd(RS, Enable, D4, D5, D6, D7);
```

به طور مثال، اگر همانند شکل ۸-٦، پایه های RS و Enable و D4 و D5 و D6 و D7 به ترتیب به پایه‌های ۱۲ و ۱۱ و ۵ و ٤ و ۳ و ۲ متصل شده باشند، می توانیم دستور زیر را بنویسیم:

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

در بردهای LCD که مثالهای آردوینو بر حسب آن نوشته شده است، ترتیب اتصال پایه های LCD به برد آردوینو به صورت فوق الذکر است. اما در برد LCD که در ایران مرسوم است (شکل ۸-۲) ترتیب اتصال پایه‌ها به گونه ای است که می بایست از دستور زیر استفاده کنید:

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

طریقه متصل کردن برد LCD (LCD shield) به برد آردوینو

از زیر برد LCD چهار دسته پین (سیخهای نازک سیمی) بیرون آمده است. بر روی برد آردوینو ۴ دسته پایه سیاه رنگ وجود دارد که از آنها به عنوان پایه‌های ورودی و خروجی و برق استفاده می‌کنیم. همه پینهای LCD می‌بایست داخل این پایه‌های سیاه رنگ برد آردوینو فرو روند. چنانچه پینهای برد LCD شما کمی کج شده‌اند و به درستی در برد آردوینو فرو نمی‌روند ابتدا آنها را صاف کنید. برای اینکه جهت متصل کردن برد LCD را تشخیص دهید به نوشته‌های روی برد LCD توجه کنید. احتمالا تعدادی از پایه‌های روی LCD به نام A0 تا A5 نامگذاری شده‌اند بر روی برد آردوینو نیز تعدادی از پایه‌ها با نام A0 تا A5 نامگذاری شده‌اند. می‌بایست پایه‌های A0 تا A5 از LCD به پایه‌های A0 تا A5 از آردوینو متصل شوند.

آماده به کار کردن LCD

اکثر وسایل دیجیتال، لازم است که آماده به کار شوند. با استفاده از دستور lcd.begin می توانیم LCD را آماده به کار کنیم. ساختار این دستور، به شکل زیر است:

```
lcd.begin(lcd تعداد سطرهای ,lcd تعداد ستونهای);
```

در این دستور می بایست اعلام کنیم که LCD متصل به آردوینو چند سطر دارد و در هر سطر، چند کاراکتر جا می شود. (اصطلاحا چند ستون دارد.) به طور مثال، اگر می خواهیم از LCD که دارای ۲ سطر و ١٦ ستون است استفاده کنیم، می بایست دستور زیر را بنویسیم:

```
lcd.begin(16, 2);
```

نمایش دادن پیغام بر روی LCD با استفاده از دستور lcd.print

با استفاده از lcd.print() می توانیم بر روی ال سی دی بنویسیم. به طور مثال، دستور زیر پیغام Hello World! را بر روی LCD نمایش می دهد:

```
lcd.print("Hello World!");
```

همچنین دستور زیر، پیغام BIRTH:1/1/2000 را نمایش می دهد:

```
lcd.print("BIRTH:1/1/2000");
```

تنظیم کردن کنتراست LCD

هنگامی که برد LCD را خریداری می‌کنید ممکن است کنتراست آن تنظیم نشده باشد و هنگامی که برنامه فوق را می‌نویسید، نوشته‌ای روی LCD ظاهر نشود. بر روی برد LCD، مکعب مستطیل آبی رنگی وجود دارد که روی آن پیچ کوچکی قرار دارد. این مکعب مستطیل، پتانسیومتر مربوط به کنتراست است. با استفاده از پیچ گوشتی آن را چندین دور در جهت عقربه‌های ساعت بچرخانید. آن قدر ولوم را بچرخانید که یا نوشته‌ای ظاهر شود یا کل صفحه سفید شود. اگر نیمه بالای تصویر سفید شد و نیمه پایین آبی بود بدین معنی است که هنوز LCD آماده بکار نشده است که در این حالت سیم کشی ها را چک کنید و در برنامه مطمئن شوید که شماره پایه‌ها را درست وارد کرده باشید و دستور lcd.begin را صدا زده باشید. اما اگر کل صفحه سفید شد و قبل از آن هیچ نوشته‌ای ظاهر نشد، بدین معنی است که احتمالا LCD آماده بکار شده اما چیزی روی LCD نوشته نشده که در این صورت کمی ولوم را در خلاف جهت بچرخانید تا سفیدی صفحه برطرف شود و سپس به رفع عیب برنامه بپردازید.
اگر هفت دور در جهت ساعت چرخاندید و تصویری ظاهر نشد و صفحه نیز سفید نشد، ولوم را در خلاف جهت و این بار به مقداری بیش از آنچه ساعتگرد چرخانده‌اید در راستای پادساعتگرد بچرخانید تا نوشته ظاهر شود یا کل صفحه سفید شود یا نیمه بالای تصویر سفید شود که اگر هر یک از این حالات پیش آمد همانند فوق عمل کنید.

دستور lcd.setCursor

در کامپیوتر، هر موقع قرار باشد مطلبی را تایپ کنیم یک مکان نما (cursor) ظاهر می شود که محلی که می خواهیم در آن تایپ کنیم را با آن انتخاب می کنیم. به طور مثال در محیط برنامه نویسی آردوینو با کلیدهای بالا و پایین و چپ و راست مکان نما را به محلی که می خواهیم تایپ کنیم می بریم و سپس در آنجا تایپ می کنیم.

بر روی lcd های کاراکتری نیز ما یک مکان نما داریم. هنگامی که LCD روشن می شود این مکان نما بر روی اولین خانه سمت چپ از خط اول LCD قرار دارد و هر گاه نوشته ای را با استفاده از دستور lcd.print می نویسیم مکان نما به جلو می رود. به طور نمونه در مثال Hello World هنگامی که پیغام Hello World! را نمایش داده ایم، مکان نما پس از پیغام Hello World! قرار می گیرد و اگر پس از نمایش دادن Hello World!، دستور نمایش دادن پیغام دیگری را بدهیم، آن پیغام در جلوی Hello World! نمایش داده می شود:

```
lcd.print("Hello World!");
lcd.print("Hi");
```

به عبارت دیگر، نوشته هایی که بر روی LCD نمایش می دهیم پشت سر هم نمایش داده می شوند. مثلا اگر دستورات زیر را بنویسیم، بر روی LCD، پیغام Hello Arduino! بر روی LCD نمایش داده می‌شود:

```
  lcd.print("He");
  lcd.print("llo ");
  lcd.print("Ard");
  lcd.print("uino!");
```

برای اینکه مکان نما را بر روی LCD جابجا کنیم و در محل دیگری از LCD پیغامی را نمایش دهیم از دستور زیر استفاده می کنیم:

```
lcd.setCursor(ستون,سطر);
```

به طور مثال دستور زیر، مکان نما را به ستون دو از خط یک منتقل می کند:

```
lcd.setCursor(2,1);
```

بر روی LCD، شمارش سطرها و ستونها را از صفر شروع می‌کنند. یعنی اولین ستون از سمت چپ، ستون صفرم است و آخرین ستون سمت راست در LCDهای ۱۶ ستونه، ستون ۱۵ است. همچنین اولین سطر LCD، سطر صفرم است و خط دوم سطر یکم است. بنابراین دستور زیر، مکان نما را به ابتدای خط دوم می برد:

```
lcd.setCursor(0,1);
```

به طور مثال دستورات زیر پیغام How are you? را در خط دوم نمایش می دهد:

```
lcd.setCursor(0,1);
lcd.print("How are you?");
```

در زیر برنامه کاملی را می بینید که پیغام Hello World! را در خط اول و پیغام How are you? را در خط دوم نمایش می دهد:

برنامه ۸-۱: نمایش بر روی LCD

```
// include the library code
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);    // set the LCD's number of columns and rows
  lcd.print("Hello world!");   // display "Hello world!" on the LCD.
  lcd.setCursor(0,1);          // set the cursor position
  lcd.print("How are you?");   // display "How are you?"
}

void loop() {
}
```

پاک کردن LCD با استفاده از دستور lcd.clear()

هر گاه بخواهیم کلیه مطالبی که از قبل بر روی LCD نمایش داده شده است را پاک کنیم از دستور lcd.clear() استفاده می‌کنیم. دستور lcd.clear() علاوه بر اینکه نوشته‌های روی LCD را پاک می‌کند، مکان نما را نیز به ابتدای خط اول منتقل می کند. انگار که LCD تازه روشن شده باشد. به طور مثال، برنامه

زیر ابتدا بر روی LCD پیغام Hello را نمایش می دهد. سپس بعد از یک ثانیه، صفحه نمایش پاک می شود و پیغام ?How are you نمایش داده می شود:

```
lcd.print("Hello");
delay(1000);
lcd.clear();
lcd.print("How are you?");
```

نمایش دادن و پنهان کردن مکان نما

گاهی اوقات مایلیم که مکان نما بر روی LCD نمایش داده شود.

به طور مثال، هنگامی که کاربر قرار است مطلبی را تایپ کند می بایست که محل تایپ شدن را با استفاده از مکان نما به کاربر نشان دهیم. برای اینکه مکان نما بر روی LCD ظاهر شود، از دستور زیر استفاده می کنیم:

```
lcd.cursor();
```

اما زمانی که پیغام Hello را بر روی LCD نمایش می دهیم، قشنگ تر است که مکان نما بر روی LCD نمایش داده نشود. دستور زیر، باعث می شود که مکان نما پنهان شود:

```
lcd.noCursor();
```

ایجاد کردن کاراکترهایی با شکل جدید

با استفاده از دستور lcd.createChar می توانیم کاراکترهایی با شکل جدید، مثلا به شکل چهره خندان ایجاد کنیم. جهت اطلاعات بیشتر به راهنمای محیط برنامه نویسی آردوینو مراجعه کنید. برای رفتن به راهنمای آردوینو، در منوی Help بر روی Reference کلیک کنید و در صفحه ای که باز می شود بر روی libraries کلیک کنید و Liquid Crystal را انتخاب نمایید تا راهنمای مربوط به LCD به شما نشان داده شود.

تمرین

۱. برنامه ای بنویسید که اسم شما را بر روی LCD نمایش دهد.

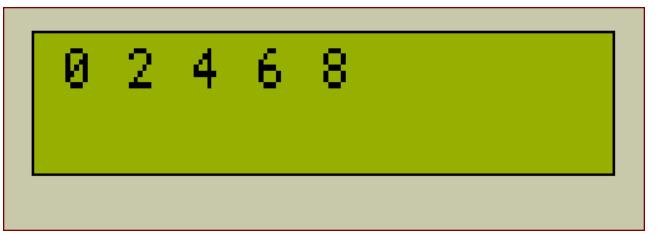
۲. برنامه ای بنویسید که بر روی LCD یک ثانیه شمار نمایش دهد. یعنی ابتدا بر روی نمایشگر عدد ۰ را نمایش دهد، سپس بعد از یک ثانیه بر روی نمایشگر عدد ۱ را نمایش دهد و همین طور هر یک ثانیه یکبار یکی به مقدار نمایش داده شده بر روی LCD افزوده شود. (راهنمایی:

یک متغیر تعریف کنید و مقدار آن را هر ثانیه یکی زیاد کنید و مقدار آن را بر روی نمایشگر نمایش دهید.)

۳. فرض کنید که کلیدی به پایه ۲ متصل است. برنامه ای بنویسید که وضعیت پایه ۲ را چک کند. اگر پایه ۲، پایین بود، بر روی نمایشگر پیغام Low را نمایش دهد و اگر بالا بود، پیغام High را نمایش دهد.

٤. برنامه ای بنویسید که اعداد ۱ تا ۵ را بر روی LCD نمایش دهد و در بین آنها کاما نمایش دهد.

٥. برنامه ای بنویسید که اعداد زوج کوچکتر از ۱۰ را در ستونهای زوج از اولین خط LCD نمایش دهد.



٦. برد آردوینو را به کامپیوتر متصل کنید. سپس برنامه ای بنویسید که هر کاراکتری که از کامپیوتر دریافت می کند را بر روی LCD در ابتدای اولین خط نمایش دهد.

۷. برد آردوینو را به کامپیوتر متصل کنید. سپس برنامه ای بنویسید که هر گاه از طریق سریال برای برد آردوینو کاراکتر H را ارسال کردیم، بر روی LCD، پیغام Hello نمایش داده شود و هر گاه کاراکتر B را ارسال کردیم، پیغام Bye را نمایش دهد.

۸. برنامه ای بنویسید که هر کاراکتری که از طریق سریال دریافت می کند را بر روی LCD نمایش دهد و هر گاه مکان نما (cursor) به انتهای خط رسید، مکان نما به ابتدای خط برگردد.
(راهنمایی: یک متغیر از جنس int تعریف کنید و آن را با صفر مقداردهی اولیه کنید. قبل از نمایش دادن هر کاراکتر، موقعیت مکان نما را بر حسب مقدار این متغیر جابجا کنید و کاراکتر مورد نظر را نمایش می دهید. سپس مقدار این متغیر را یکی زیاد کنید. اگر مقدار این متغیر به ۲۰ رسید می دانید که به انتهای خط رسیده اید و بنابراین متغیر را با صفر مقداردهی می کنید.)

فصل ۹: تابع (Function)

بخش ۹-۱: ایجاد کردن توابع جدید

شاید در ایام کودکی به کنار دریا رفته باشید و با شن و ماسه چاه آب یا قلعه کوچکی درست کرده باشید. درست کردن وسایل کوچک با استفاده از شن و ماسه راحت است. اما به سختی بتوان با آنها خانه ای ساخت. بلکه معمولا ابتدا با استفاده از خاک، بلوک های آجری یا سیمانی را آماده می کنند و سپس با کنار هم گذاشتن آجرها، دیوارها و خانه ها را بنا می کنند.

در برنامه نویسی نیز بسیار مرسوم است که مجموعه ای از دستورات را در کنار هم قرار می دهند و بلوک هایی را به وجود می آورند که کار مشخصی را انجام می دهد. و سپس با استفاده از این بلوک ها یک برنامه را بنا می نهند. اصطلاحا به مجموعه ای از دستورات که کار مشخصی را انجام می دهند تابع می گویند. در فصل های قبل شما با توابع مختلفی آشنا شده اید. مثلا تابع println() تابعی بود که پیغامی را از طریق سریال ارسال می کرد. همچنین تابع gotoxy() تابعی بود که محل مکان نما را روی LCD جا به جا می کرد. در عین حال delay() تابعی بود که تاخیری ایجاد می کرد. بعضی از توابع قبلا توسط دیگران نوشته شده اند. ما در این فصل فرا می گیریم که خودمان توابعی را بنویسیم. حسن ایجاد کردن تابع این است که یکبار تابعی را می نویسیم و سپس در برنامه، هر کجا که مایل بودیم آن را صدا می زنیم. مثلا تابع println() یکبار قبلا نوشته شده است و ما در برنامه هر کجا که مایل باشیم آن را صدا می زنیم. در زیر ما تابعی به نام printLine تعریف می کنیم که با استفاده از ستاره، یک خط پر از ستاره رسم می‌کند.

برنامه ۹-۱

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.println("Hello");
    printLine();
    Serial.println("It is a nice day.");
    delay(1000);
}

void printLine()
{
    for(int i=0; i<30; i++)
    {
        Serial.print("*");
```

```
    }
    Serial.println("");
}
```

همانطور که در برنامه فوق می بینید تابع printLine() با ساختار زیر تعریف شده است. یعنی خط اول اسم تابع ذکر شده است و در بین } و { مجموعه دستورات تابع ذکر شده است. اصطلاحا به علامتهای { و }، کروشه باز و کروشه بسته گفته می شود.

```
void اسم تابع ()
{
    مجموعه دستورات
}
```

در برنامه مثال ۹-۱ مثل همیشه ابتدا اجرا از setup آغاز می شود که در آن سرعت سریال مشخص شده است. سپس loop دائما اجرا می شود که در آن صدا زدن تابع println("Hello") باعث ارسال Hello از طریق سریال می شود و سپس صدا زدن تابع printLine() باعث می شود که دستورات داخل تابع printLine یکی بعد از دیگری اجرا شوند و یک خط حاوی ۳۰ تا ستاره نمایش داده شود. پس از اتمام تابع، اجرا به محلی که تابع را صدا زده ایم بر می گردد و دستوری که بعد از صدا زدن تابع قرار دارد اجرا می شود. در حقیقت صدا زدن تابع درست مانند اجرا شدن سایر دستورات برنامه نویسی است و زمانی که تابعی را صدا می زنیم، پس از اجرا شدن تابع، دستوری که بعد از صدا زدن تابع قرار دارد، اجرا می شود.

در این برنامه در تابع loop ابتدا پیغام Hello به کامپیوتر ارسال می شود. سپس یک خط پر ستاره ارسال شده و سپس پیغام It is a nice day ارسال می گردد و در نهایت یک ثانیه صبر می کند.

```
Hello
******************************
It is a nice day
```

ارسال مقادیر به تابع و باز گرداندن مقادیر از داخل تابع

تابع printLine همیشه یک خط که حاوی ۳۰ ستاره است را ارسال می کند. اما ممکن است ما مایل باشیم که تعداد ستاره هایی که نمایش داده می شود کمتر یا بیشتر از ۳۰ ستاره باشد. بنابراین این امکان فراهم شده است که بتوانیم مقداری را به داخل توابع بفرستیم و یا از توابع مقداری را باز گردانیم.

در برنامه ۹-۲، تابع printLine را به گونه ای بازنویسی می کنیم تا بتوانیم تعداد ستاره های ارسالی را مشخص کنیم.

برنامه ۹-۲

```
void setup()
{
	Serial.begin(9600);
}

void loop()
{
	Serial.println("Hello");
	printLine(5);
	Serial.println("It is a nice day.");
	printLine(15);
	delay(1000);
}

void printLine(int numOfStricks)
{
	for(int i = 0; i < numOfStricks; i = i + 1)
	{
		Serial.print("*");
	}
	Serial.println("");
}
```

ساختار کلی تعریف کردن توابع به صورت زیر است:

(مقادیری که برنامه به تابع می فرستد و نوع آنها) ***نام تابع*** *مقدار برگشتی تابع*
{
}

همانطور که در مثال فوق می بینید زمان صدا زدن تابع printLine در داخل پرانتز اعلام کردیم که چه تعداد ستاره را نمایش دهد. در ابتدا گفتیم که پیغام Hello را نمایش دهد. سپس خطی با ۵ ستاره ترسیم کند. سپس پیغام It is a nice day را ارسال می کند. سپس خطی با ۱۵ ستاره رسم می کند.

Hello

*****

It is a nice day

****

در تابع println که قبلا با آن آشنا شده ایم، پیغامی که می خواهیم نمایش دهد را به عنوان مقدار ورودی به آن می دهیم:

```
Serial.println("Our message");
```

همچنین به تابع setCursor، مختصات نقطه ای که مایلیم مکان نما به آنجا برود را می دهیم:

```
lcd.setCursor(2,1);
```

در تابع delay نیز مدت زمانی را که می خواهیم صبر کند را به عنوان مقدار ورودی به تابع می دهیم. در اصطلاح برنامه نویسی به مقداری که به داخل تابع ارسال می شود اصطلاحا آرگومان (argument) می‌گوییم. به طور مثال در تابع printLine یک آرگومان از جنس int دارد که بیانگر تعداد ستاره هایی است که تابع نمایش می دهد. توابع می توانند بیش از یک آرگومان نیز دریافت کنند. مثلا تابع setCursor دو تا آرگومان دریافت می کند. آرگومان اول شماره ستون و آرگومان دوم شماره سطری است که می بایست به آن منتقل شود. ما نیز ممکن است تمایل داشته باشیم بتوانیم کاراکتر هایی را که تابع printLine نمایش می دهد را مشخص کنیم. برای مثال گاهی می خواهیم خطی با + و یا – بکشیم. برای این کار تابع printLine را اینگونه بازنویسی می کنیم:

برنامه ۹-۳

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.println("Hello");
    printLine('+' ,5);
    Serial.println("It is a nice day.");
    printLine('-', 15);

    delay(2000);
}

void printLine(char c, int numOfStricks)
{
    for(int i = 0; i < numOfStricks; i = i + 1)
    {
        Serial.print(c);
    }
    Serial.println("");
}
```

مقدار برگشتی

توابعی که تاکنون نوشته ایم هیچ مقداری را از خود برنمی‌گردانند. به همین خاطر، قسمت بازگشتی تابع void اعلام شده است. اما گاهی ممکن است لازم باشد مقداری را از درون تابع به محلی که تابع را صدا زده ایم برگردانیم. به طور مثال در برنامه زیر تابعی به نام square تعریف شده است که آرگومان x را به عنوان ورودی دریافت می کند. مقدار x^2 را محاسبه می کند و مقدار محاسبه شده را به محلی که تابع صدا زده شده است باز می گرداند. با استفاده از دستور return مقداری که مایلیم تابع بازگرداند را اعلام می کنیم.

برنامه ۹-۴

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    a=square(5);
    Serial.print("5 to the power of 2 = ");
    Serial.println(a);
    delay(2000);

}

int square(int x)
{
    return (x*x);
}
```

در برنامه فوق در داخل loop دستور "a=square(5);" را نوشته ایم. بنابراین مقدار ۵ به تابع square داده می شود. در داخل تابع square، مقدار x که برابر ۵ است در خودش ضرب می شود که حاصل، ۲۵ می شود. سپس دستور return مقدار ۲۵ را باز می گرداند. با توجه به اینکه در داخل loop متغیر a را برابر با square(5) قرار داده ایم (نوشته ایم a=square(5);) بنابراین مقدار ۲۵ که مقدار برگشتی تابع square است به داخل متغیر a ریخته می شود و a برابر ۲۵ می شود. بنابراین برنامه فوق پیغام زیر را نمایش می‌دهد:

5 to the power of 2 = 25

توجه داشته باشید که به مجرد اینکه دستور return اجرا شود، از تابع خارج می شود و بنابراین اگر بعد از دستور return دستورات دیگری وجود داشته باشند، اجرا نمی شوند.

برای ممارست هر چه بیشتر با مفهوم تابع، در زیر به نوشتن چندین برنامه می پردازیم.

مثال ۹-۱

تابعی بنویسید که عددی را به عنوان آرگومان ورودی دریافت کند. سپس حاصل جمع اعداد ۰ تا آن عدد را به عنوان مقدار برگشتی برگرداند. مثلا اگر عدد ۵ را به تابع بدهیم حاصل جمع 1+2+3+4+5 که برابر با ۱۵ است را برگرداند.

پاسخ:

برای اینکه حاصل جمع اعداد ۰ تا آن عدد را محاسبه کنیم، کافی است که یک حلقه for بنویسیم که از ۰ تا آن عدد برود و این اعداد را با هم جمع بزنیم. بنابراین تابع به صورت زیر می شود:

```
int sumValues(int number)
{
    int sum = 0;
    int i;

    for(i = 0; i <= number; i++)
    {
        sum = sum + i;
    }

    return sum;
}
```

مثال ۹-۲

تابعی بنویسید که دو عدد به عنوان آرگومان ورودی دریافت کند. سپس از بین این دو عدد هر یک که بزرگ تر است را به عنوان مقدار برگشتی برگرداند.

پاسخ:

برای اینکه حاصل جمع اعداد ۰ تا آن عدد را محاسبه کنیم، کافی است که یک حلقه for بنویسیم که از ۰ تا آن عدد برود و این اعداد را با هم جمع بزنیم. بنابراین تابع به صورت زیر می شود:

```
int findMax(int a, int b)
{
    if(a > b)
    {
        return a;
```

```
    }
    else
    {
        return b;
    }
}
```

توضیح: در محیط آردوینو، تابع max از قبل تعریف شده است که دو مقدار را دریافت می کند و مقدار عدد بزرگتر را بر می گرداند و هدف ما از نوشتن این برنامه صرفا تمرین کردن نوشتن تابع بود. درون محیط آردوینو تابع min نیز از قبل تعریف شده است که مقدار عدد کوچک تر را بر می گرداند.

مثال ۹-۳

تابعی بنویسید که عددی را به عنوان آرگومان دریافت کند و مقداری از جنس int برگرداند. اگر این عدد زوج بود، مقدار ۱ را برگرداند و در غیر این صورت مقدار 0 را بازگرداند.

پاسخ:

به عددی زوج می گویند که آن عدد بر ۲ بخش پذیر باشد؛ یعنی باقی مانده تقسیم آن عدد بر ۲، برابر با صفر باشد. بنابراین تابع به صورت زیر می شود:

```
int isEven(int n)
{
    if((n % 2) == 0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

مثال ۹-۴

تابعی بنویسید که حرفی را به عنوان آرگومان دریافت کند. سپس اگر این حرف از حروف کوچک انگلیسی بود (a تا z بود)، حرف بزرگ انگلیسی متناظر با آن را بازگرداند. در غیر این صورت خود حرف را بازگرداند.

پاسخ:

اگر به جدول کدهای اسکی که در ابتدای فصل ۷ وجود دارد، مراجعه کنیم، حروف کوچک انگلیسی (a تا z) به ترتیب دارای کد اسکی ۹۷ تا ۱۲۲ هستند. همچنین هر حرف بزرگ انگلیسی با حرف کوچک متناظر با آن ۳۲ تا فاصله دارد. مثلا حرف A دارای کد اسکی ۶۵ است و حرف a دارای کد اسکی ۹۷ است که با هم ۳۲ تا فاصله دارند. بنابراین اگر بخواهیم حرف بزرگ متناظر با حروف کوچک را به دست آوریم کافی است که از آن ۳۲ تا کم کنیم. (مثلا اگر از ۹۷ که کد اسکی a است ۳۲ تا کم کنیم عدد ۶۵ به دست می‌آید که کدِ اسکیِ حرفِ A است.) بنابراین تابع به صورت زیر می شود:

```
char toCapital(char c)
{
    if((c >= 97)&&(c <= 122)) //is it a small letter?
    {
        return c - 32; //make it capital letter
    }
    else
    {
        return c;
    }
}
```

توضیح: در برنامه فوق به جای اعداد ۹۵ و ۱۲۲ که کد اسکی حروف a و z هستند می توانیم از خود این حروف نیز استفاده کنیم و دستور if را به صورت زیر بنویسیم:

```
    if((c >= 'a')&&(c <= 'z')) //is it a small letter?
```

مثال ۹-۵: عدد اول

تابعی بنویسید که عددی را به عنوان آرگومان دریافت کند. سپس چک کند که آیا آن عدد اول است یا نه. اگر اول بود مقدار ۱ را برگرداند و در غیر این صورت مقدار ۰ را برگرداند.

پاسخ:

عددی اول است که جز بر خودش و یک بر هیچ عدد کوچکتر از آن عدد بخش پذیر نباشد. بنابراین برای اینکه چک کنیم که آیا عددی اول است یا نه یک حلقه for می نویسیم که از ۲ تا آن عدد بشمارد. اگر بر هر یک از اعداد بخش پذیر بود، مشخص می شود که عدد اول نبوده و بنابراین مقدار ۰ را برمی گردانیم. اما اگر حلقه for تا انتها رسید و بر هیچیک از اعداد بخش پذیر نبود، عدد اول بوده است.

```
int isPrime(int number)
{
    int i;

    for(i = 2; i < number; i++)
    {
        if((number%i) == 0) //is it divisible?
        {
            return 0; //Not prime
        }
```

```
    }

    return 1; //prime
}
```

توضیح: به مجرد اینکه دستور return اجرا شود، از تابع خارج می شود. بنابراین در برنامه فوق اگر متغیر number بر هر یک از اعداد بخش پذیر باشد، بلافاصله از داخل تابع بیرون می آید و حلقه for ادامه پیدا نمی کند. بنابراین صرفا زمانی حلقه for به پایان می رسد و دستور return 1 اجرا می شود که عدد بر هیچ یک از اعداد کوچکتر از number بخش پذیر نبوده باشد.

بازگشت از توابعی که مقدار برگشتی ندارند

در فوق ذکر شد که تابع return علاوه بر برگرداندن مقدار برگشتی، سبب می شود که اجرا شدن تابع به اتمام برسد. بنابراین دستور return درون توابعی که مقدار برگشتی ندارند نیز کاربرد دارند و در مواقعی که بخواهیم اجرای تابعی را به پایان برسانیم بدون اینکه مقداری برگردانیم نیز می توانیم از دستور return استفاده کنیم. دستور زیر باعث بازگشت از تابع می شود بدون آنکه مقداری را برگرداند:

```
return;
```

توجه داشته باشید که در توابعی که مقدار برگشتی دارند، حتما مقدار برگشتی را در جلوی دستور return اعلام کنید. زیرا اگر مقدار برگشتی را اعلام نکنید، کامپایلر به شما error نمی دهد؛ اما با توجه به اینکه مقدار درستی بازگشت داده نمی شود، کد نوشته شده به درستی کار نمی کند.

بخش ۹-۲: دسترسی به متغیرها (scope)

متغیرهایی که تاکنون تعریف کردیم در درون توابع بودند. هنگامی که متغیری را درون توابعی تعریف می کنیم، زمانی که اجرا شدن برنامه به نقطه ای که متغیر را تعریف کرده ایم می رسد، فضایی از حافظه به متغیر اختصاص می یابد و هنگامی که به انتهای تابع می رسیم، حافظه ای که به متغیر اختصاص یافته است، آزاد می شود تا فضای حافظه بعدا قابل استفاده باشد. بنابراین متغیرهایی که درون توابع تعریف می‌کنیم فقط درون آن تابع قابل دسترس است.

گاهی اوقات لازم است که مقدار متغیری در تمام مدتی که برنامه اجرا می شود، محفوظ بماند و یا لازم است که در توابع متعددی به آن دسترسی داشته باشیم. در چنین مواقعی می توانیم متغیرها را در بیرون از توابع تعریف کنیم. هنگامی که متغیری در بیرون از توابع تعریف شود، ابتدای اجرا شدن برنامه قسمتی از حافظه به آن اختصاص می یابد و تا زمانی که برنامه اجرا شود، این فضا محفوظ می ماند. (در کامپیوترها،

اگر برنامه ای را ببندیم اجرای آن به پایان می رسد. اما در آردوینو پایان یافتن اجرای برنامه بدون معنی است. بلکه تا مادامی که آردوینو روشن باشد اجرا شدن برنامه ادامه می یابد.)

اصطلاحا به متغیرهایی که بیرون از توابع تعریف شوند، متغیرهای عمومی (Global) و به متغیرهایی که درون توابع تعریف شوند، متغیرهای محلی (Local) می گویند.

در مثالهای زیر برنامه هایی را با استفاده از متغیر عمومی نوشته ایم.

مثال ۹-۶: ثانیه شمار

می خواهیم با استفاده از آردوینو یک ثانیه شمار بسازیم که مدت زمانی که از روشن شدن آردوینو می گذرد را بر حسب ثانیه از طریق سریال اعلام کند. یک متغیر به نام n از جنس int تعریف کنید و با مقدار ۰ مقداردهی اولیه کنید. سپس هر ثانیه یکبار، مقدار n را یکی زیاد کنید و مقدار آن را از طریق سریال به کامپیوتر بفرستید.

پاسخ:

برای اینکه هر ثانیه یکبار مقدار n را زیاد کنیم، تاخیری به اندازه یک ثانیه در داخل loop می گذاریم و پس از ۱ ثانیه مقدار آن را زیاد می کنیم. یعنی به صورت زیر:

```
void setup()
{
    Serial.begin(9600);
}

int n = 0;
void loop()
{
    Serial.println(n);
    n = n + 1;
    delay(1000);
}
```

توضیح: در برنامه فوق، اگر دستور int n = 0; را در داخل loop می گذاشتیم، هر بار که تابع loop به اتمام می رسید حافظه مربوط به متغیر n آزاد می شد و هنگامی که به ابتدای loop می رفت فضایی را برای متغیر n اشغال می کرد و مقدار متغیر n را با صفر مقداردهی می کرد. بنابراین، متغیر n را بیرون از loop تعریف کرده ایم تا هنگامی که تابع loop به پایان می‌رسد حافظه مربوط به متغیر n آزاد نشود و مقدار متغیر n محفوظ بماند.

زمان سپری شده همیشه عددی مثبت است. بنابراین در برنامه فوق برای اینکه متغیر n اعداد بزرگتری را در خود جای دهد، خوبست که آن را از جنس unsigned int تعریف کنیم و برنامه را به صورت زیر بنویسیم:

```
void setup() {
```

```
        Serial.begin(9600);
}

unsigned int n = 0;
void loop() {
        Serial.println(n);
        n = n + 1;
        delay(1000);
}
```

مثال ۹-۷

برنامه ای که در مثال ۹-۶ نوشته ایم را به گونه ای تغییر دهید که با دقت هزارم ثانیه مدت زمانی که از روشن شدن آردوینو می گذرد را از طریق سریال اعلام کند. یک متغیر به نام n از جنس int تعریف کنید و با مقدار ۰ مقداردهی اولیه کنید. سپس هر ثانیه یکبار، مقدار n را یکی زیاد کنید و مقدار آن را از طریق سریال به کامپیوتر بفرستید.

پاسخ:

برای اینکه هر هزارم ثانیه یکبار مقدار n را زیاد کنیم، تاخیری به اندازه یک هزارم ثانیه در داخل loop می گذاریم و پس از ۱ ثانیه مقدار آن را زیاد می کنیم. البته با توجه به اینکه با دقت هزارم ثانیه در حال شمارش است، برای اینکه بتواند تعداد ارقام بیشتری را در خود جای دهد، خوبست که متغیر را از جنس unsigned long تعریف کنیم. یعنی به صورت زیر:

```
void setup() {
    Serial.begin(9600);
}

unsigned long n = 0;
void loop() {
    Serial.println(n);
    n = n + 1;
    delay(1);
}
```

اگر برنامه مثال ۹-۷ را روی برد آردوینو بریزید و همزمان یک کورنومتر را روشن کنید می بینید که آردوینو کمی آهسته تر از کورنومتر می شمارد و پس از مدتی از کورنومتر عقب می ماند. در برنامه فوق فرض کرده ایم که هر یک هزارم ثانیه یکبار مقدار متغیر n یکی زیاد می شود. در حالی که در واقعیت چنین نیست. از زمانی که مقدار n را زیاد می کنیم تا دفعه بعدی که مقدار n را زیاد می کنیم، اتفاقات زیر می افتد: تابع delay صدا زده می شود، از طریق سریال مقدار n به کامپیوتر ارسال می شود و مقدار n

یکی زیاد می شود. صدا زدن تابع delay(1) باعث ایجاد شدن تاخیری در حدود یک میلی ثانیه می شود. اما اجرا شدن دستورات println و زیاد کردن n نیز زمان می برد که به میزان تاخیری که تابع delay(1) ایجاد می کند افزوده می شود. بنابراین از زمانی که n یکی زیاد می شود تا دفعه بعدی که n یکی زیاد می‌شود کمی بیشتر از یک هزارم ثانیه طول می‌کشد و سرعت شمارش برنامه فوق آهسته تر از کورنومتر می‌شود. در فصل ۱۳ با توابع micros و millis آشنا می شوید که با استفاده از آنها می توان، مدت زمانی که آردوینو روشن شده است را با دقت بالا اندازه گرفت. با استفاده از توابع micros و millis می‌توانیم برنامه فوق را بازنویسی کنیم به گونه ای که عدم دقت پیش آمده در برنامه فوق به طور کامل برطرف شود.

بخش ۹-۳: متغیرهایی از نوع bool

در فصل۵ با برخی از انواع متغیرهایی که در زبان سی وجود دارد (مثل char، int و long) آشنا شدید. در این بخش با متغیرهایی از نوع bool آشنا می‌شوید. متغیرهای نوع bool، می‌توانند یک بیت را داخل خود ذخیره کنند و بنابراین دو حالت می‌توانند داشته باشند: true و false.

به طور مثال در زیر متغیر result را از جنس bool تعریف می‌کنیم و مقدار false را درون آن می‌ریزیم:

```
bool result = false;
```

در تکه برنامه زیر چک می‌کنیم که اگر مقدار a بزرگتر از 5 بود، درون result مقدار true و در غیر این صورت مقدار false را می‌ریزیم:

```
bool result;
if(a > 5)
{
    result = true;
}
else
{
    result = false;
}
```

همچنین می‌توانیم نتیجه عبارات شرطی را مستقیما درون متغیرهای bool بریزیم. به طور مثال، در زیر اگر عبارت شرطی درست باشد (اگر مقدار a بزرگتر از 20 باشد) درون متغیر b مقدار true و در غیر این صورت مقدار false را می‌ریزد:

```
bool b;
b = a > 20;
```

در دستورات if و while و سایر دستوراتی که حاوی عبارات شرطی هستند، می‌توانیم به جای عبارت شرطی، متغیری از جنس bool را قرار دهیم. مثلا در تکه برنامه زیر چک می‌کنیم که اگر مقدار متغیر b برابر با true بود، کلمه yes و در غیر این صورت کلمه no را نمایش می‌دهد:

```
if(b)
{
    Serial.println("Yes!");
}
else
{
    Serial.println("No!");
}
```

نوع bool را می‌توانیم، همانند سایر متغیرها، به عنوان مقدار برگشتی تابع و یا آرگومان تابع استفاده کنیم. به طور نمونه در مثال زیر تابع isEven را با استفاده از نوع bool بازنویسی کرده ایم.

مثال ۹-۸

تابع isEven که در مثال ۹-۳ نوشته‌ایم را بازنویسی کنید به گونه‌ای که عددی را به عنوان آرگومان دریافت کند و مقداری از جنس bool برگرداند. اگر عددی که به عنوان آرگومان دریافت کرده زوج بود، مقدار true را برگرداند و در غیر این صورت مقدار false را بازگرداند.

پاسخ:

```
bool isEven(int n)
{
    if((n % 2) == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

مثال ۹-۹

با استفاده از تابع isEven که در مثال قبل نوشته شده، برنامه‌ای بنویسید که از ۱ تا ۱۰۰ بشمارد و اعداد را از طریق سریال به کامپیوتر بفرستد و جلوی هر یک از اعداد که زوج است کلمه (yes) را نمایش دهد.

پاسخ:

```
void setup()
{
    Serial.begin(9600);
}
int loop()
{
    int i;
    for(i = 0; i <= 100; i = i + 1)
    {
        Serial.print(i);
        if(isEven(i)) //is it even?
        {
            Serial.print("(Yes)");
        }
        Serial.print(",");
    }
}
```

تمرین

۱. تابعی بنویسید که عددی را به عنوان آرگومان دریافت کند. سپس مجموع اعداد فرد کوچکتر از این عدد را محاسبه کند و به عنوان مقدار برگشتی برگرداند.

۲. تابعی بنویسید که دو عدد را به عنوان آرگومان دریافت کند و میانگین آنها را به عنوان مقدار برگشتی برگرداند.

۳. تابعی بنویسید که سه عدد را به عنوان آرگومان دریافت کند و مقدار بزرگ ترین عدد را برگرداند.

٤. تابعی بنویسید که عددی را به عنوان آرگومان دریافت کند. سپس قدر مطلق آن عدد را حساب کند و به عنوان مقدار برگشتی برگرداند. (راهنمایی: برای اینکه قدر مطلق عددی را حساب کنیم، کافی است چک کنیم که آیا آن عدد کوچکتر از ۰ است یا نه. اگر کوچکتر از صفر بود، آن عدد را در 1- ضرب می کنیم تا به عددی مثبت تبدیل شود و در غیر این صورت خود آن عدد را باز می گردانیم.)

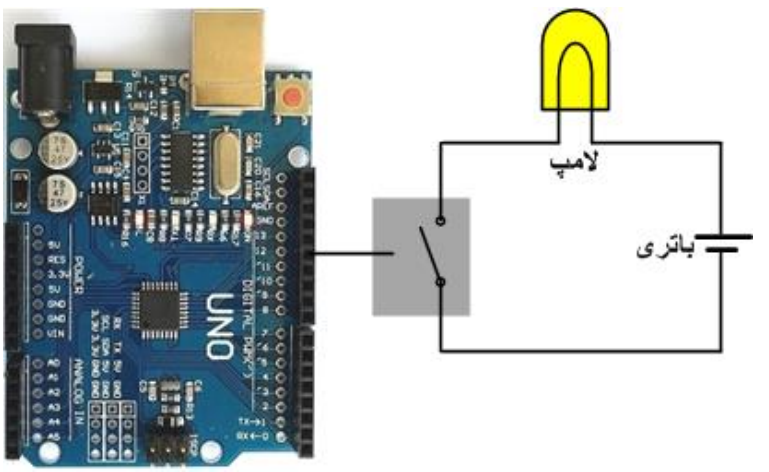
فصل ۱۰: آشنایی با ترانزیستور و رله

در این فصل فرا می گیریم که وسایل الکتریکی را به برد آردوینو متصل کنیم و از طریق برنامه هایی که برای برد آردوینو می نویسیم این وسایل الکتریکی را روشن و خاموش کنیم.

بخش ۱۰-۱: آشنایی با ترانزیستور

ولتاژ پایه های خروجی آردوینو ۵ ولت است و جریانی که می توانند فراهم کنند 0.02 آمپر است. اما بسیاری از وسایل برای اینکه کار کنند به جریانهای بیشتری نیاز دارند و ممکن است به ولتاژ بیشتری نیاز داشته باشند. به طور مثال ما اگر بخواهیم با آردوینو به یک لامپ ۲۲۰ ولت و یا یک کولر دستور بدهیم که روشن و خاموش شود به چه طریقی می توانیم این کار را انجام دهیم؟

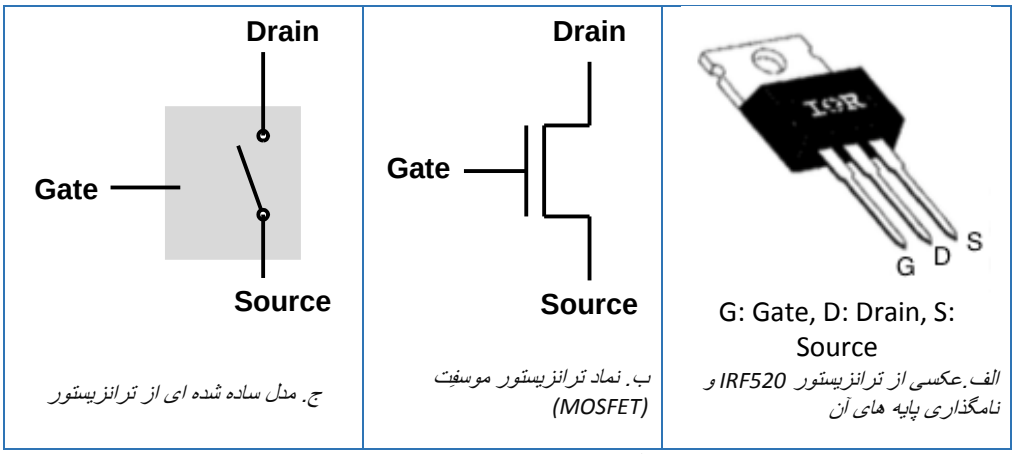
در فصل ٤، در شکل ٤-٥ دیدیم که با استفاده از کلید می توانیم وسایل برقی را روشن و خاموش کنیم. حال ما نیاز به وسیله ای شبیه کلید داریم که به پایه آردوینو نگاه کند و هرگاه ولتاژ پایه آردوینو ۵ ولت بود کلید را ببندد و هرگاه پایه آردوینو ۰ ولت بود کلید را باز کند. عملا رله و ترانزیستور چنین کاری انجام می دهند. شکل ۱۰-۱ را ببینید.



شکل ۱۰-۱: روشن و خاموش کردن یک لامپ

ترانزیستور

ترانزیستور یک عنصر سه سر است که عکس و نماد آن را در شکل ۱۰-۲ می بینید. ترانزیستور سه پایه به نامهای درین (drain)، سُورس (source) و گِیت (gate) دارد. هرگاه پایه گیت را به ۵ ولت وصل کنیم، پایه درین را به سورس وصل می کند و هرگاه ولتاژ پایه گیت ۰ ولت باشد، پایه درین را از سورس جدا می کند. بنابراین ترانزیستور را می توانیم به صورت شکل ۱۰-۲-ج مدل کنیم.



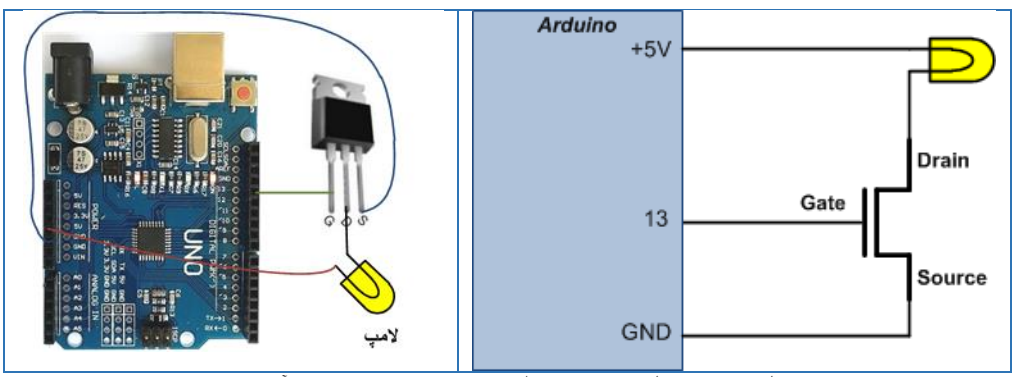
الف. عکسی از ترانزیستور IRF520 و نامگذاری پایه های آن

ب. نماد ترانزیستور موسفِت (MOSFET)

ج. مدل ساده شده ای از ترانزیستور

شکل ۱۰-۲: ترانزیستور (Transistor)

به عبارت دیگر، هر گاه ولتاژ پایه گیت، ۵ ولت بیشتر از ولتاژ پایه سورس باشد، کلید داخل ترانزیستور بسته می شود و هر گاه ولتاژ پایه گیت با ولتاژ پایه سورس برابر باشند کلید داخل ترانزیستور باز است. در مدار شکل ۱۰-۳ یک لامپ ۵ ولت را با استفاده از ترانزیستور به آردوینو متصل کرده ایم. در نظر داشته باشید که در این مدار می توانیم به جای پایه ۱۳ از هر یک از پایه های برد آردوینو که مایل باشیم استفاده کنیم. مثال ۱۰-۵ را ببینید.



شکل ۱۰-۳: اتصال یک لامپ ۵ ولت با استفاده از ترانزیستور به آردوینو

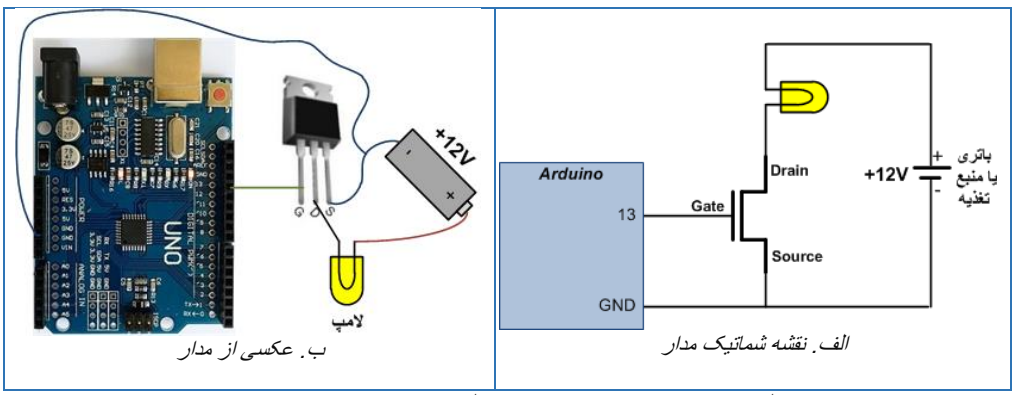
مثال ۱۰-۱

برای مدار شکل ۱۰-۳ برنامه ای بنویسید که چراغ را هر ۲ ثانیه یکبار روشن و خاموش کند.

پاسخ:

```
void setup() {
    pinMode(13,OUTPUT); //13 as output
}
void loop() {
    digitalWrite(13,HIGH);
    delay(2000); //wait 2 seconds
    digitalWrite(13,LOW);
    delay(2000); //wait 2 seconds
}
```

چنانچه ولتاژ کاری یک وسیله برقی بیشتر از ۵ ولت باشد و یا جریان مورد نیاز آن زیاد باشد، لازم است که یک منبع تغذیه جداگانه درون مدار قرار دهیم که شکل آن به صورت زیر می شود. مدار زیر مشابه مدار ۱۰-۳ است. توجه داشته باشید که در مدار زیر، پایه منفی از منبع تغذیه به پایه GND از برد آردوینو متصل شده است. به عبارت دیگر، پایه سورس از ترانزیستور هم به پایه GND از آردوینو متصل است و هم به سر منفی از منبع تغذیه.



الف. نقشه شماتیک مدار — ب. عکسی از مدار

شکل ۱۰-۴: روشن و خاموش کردن وسایل برقی با استفاده از ترانزیستور

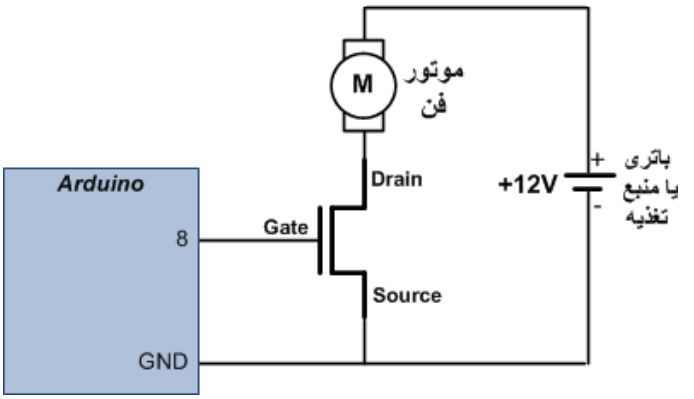
مشابه مدار فوق را می توانیم برای روشن کردن وسایل برقی دیگر استفاده کنیم. مثلا در مثال ۱۰-۲ مدار فوق را به گونه ای تغییر داده ایم که یک فن ۱۲ ولت را روشن و خاموش کند.

مثال ۱۰-۲

یک فن ۱۲ ولت داریم می خواهیم که در طی مدت ۳ دقیقه، به مدت ۲ دقیقه روشن و ۱ دقیقه خاموش باشد. مدار لازم را با آردوینو طراحی کنید و برنامه مربوطه را بنویسید.

پاسخ:

مدار مورد نیاز، همانند مدار شکل ۱۰-۴ است. فقط کافی است که به جای لامپ، فن بگذاریم.



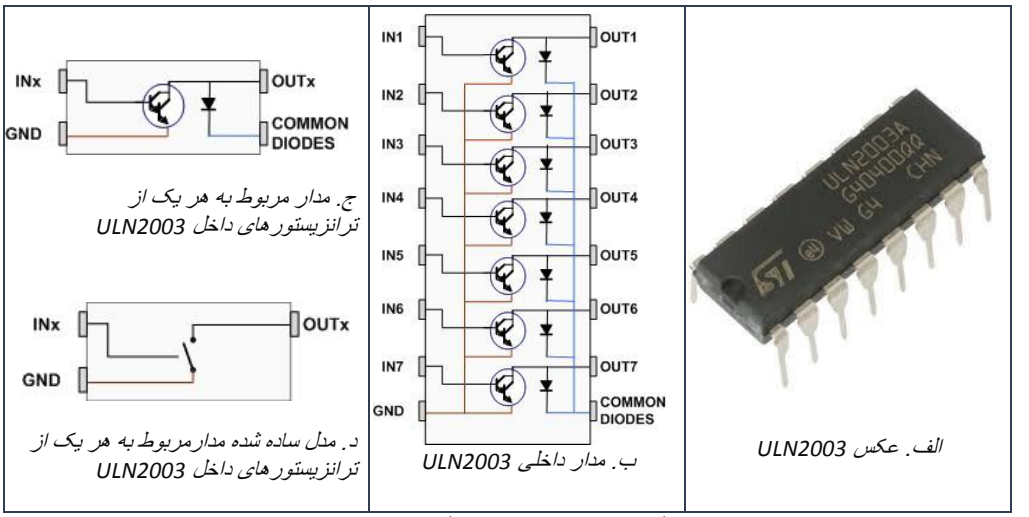

```
void setup() {
    pinMode(8,OUTPUT); //8 as output
}

void loop() {
    digitalWrite(8,HIGH);
    delay(120000); //wait 2 minutes
    digitalWrite(8,LOW);
    delay(60000);     //wait 1 min
}
```

در بازار ترانزیستورهای بسیار متنوعی وجود دارند که هر یک از آنها مزایا و خصوصیات منحصر به فردی دارند. از جمله ترانزیستورهای رایج، ترانزیستورهای خانواده IRF هستند و شما می توانید IRF520، IRF530، IRF540 و IRF750 را به سهولت از فروشگاه های داخل ایران تهیه کنید. ترانزیستورهای فوق بسیار شبیه یکدیگر هستند و هر یک از آنها را که مایل باشید می توانید استفاده کنید.

عکس یک ترانزیستور IRF520 را به همراه اسامی پایه های آن در شکل ۱۰-۲ می بینید. پایه ای که با G مشخص شده است، پایه گیت است، پایه ای که با حرف D مشخص شده، پایه درین است و پایه ای که با حرف S مشخص شده، پایه سورس است.

مزیت ترانزیستورهای IRF این است که جریان زیادی را می توانند از خود عبور دهند و بنابراین برای روشن و خاموش کردن اکثر وسایل برقی می توانید از آن استفاده کنید.

از دیگر ترانزیستورهای موجود در بازار ULN2003 است که ۷ ترانزیستور در دل یک آی سی جای گرفته اند و بنابراین با استفاده از آن می توانید تا ۷ وسیله برقی را روشن و خاموش کنید. در زیر عکس یک ULN2003 را به همراه مدار داخلی آن می بینید.



ج. مدار مربوط به هر یک از ترانزیستورهای داخل ULN2003

د. مدل ساده شده مدارمربوط به هر یک از ترانزیستورهای داخل ULN2003

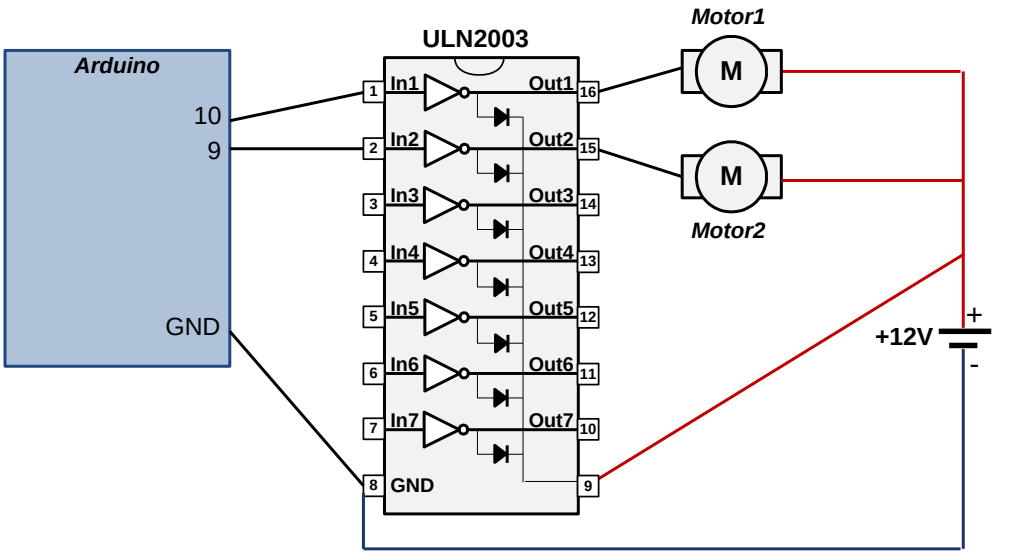
ب. مدار داخلی ULN2003

الف. عکس ULN2003

شکل ۱۰-۵: عکس و مدار داخلی ULN2003

در شکل ۱۰–۶ دو تا موتور با استفاده از یک عدد ULN2003 به برد آردوینو متصل شده اند و در مثال ۱۰–۳ برنامه ای برای آن نوشته شده است. همان طوری که در شکل ۱۰–۵–د و شکل ۱۰–۶ می بینید، ULN2003 در حکم کلیدی عمل می کند که پایه منفی وسیله برقی را نسبت به پایه منفی منبع تغذیه قطع و وصل می کند و فرمان قطع و وصل کردن این کلید را از طریق پایه In از آردوینو دریافت می کند. نکته قابل توجه دیگر اینکه پایه GND از ULN2003 به پایه GND از آردوینو و نیز پایه منفی از منبع تغذیه متصل شده است.

نکته دیگری که در شکل ۱۰–۶ وجود دارد، متصل کردن پایه ۹ از ULN2003 به سر مثبت منبع تغذیه است. صحبت در مورد خاصیت سلفی، خارج از موضوع صحبت ماست. اکنون همین قدر کافی است که بدانیم که در مواردی که می خواهیم موتور یا سیم پیچی را به ULN2003 وصل کنیم خوبست که پایه ۹ را به سر مثبت منبع تغذیه متصل کنیم.



شکل ۱۰-۶:متصل کردن دو موتور با استفاده از ULN2003 به آردوینو

مثال ۱۰-۳

برای مدار شکل ۱۰-۶ برنامه ای بنویسید که به مدت ۱۰ ثانیه موتور ۱ روشن شود. سپس موتور ۲ را نیز روشن کند و مدت ۱۵ ثانیه هر دو موتور روشن باشند. سپس مدت ۱۰ ثانیه هر دو موتور را خاموش کند.

پاسخ:

```
void setup() {
    pinMode(9,OUTPUT); //9 as output
    pinMode(10,OUTPUT); //10 as output
}
void loop() {
    digitalWrite(10,HIGH); //Motor1 on
    delay(10000); //10 sec
    digitalWrite(9,HIGH); //Motor2 on
    delay(15000); //15 sec
    digitalWrite(9,LOW); //Motor1 off
    digitalWrite(10,LOW); //Motor2 off
    delay(10000); //10 sec
}
```

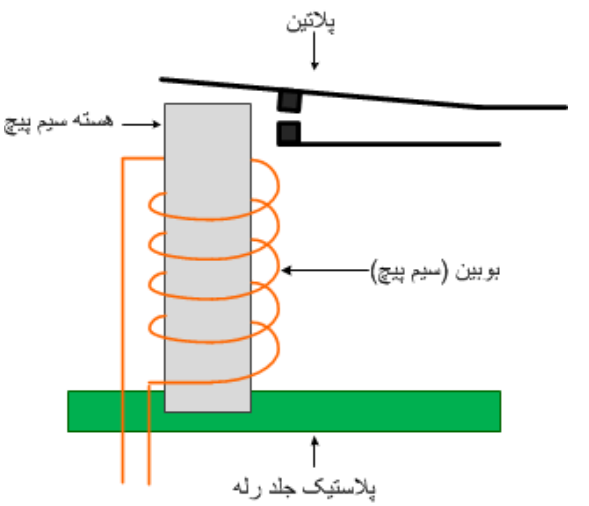
توجه داشته باشد که حداکثر جریانی که هر یک از ترانزیستورهای داخل ULN2003 می‌تواند فراهم کند نیم آمپر است و بنابراین در مواردی که وسیله برقی به جریان زیادی نیاز دارد ULN2003 قادر به روشن کردن وسیله نیست و می‌بایست از ترانزیستورهای IRF استفاده کنید.

بخش ۱۰-۲: رله

تاکنون با ترانزیستور آشنا شده اید. با استفاده از ترانزیستورها می‌توانیم ولتاژهای در حد ۱۲ ولت (ولتاژهای کم) را قطع و وصل کنیم. اما اگر بخواهیم برق متناوب (AC) را قطع و وصل کنیم یا برق وسایلی که با برق شهر کار می‌کنند را قطع و وصل کنیم از رله استفاده می‌کنیم. رله همانند کلید می‌تواند هر برقی را قطع و وصل کند و برایش فرقی نمی‌کند که ولتاژ برق، کم یا زیاد باشد.

رله در حقیقت یک کلید است که در داخل خود یک سیم پیچ دارد. تا مادامی که سیم پیچ خاموش باشد، کلید خاموش است. هرگاه سیم پیچ برق دار شود، خاصیت آهنربایی پیدا می کند و باعث بسته شدن کلید می شود.

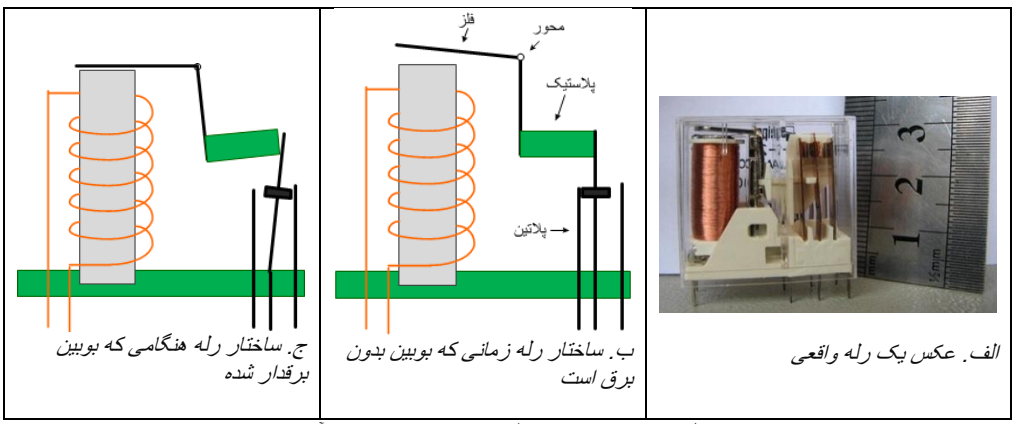
در شکل ۱۰-۷ اجزای یک رله را می بینید. قسمت کلید از رله در حقیقت یک پلاتین (ورقه رسانا) است که خاصیت فنری دارد و به سمت بالا خم شده است. بنابراین در حالت عادی از خود جریانی را عبور نمی دهد. اما اگر دو سر سیم پیچ را به برق متصل کنیم هسته فلزی که در داخل سیم پیچ قرار گرفته است آهنربا می شود و صفحه رسانای کلید را جذب می کند و کلید بسته می شود و می تواند از خود جریانی را عبور دهد.



شکل ۱۰-۷: ساختار ساده شده رله

در زیر عکس یک رله واقعی را به همراه ساختار داخلی آن می بینید. در شکل ۱۰-۷، زمانی که سیم پیچ برقدار می شد و پلاتین را جذب می کرد، پلاتین مستقیما به هسته سیم پیچ تماس پیدا می کرد. برای

اینکه پلاتین مستقیما به هسته سیم پیچ متصل نشود و هسته سیم پیچ برقدار نشود، در ساختار زیر یک فلز L شکل اضافه شده است که ته این فلز، یک تکه پلاستیک (که عایق است) قرار دارد. هرگاه سیم پیچ برقدار شود، فلز L شکل را به سمت خود جذب می کند و فلز L شکل حول محور خود به سمت پایین می چرخد. در این حالت ته فلز L شکل، پلاتین وسطی را به سمت پلاتین دیگر هل می دهد که پلاتینها به هم متصل می شوند. در زیر عکس رله را در زمانی که سیم پیچ رله برقدار است و حالتی که بدون برق است می بینید.



الف. عکس یک رله واقعی

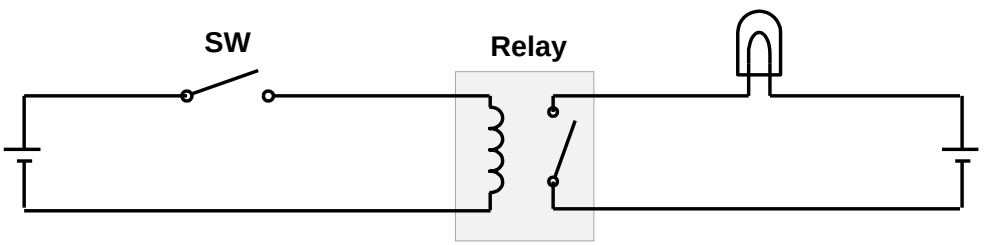
ب. ساختار رله زمانی که بوبین بدون برق است

ج. ساختار رله هنگامی که بوبین برقدار شده

شکل ۱۰-۸: عکس یک رله واقعی و عکس اجزای آن

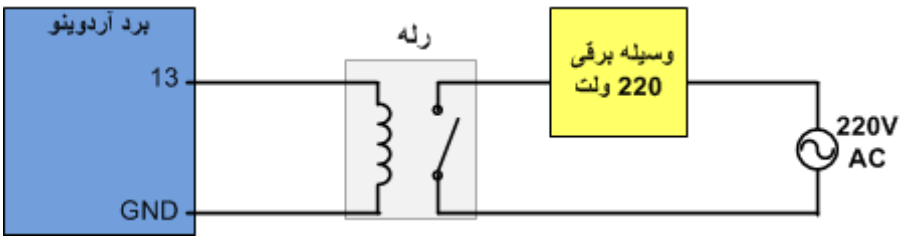
در شکل ۱۰-۹ نماد رله را می بینید. همچنین در شکل ۱۰-۹ یک مدار ساده را می بینید. هرگاه کلید ۱ خاموش باشد، سیم پیچ خاموش است و رله باز است و بنابراین لامپ خاموش است. اما هر گاه کلید ۱ را ببندیم، برق ۵ ولت به دو سر سیم پیچ رله متصل می شود و کلید رله بسته می شود و سبب می شود که چراغ روشن شود.

بنابراین می توانیم رله را به صورت نمایش داده شده در شکل ۱۰-۱۰ به برد آردوینو متصل کنیم تا هرگاه پایه آردوینو روشن شد، یک لامپ ۲۲۰ ولت روشن شود.

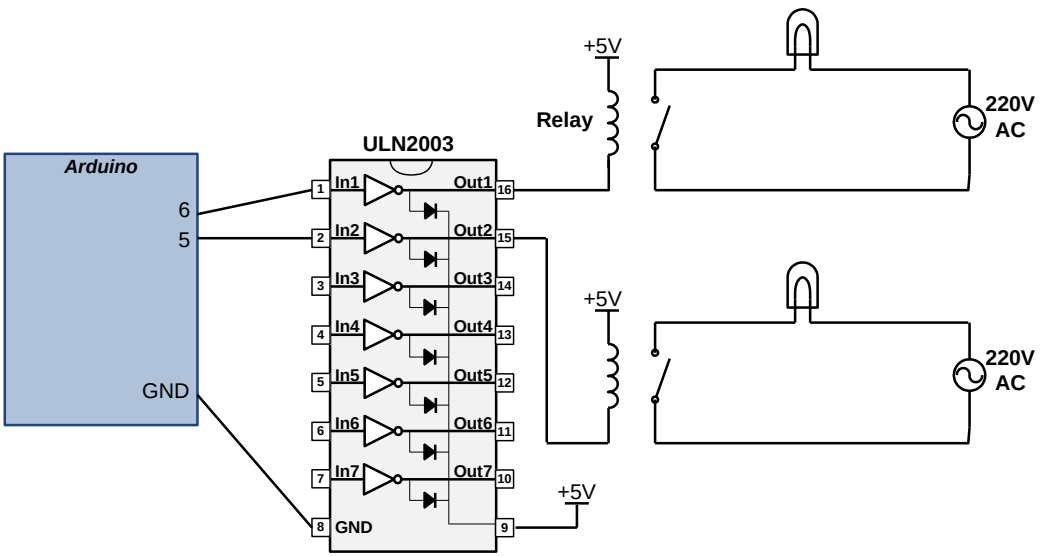


شکل ۱۰ - ۹: مداری ساده برای مشاهده عملکرد رله

ممکن است که برق مورد نیاز جهت برقدار کردن بوبین رله به اندازه ای باشد که پایه های آردوینو نتواند مستقیما جریان مورد نیاز را فراهم کند. بنابراین معمولا بین آردوینو و رله ترانزیستور قرار می دهیم. در شکل ۱۰-۱۱، دو تا لامپ ۲۲۰ ولت را با استفاده از رله و ترانزیستور به برد آردوینو متصل کرده ایم.



شکل ۱۰-۱۰: اتصال رله به آردوینو



شکل ۱۰-۱۱: اتصال دو وسیله برقی ۲۲۰ ولت به آردوینو

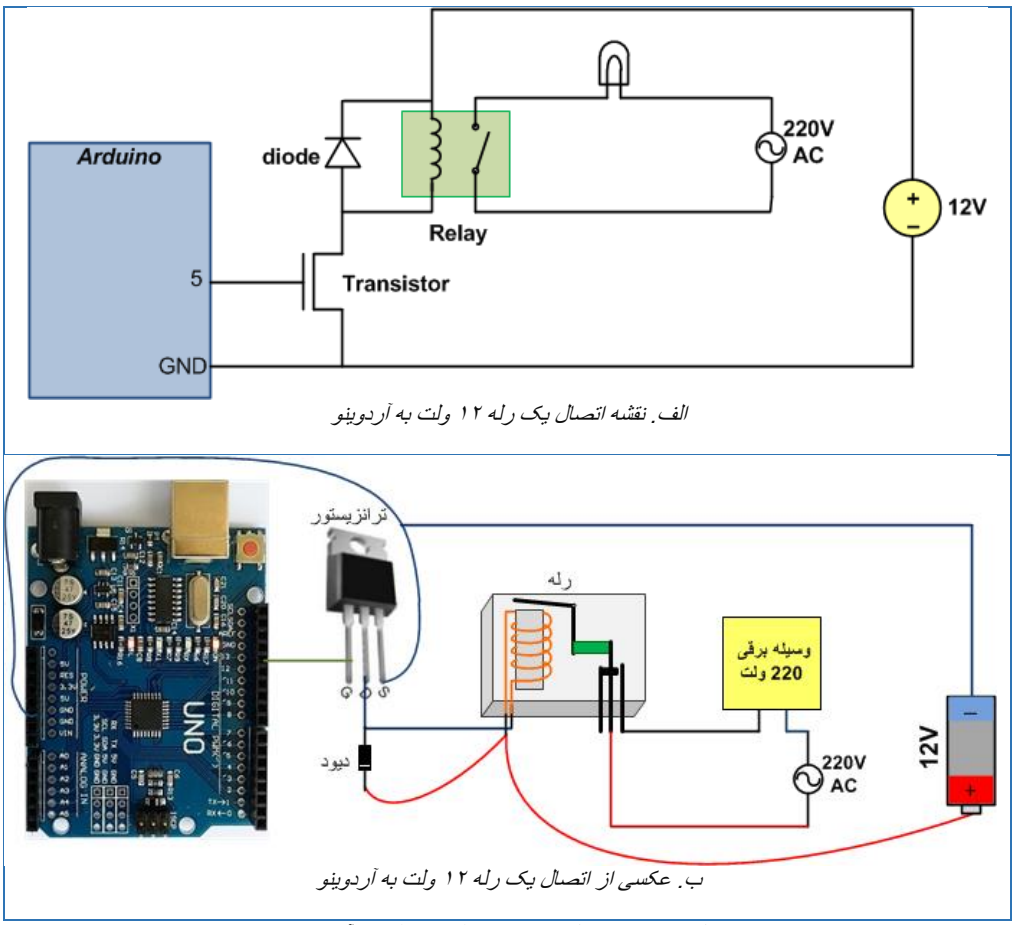
توجه

با توجه به خطرناک بودن برق شهر و امکان برق گرفتگی، به جای برق شهر، جمیع آزمایشهای این قسمت را با برق ۱۲ ولت و وسایل برقی ۱۲ ولت انجام دهید. هیچ گاه در عدم حضور شخصی مطلع، اقدام به انجام آزمایشها بر روی برق شهر ننمایید!

مشخصات رله ها

در بازار رله های بسیار متنوعی وجود دارند. هنگامی که بخواهیم رله را از بازار تهیه کنیم، ۳ تا مشخصه را در مورد رله می بایست انتخاب کنیم:

- **ولتاژ سیم پیچ:** سیم پیچهای رله های مختلف برای اینکه آهنربا شوند به ولتاژهای مختلفی نیاز دارند. اگر ولتاژی که به سیم پیچ متصل می کنیم کمتر از ولتاژ سیم پیچ باشد، ممکن است سیم پیچ نیروی کافی جهت جذب کردن پلاتین را نداشته باشد و رله وصل نشود. همچنین اگر ولتاژی که به سیم پیچ رله می دهیم بیشتر از ولتاژ کاری سیم پیچ باشد، ممکن است سیم پیچ آسیب ببیند. بنابراین ولتاژی که به سیم پیچ می دهیم می بایست برابر با ولتاژ کاری رله باشد. در شکل زیر، از یک رله ۱۲ ولت استفاده شده است و بنابراین ولتاژی که برای رله فراهم شده است نیز ۱۲ ولت است.



الف. نقشه اتصال یک رله ۱۲ ولت به آردوینو

ب. عکسی از اتصال یک رله ۱۲ ولت به آردوینو

شکل ۱۰-۱۲: متصل کردن یک رله ۱۲ ولت به آردوینو

- **جریان قابل گذر از رله:** مقدار جریانی که پلاتین های رله های مختلف، می توانند از خود عبور دهند از رله ای به رله دیگر فرق می کند. اگر جریان گذرا از پلاتینهای رله بیشتر از توانایی رله باشد، ممکن است پلاتین های رله آسیب ببینند.

- **شکل پلاتین های رله:** در قسمت کلید دیدیم که پلاتین های کلید ممکن است معمولا باز (Normally Open)، معمولا بسته (Normally Close) یا دو راهه (Double Throw) باشند. پلاتین های رله ها نیز می توانند هر یک از این انواع باشند. در عین حال، در برخی از رله ها ممکن است که بیشتر از یک پلاتین وجود داشته باشد. مثلا برخی از رله ها درون خود ۲ یا ۳ پلاتین دارند و بنابراین می توانند به طور همزمان چندین برق را با هم قطع و وصل کنند. به طور مثال با استفاده از رله ای که ۲ تا پلاتین دارد می توانیم به طور همزمان یک برق ۵ ولت و یک برق ۱۲ ولت را قطع و وصل کنیم و بدین وسیله یک وسیله برقی ۵ ولت و یک وسیله ۱۲ ولت را به طور همزمان روشن و خاموش کنیم.

تمرین

۱. یک فن ۵ ولت را به کمک ترانزیستور به یکی از پایه های برد آردوینو متصل کنید. سپس برنامه ای بنویسید که هر ۵ دقیقه یکبار، به مدت ۱ دقیقه فن را روشن کند.

۲. برای قطع و وصل کردن برق شهر از چه وسیله ای می توانیم استفاده کنیم؟

۳. می خواهیم با استفاده از سه لامپ ۲۲۰ ولت چراغ راهنمایی درست کنیم. طریقه متصل کردن این لامپها به برد آردوینو را رسم کنید.

فصل ۱۱: آشنایی با رباتیک و موتور

در این فصل می خواهیم ربات چرخدار متحرک بسازیم. امروزه رباتهای متحرک بسیار متنوعی وجود دارند. به طور مثال برخی از آنها پا دارند و راه می روند، برخی دیگر چرخ دارند و حرکت کنند و برخی دیگر پرواز می کنند. در زیر عکس چندین ربات متحرک را می بینید. ما در این فصل تمرکز خود را روی ربات چرخدار می گذاریم. اما آنچه که در این فصل فرا می گیرید، دانش اولیه ای است که بعدا برای ساخت سایر انواع رباتها نیز برای شما مفید است.

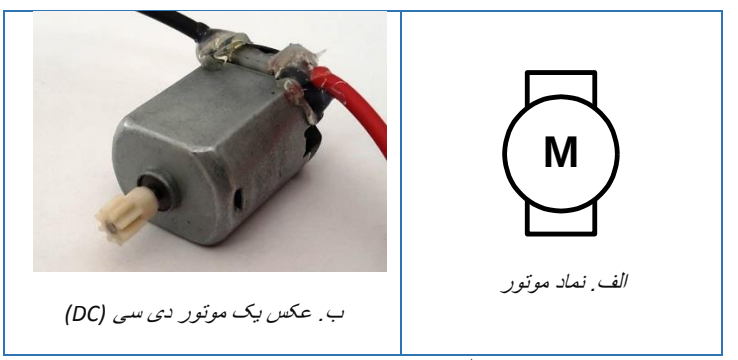
الف. یک نمونه ربات پرنده | ب. ربات چرخدار | پ. نمونه ای از ربات پا دار

شکل ۱۱-۱: عکس تعدادی ربات متحرک

برای اینکه رباتهای چرخدار را به حرکت آوریم معمولا از موتورهای دی سی (DC) استفاده می کنیم. بنابراین ابتدا با موتورهای DC آشنا می شویم و طریقه اتصال آنها به آردوینو را بررسی می کنیم. سپس به چگونگی ساخت ربات می پردازیم.

بخش ۱۱-۱: موتورهای دی سی (DC)

در شکل ۱۱-۲، عکس یک موتور DC را می بینید. موتورهای DC دو سر ورودی برای برق دارند که اگر به آنها برق DC وصل کنیم، موتور شروع به چرخش می کند.



الف. نماد موتور

ب. عکس یک موتور دی سی (DC)

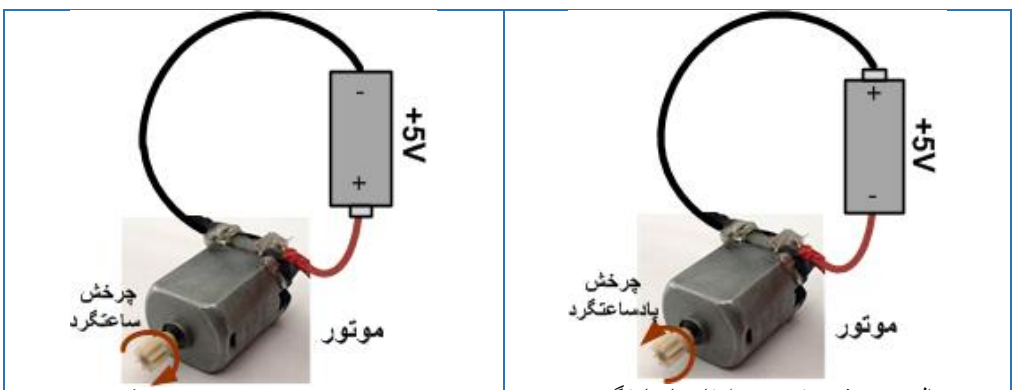
شکل ۱۱-۲: موتور دی سی (DC)

هر موتور DC ولتاژ کاری مشخصی دارد که اگر ولتاژ برقی که به موتور وصل می کنیم به اندازه ولتاژ کاری اعلام شده باشد، موتور بیشترین قدرت و سرعت خود را خواهد داشت. هیچ گاه نباید ولتاژ برقی که به موتوری متصل می کنیم بیشتر از ولتاژ کاری موتور باشد در غیر این صورت موتور خواهد سوخت. معمولا موتورها بر حسب ولتاژ کاریشان نامیده می شوند. مثلا موتوری که ولتاژ کاری آن ۵ ولت باشد، موتور ۵ ولت نامیده می شود و منظور از موتور ۱۲ ولت موتوری است که ولتاژ کاری آن ۱۲ ولت است.

اگر ولتاژ برقی که به موتور DC متصل می کنیم کمتر از ولتاژ کاری آن باشد، سرعت چرخش موتور آهسته تر از حداکثر سرعت موتور خواهد بود. به عبارت دیگر، ولتاژ برقی که به موتور متصل می کنیم با سرعت چرخش آن رابطه مستقیم دارد و هر چه ولتاژ برق بیشتر باشد، سرعت چرخش موتور نیز سریعتر است و البته قدرت موتور نیز بیشتر خواهد بود.

مقدار جریان مصرفی موتورها، به مقدار فشاری که بر روی موتور وارد می شود، بستگی دارد. زمانی که هیچ فشاری به موتور وارد نمی شود و سر آن آزاد می چرخد، کمترین جریان مصرفی را دارد و هر چه فشار بیشتری بر آن وارد شود، سرعت چرخش موتور کاهش می یابد و جریان مصرفی آن نیز افزایش می‌یابد.

یکی از خصوصیتهای جالب موتورهای DC این است که اگر جهت ولتاژ برقی که به دو سر موتور متصل کرده ایم را برعکس کنیم، جهت چرخش موتور نیز برعکس می شود. به عبارت دیگر موتورهای DC، هم ساعتگرد می توانند بچرخند و هم پادساعتگرد و برای اینکه جهت چرخش آنها را برعکس کنیم کافی است که دو سر سیمی که به آنها متصل کرده ایم را برعکس کنیم.



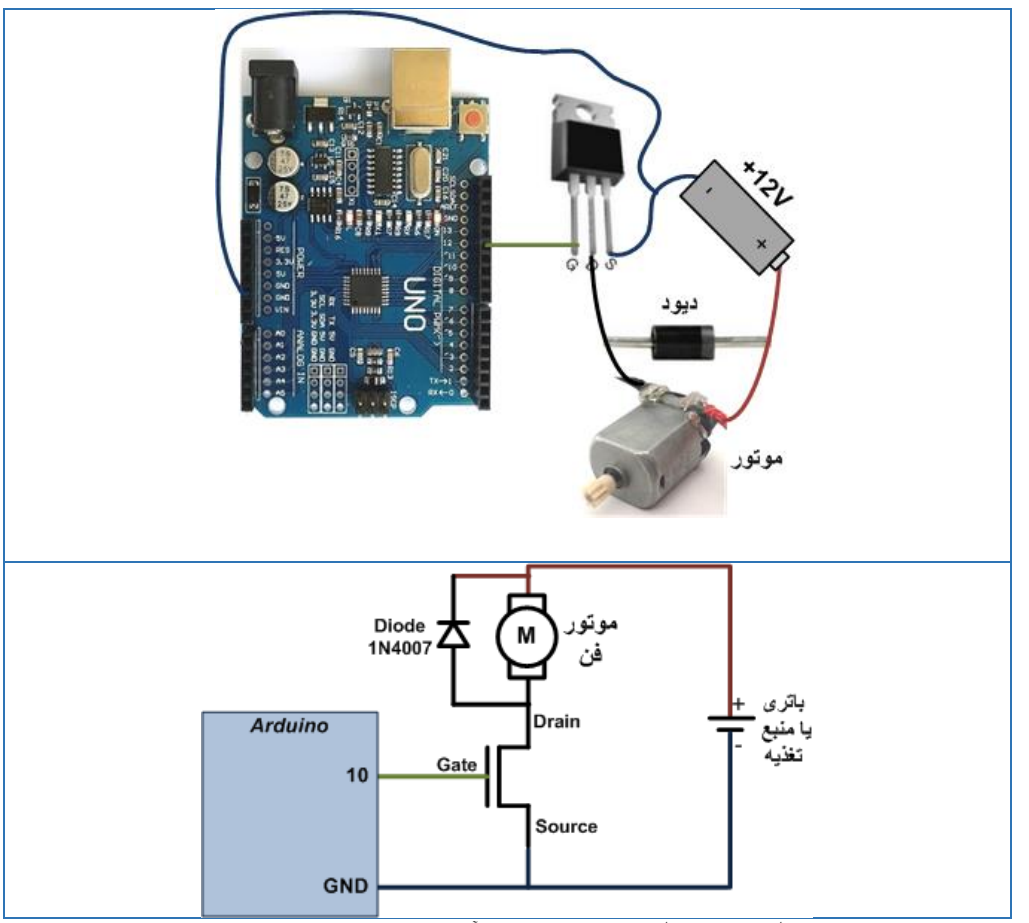
الف. چرخش موتور در راستای پادساعتگرد

ب. چرخش موتور در راستای ساعتگرد

شکل ۱۱-۳: برعکس کردن جهت چرخش موتور DC، از طریق برعکس کردن ولتاژ متصل به آن

اتصال موتور DC به آردوینو

گاهی اوقات مایلیم که موتور DC در جهت ثابتی بچرخد. مثلا مایلیم که موتور فن همیشه در جهت مشخصی بچرخد و به سمت معینی باد بزند و نیازی نیست که در جهت برعکس بچرخد. در چنین مواردی موتورهای DC را می توانیم همانند سایر وسایل برقی با استفاده از ترانزیستور و یا رله به آردوینو متصل کنیم. در زیر مدار اتصال یک موتور DC به آردوینو را می بینید. در مدار زیر، هرگاه پایه ۱۰ از آردوینو را HIGH کنیم، ترانزیستور روشن می‌شود و موتور شروع به چرخش می کند.

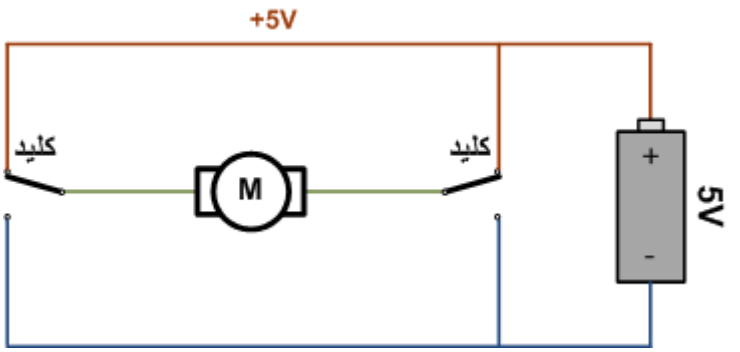


شکل ۱۱-۴: اتصال موتور دی سی به برد آردوینو، با استفاده از ترانزیستور

در مدار فوق، یک باتری ۱۲ ولت کشیده شده است. ولتاژ این باتری، بستگی به ولتاژ کاری موتور دارد و در صورتی که موتور ۵ ولت باشد و جریان مصرفی موتور کم باشد، می توانیم به جای گذاشتن باتری جداگانه، از ولتاژ برد آردوینو استفاده کنیم. در این صورت، آن سر موتور که در مدار فوق به سر مثبت باتری متصل شده است به جای باتری به پایه 5V+ از برد آردوینو متصل می شود.

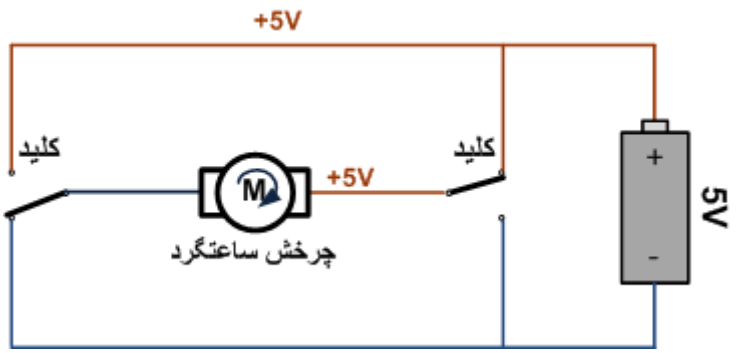
در فصل ۱۰، مشابه مدار فوق را با استفاده از ULN2003 نیز بسته ایم که در صورت تمایل می توانید به آن مراجعه کنید.

در مواقعی که بخواهیم موتور DC در هر دو جهت بچرخد می توانیم مدار H-Bridge (شکل زیر) را پیاده سازی کنیم. در مدار H-Bridge هنگامی که هر دو کلید پایین یا بالا باشند، هر دو سر موتور به ولتاژ یکسانی متصل می شود و موتور خاموش است.

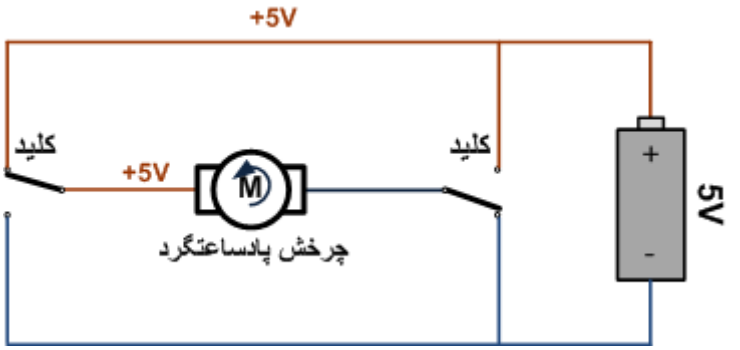


شکل ۱۱-۵: پل اچ شکل (H-Bridge)

در زمانی که کلید سمت چپ پایین باشد و کلید سمت راست بالا باشد (شکل ۱۱-۶)، سر چپ موتور به زمین متصل می شود و سر راستی به ولتاژ متصل می شود که در این حالت موتور ساعتگرد می چرخد. زمانی که کلید سمت چپ بالا باشد و کلید راست پایین باشد (شکل ۱۱-۷)، موتور پادساعتگرد می چرخد.



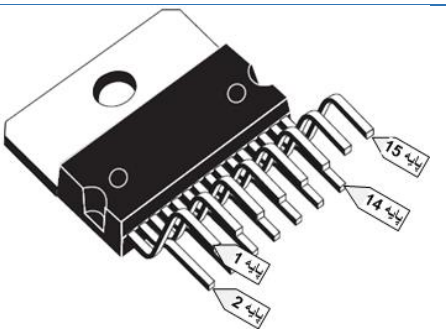
شکل ۱۱-۶: چرخش ساعتگرد



شکل ۱۱-۷: چرخش پادساعتگرد

مدار H-Bridge را می توان با استفاده از ترانزیستور یا رله پیاده سازی کرد. همچنین آی سی هایی نیز وجود دارند که در دل خود، کل مدار H-Bridge را جای داده‌اند که از جمله می توان به L298 و L293 اشاره کرد. در شکل ۱۱–۸ عکس پایه های L298 را می بینید. در L298 پایه های V_S و GND به ترتیب به سر مثبت و سر منفی منبع تغذیه متصل می‌شوند تا برق مورد نیاز موتورها را از منبع تغذیه دریافت کنند. همان طوری که در شکل نشان داده شده است، از طریق پایه های IN1 و IN2 و IN3 و IN4 انتخاب می‌کنیم که هر یک از پایه های خروجی (OUT1 و OUT2 و OUT3 و OUT4) به V_S یا GND متصل شوند. اگر هر یک از پایه های ورودی (INx) را صفر کنیم، پایه خروجی متناظر به GND متصل می‌شود و هرگاه پایه ورودی را یک کنیم، پایه خروجی متناظر به V_S متصل می‌شود. آی سی L298 همچنین دو تا پایه فعال سازی (Enable) به نامهای Enable A و Enable B دارد. همان طوری که در شکل زیر می‌بینید، پایه Enable A مربوط به خروجی های OUT1 و OUT2 است و پایه Enable B مربوط به خروجیهای OUT3 و OUT4 است. اگر پایه Enable را صفر کنیم، اتصال پایه های خروجی از مدار داخلی L298 جدا می‌شود و هیچ ولتاژی به پایه های خروجی اعمال نمی شود؛ اما اگر پایه Enable را یک کنیم، پایه های خروجی به مدار داخلی L298 متصل می‌شوند.

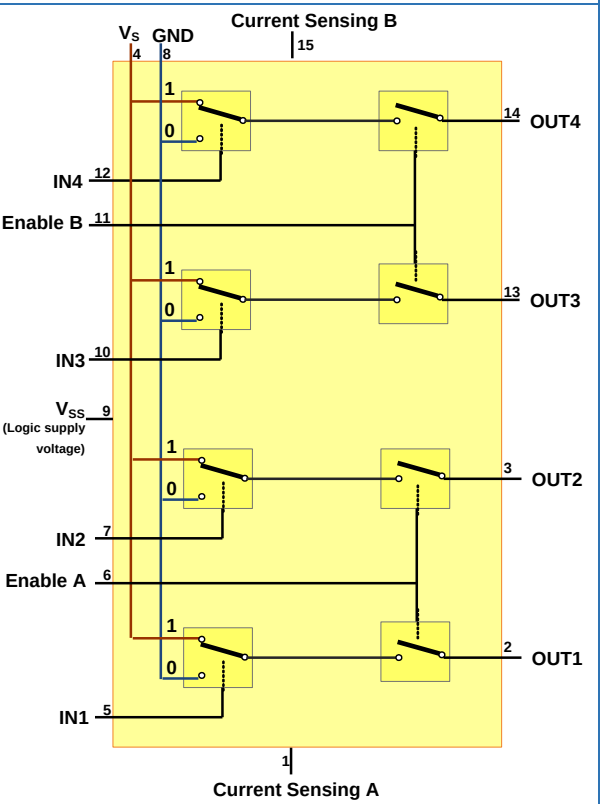
با توجه به توضیحات فوق، با استفاده از هر دو تا پایه خروجی از L298 می توانیم یک H-Bridge (پل H شکل) بسازیم و موتور را در راستاهای ساعتگرد و پادساعتگرد بچرخانیم و بنابراین با استفاده از L298 می‌توانیم دو تا موتور را کنترل کنیم. در شکل ۱۱–۹ طریقه اتصال دو موتور دی سی را به L298 ملاحظه می کنید. در این شکل پایه های Enable را به پایه های آردوینو متصل کرده‌ایم. اما در صورت تمایل می‌توانیم پایه های Enable را مستقیما به +5V متصل کنیم تا L298 پایه های خروجی را دائما به مدار داخلی L298 متصل کند.



الف. عکس L298

ردیف جلو		ردیف عقب	
پایه	نام	پایه	نام
۱	Current Sensing A	۲	OUT1
۳	OUT2	۴	Vs (ولتاژ تغذیه موتورها)
۵	IN1	۶	Enable A
۷	IN2	۸	GND (زمین)
۹	VSS (برق تغذیه)	۱۰	IN3
۱۱	Enable B	۱۲	IN4
۱۳	OUT3	۱۴	OUT4
۱۵	Current Sensing B		

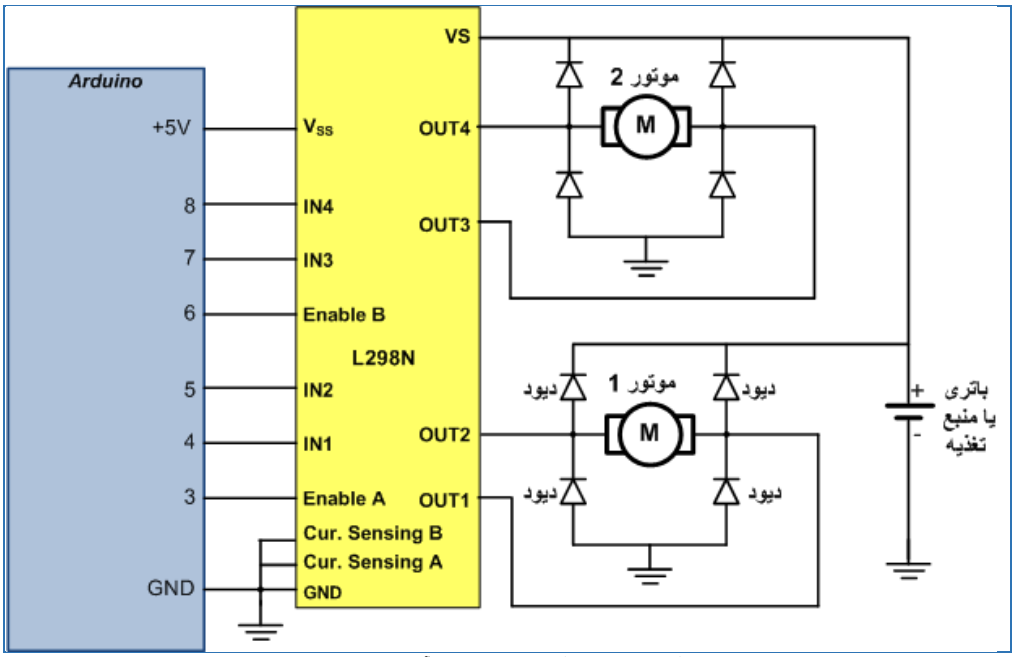
پ. پایه های L298



ب. نقشه ساده شده مدار داخلی L298

شکل ۱۱-۸: آی سی L298

البته آی سی L298N به صورت ماژول آماده نیز با قیمت مناسب در بازار وجود دارد که در صورت تمایل می توانید از آن استفاده کنید. درون ماژول آماده L298، آی سی L298 به همراه دیودهای مورد نیاز قرار گرفته اند که به سهولت می توانید موتورها را به پایه های ماژول متصل کنید و از آن استفاده کنید. در شکل ۱۱-۱۰-الف عکس یک ماژول L298 را می بینید.

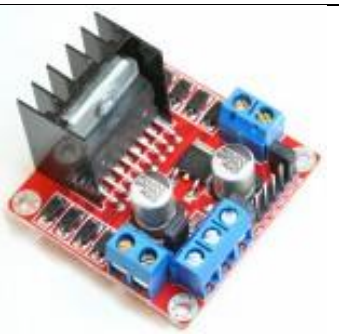


شکل ۱۱-۹: اتصال دو تا موتور به آی سی L298

در ماژول L298، در کنار پایه های Enable، پایه های ولتاژ 5+ قرار گرفته اند و بر روی آنها جامپری قرار گرفته است. در شکل ۱۱–۱۰–ب عکس تعدادی جامپرِ نمایش داده شده است. جامپرِ (Jumper)، در حقیقت یک تکه سیم است که دو نقطه از مدار را به هم متصل می‌کنند. با استفاده از جامپرهایی که بر روی پایه های Enable قرار گرفته اند، پایه های Enable به 5+ متصل می شوند. هنگامی که پایه Enable را به 5V+ متصل کنیم، مدار داخلی L298 به پایه های خروجی متصل می شود. در شکل ۱۱–۱۰–پ طریقه متصل کردن موتور و برد آردوینو را به ماژول L298، می بینید.

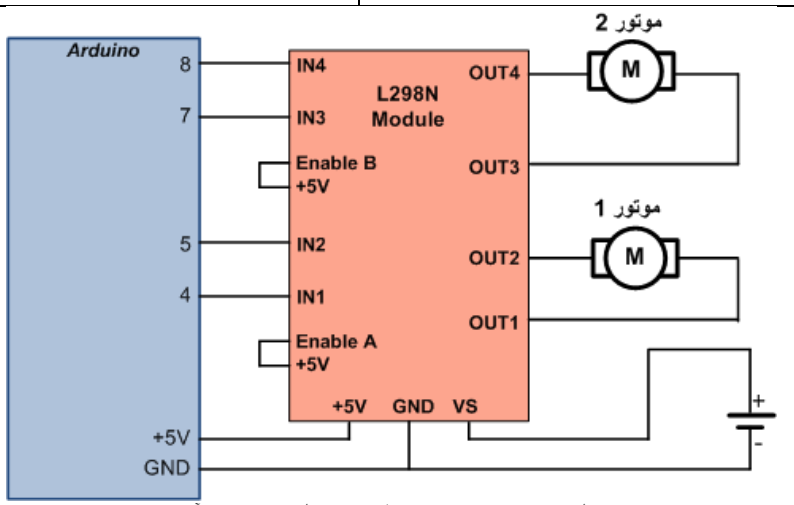
در شکل ۱۱–۱۰–پ، جامپرها، پایه های Enable را به ولتاژ ۵ ولت متصل کرده اند. اما اگر بخواهیم پایه های Enable را به پایه های آردوینو متصل کنیم می بایست این جامپرها را برداریم. در مثالهای اولیه این فصل، به جامپرهای پایه های Enable دست نمی زنیم و می گذاریم که پایه های Enable دائما فعال باشند. اما در بخشهای بعد، جامپرهای Enable را بر می داریم و پایه های Enable را به پایه های خروجیِ آنالوگِ (PWM) آردوینو متصل کنیم تا به کمک آن سرعت چرخش موتور را کنترل کنیم.

در مثال ۱۱–۱، برنامه ای بر حسب مدار شکل ۱۱–۹ (و شکل ۱۱–۱۰) نوشته شده است و پایه های آی سی L298 را به گونه ای مقداردهی می کند که موتور یک در راستای ساعتگرد شروع به چرخش می‌کند.



الف. عکس ماژول L298N

ب. عکس تعدادی جامپر (کپی شده از سایت آمازون)



پ. متصل کردن دو تا موتور از طریق ماژول L298N به آردوینو

شکل ۱۱-۱۰: طرز متصل کردن موتورها به ماژول L298

مثال ۱۱-۱

بر حسب شکل فوق، برنامه ای بنویسید که موتور یک در راستای ساعتگرد بچرخد.

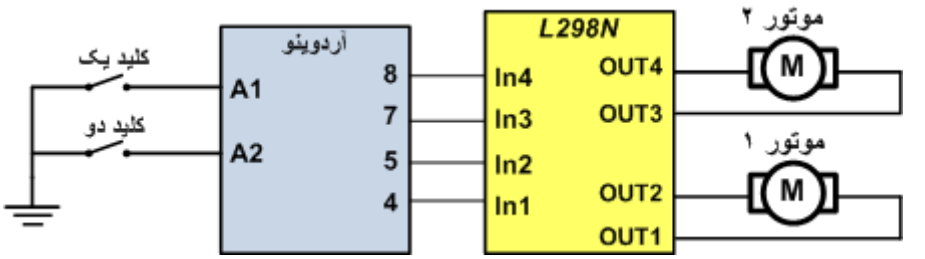
پاسخ:

```
void setup() {
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output
}

void loop() {
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low
}
```

در مثال ۱۱-۲، برنامه ای نوشته شده است که هر دو موتور را کنترل کند.

مثال ۱۱-۲

در شکل زیر دو تا کلید به پایه های A1 و A2 از آردوینو متصل کرده ایم. برنامه ای بنویسید که هر گاه کلید ۱ فشرده شده بود، هر دو موتور در راستای ساعتگرد بچرخند. هر گاه کلید ۲ فشرده شد، هر دو موتور در راستای پادساعتگرد بچرخند و هنگامی که هر دو کلید رها بودند، موتورها بایستند.



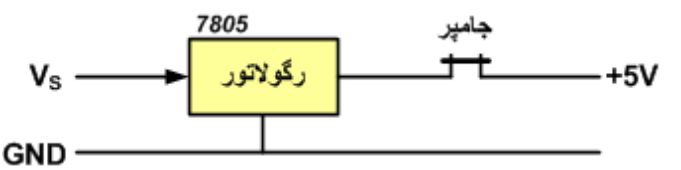
پاسخ:

```
void setup() {
    pinMode(3,OUTPUT); // 3(enableA) = output
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output

    pinMode(6,OUTPUT); // 6(enableB) = output
    pinMode(7,OUTPUT); // 7(out3) = output
    pinMode(8,OUTPUT); // 8(out4) = output

    pinMode(A1,INPUT_PULLUP);
    pinMode(A2,INPUT_PULLUP);
}

void loop() {
    if(digitalRead(A1) == LOW)
    {
    digitalWrite(3,HIGH); //EnableA = high
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low

    digitalWrite(6,HIGH); //EnableB = high
    digitalWrite(7,HIGH); //out3 = high
    digitalWrite(8,LOW);  //out4 = low
    }
    else
    if(digitalRead(A2) == LOW)
    {
        digitalWrite(3,HIGH); //EnableA = high
        digitalWrite(4,LOW);  //out1 = low
        digitalWrite(5,HIGH); //out2 = high
```

```
        digitalWrite(6,HIGH); //EnableB = high
        digitalWrite(7,LOW);  //out3 = low
        digitalWrite(8,HIGH); //out4 = high
    }
    else
    {
        digitalWrite(3,HIGH); //EnableA = high
        digitalWrite(4,LOW);  //out1 = low
        digitalWrite(5,LOW);  //out2 = low

        digitalWrite(6,HIGH); //EnableB = high
        digitalWrite(7,LOW);  //out3 = low
        digitalWrite(8,LOW);  //out4 = low
    }
}
```

رگولاتور موجود در ماژولهای L298

درون برخی از ماژولهای L298 یک آی سی رگولاتور وجود دارد. در شکل ۱۱-۱۱ مدار مربوط به رگولاتور را می بینید. کار آی سی های رگولاتور، کاهش دادن ولتاژ و ایجاد کردن ولتاژهای ثابت است. مثلا می توانیم به رگولاتور موجود در ماژولهای L298، ولتاژهای ۷ ولت تا ۱۲ ولت بدهیم و از خروجی آن ولتاژ ثابت ۵ ولت را دریافت کنیم. همان طوری که در شکل زیر می بینید، پایه ورودی این آی سی رگولاتور به پایه Vs از ماژول L298 متصل شده است و پایه خروجی آن، پس از گذشتن از یک جامپر، به پایه 5+ از ماژول L298 می‌رسد. (در برخی از ماژولهای L298، پایه Vs با نام 12+ نامگذاری شده است.) در مواقعی که ولتاژ کاری موتور رباتمان بیشتر از ٦ ولت است اگر جامپر مربوط به رگولاتور، سر جایش قرار داشته باشد، رگولاتور بر روی پایه 5+ ولت از ماژول L298، ولتاژ ۵ ولت را ایجاد می کند که از آن می توانیم برق مورد نیاز جهت کار کردن آردوینو را فراهم کنیم. اما در مواردی که خودمان برق مورد نیاز آردوینو را فراهم می کنیم و یا در مواردی که می خواهیم ولتاژ ۵ ولت را به طور همزمان جهت تغذیه کردن موتور و برد آردوینو استفاده کنیم، جامپر رگولاتور را بر می‌داریم تا رگولاتور غیرفعال شود.



شکل ۱۱-۱۱:مدار رگولاتور موجود در ماژولهای L298

بخش ۱۱-۲: ساخت ربات چرخدار

بررسی چگونگی عملکرد وسایل چرخدار

قبل از اینکه ربات خود را بسازیم خوبست که وسایل چرخدار اطراف خود را بررسی کنیم و از آنها ایده بگیریم. شما در اطراف خود وسایل چرخدار بسیار متنوعی را دیده اید که هر یک از آنها کاربرد خاص خود را دارند. از جمله تفاوتهایی که بین وسایل چرخدار مختلف وجود دارد به قرار زیر است:

- تعداد چرخهایی که وسیله دارد
- چرخهایی که به آنها نیروی محرکه وارد می شود
- روشی که وسیله تغییر جهت می دهد و به سمت راست یا چپ دور می زند

تعداد چرخهای وسایل

برخی وسایل مثل دوچرخه و موتورسیکلت دو چرخ دارند که در این وسایل، شخص راننده نقش حفظ تعادل وسیله را برعهده دارد. در وسایلی که سه چرخ و بیشتر دارند، (مثل سه چرخه، اتومبیل، تریلی) وسیله نقلیه تعادل دارد و اگر آن را رها کنیم روی زمین نمی افتند.

چرخهایی که نیروی محرکه به آنها وارد می شود

در دو چرخه و موتور سیکلت، نیروی محرکه به چرخهای عقب وارد می شود. در برخی از سه چرخه ها نیروی محرکه به چرخ جلو وارد می شود و در برخی دیگر به چرخهای عقب وارد می شود.

در برخی از وسایل چهار چرخ و اتومبیلها، نیروی محرکه به چرخهای جلو وارد می شود و در برخی دیگر به چرخهای عقب. البته اتومبیلهایی نیز هستند که نیروی محرکه به همه چهار چرخ وارد می شود. فایده وارد آمدن نیروی محرکه بر چهار چرخ این است که هنگامی که یکی از چرخهای اتومبیل در سطوح ناصاف یا برف گیر می کند، وسیله نقلیه می تواند به کمک سایر چرخها، خود را از چاله و یا برف بیرون بکشد. اما به حرکت آوردن تعداد بیشتری از چرخها، هزینه بیشتر و مصرف سوخت بیشتری را به دنبال دارد.

در بلدوزر و تانک برای اینکه بتوانند بر روی سطوح کاملا ناصاف حرکت کنند تعداد زیادی چرخ در دو طرف وسیله قرار گرفته است که دور آنها زنجیری کشیده شده است و نیروی محرکه به زنجیر وارد می‌شود. با توجه به اینکه زنجیر تقریبا تمام سطح زیرین وسیله نقلیه را پوشانده است، در ناصاف ترین سطوح نیز حتما قسمتی از زنجیرها با زمین در تماس است و می تواند بلدوزر را به جلو براند.

روشی که وسیله تغییر جهت می دهد (فرمان وسیله)

در اکثر وسایل نقلیه از قبیل دوچرخه، موتورسیکلت و اتومبیلها برای اینکه راستای حرکت وسیله نقلیه تغییر کند، چرخهای جلو به چپ یا راست منحرف می شوند و راستای چرخ جلو مشخص کننده جهت حرکت وسیله است.

در معدودی از وسایل (مثلا لیفتراک) به خاطر کاربرد خاصی که دارند، فرمان به چرخهای عقب وارد می شود و نیروی محرکه نیز به چرخهای عقب وارد می شود و بنابراین بدون اینکه جلوی آنها جابجا شود، قادرند که دم خود را به چپ و راست ببرند.

در بلدوزر و تانک، هر گاه بخواهند به چپ بروند، چرخهای سمت چپ می ایستند و چرخهای سمت راست می چرخند که موجب چرخش بلدوزر و تانک به چپ می شود. همچنین زمانی که بخواهند به راست بروند، چرخهای سمت راست می ایسند و یا سرعت آنها آهسته میشود و چرخهای چپ می چرخند که در نتیجه بلدوزر به سمت راست می رود.

ساخت ربات

برای ساخت ربات می توانیم تک تک اجزای ربات از قبیل چرخ موتور گیربکس و غیره را به طور جداگانه از بازار تهیه کنیم و آنها را سر هم کنیم. همچنین می توانیم از شاسی رباتهای آماده ای که در بازار وجود دارند استفاده کنیم. در زیر عکس تعدادی از شاسی رباتهای موجود در بازار را ملاحظه می کنید.

شکل ۱۱- ۱۲: نمونه هایی از شاسی های رباتهای چرخدار

ما در گام نخست رباتی با سه چرخ می سازیم که نیروی محرکه بر روی دو چرخ عقب وارد می شود و همانند بلدوزر، هنگامی که می خواهیم به چپ یا راست برویم، چرخ سمت چپ یا راست را متوقف می کنیم. چرخ جلو از طریق بلبورینگ یا هرزگرد به ربات متصل می شود و کاملا آزاد در راستاهای مختلف می چرخد و نقش آن صرفا حمل کردن قسمتی از وزن ربات و حفظ تعادل آن است. البته اصولی

که ما جهت کنترل ربات سه چرخ مطرح می کنیم عینا برای رباتهای تانکی شکل نیز قابل انجام است. اما از آنجایی که رباتهای سه چرخ بسیار ارزان تر هستند، تمرکز خود را بر روی ربات سه چرخ می گذاریم.

چنانچه می خواهید اجزای ربات را به صورت جداگانه خریداری کنید، لازم است که دو تا موتور و گیربکس همسان به همراه ۲ تا چرخ به همراه یک چرخ بلبورینگ دار، از فروشگاه هایی که وسایل ربات می فروشند خریداری کنید. به عنوان شاسی ربات خود می توانید یک صفحه چوبی یا پلاستیکی را به شکل مستطیل ببرید و یا جعبه پلاستیکی یا فلزی شیرینی که شکلی مستطیلی داشته باشد، را برای این منظور استفاده کنید. سپس با استفاده از پیچ، موتورها و چرخها را بر روی شاسی محکم کنید.

کنترل ربات

همان طوری که در قسمت قبل دیدیم با استفاده از L298 می توانیم دو تا موتور DC را به سهولت به برد آردوینو متصل کنیم.

در جدول زیر رابطه حرکتی که هر یک از چرخهای ربات می کنند با حرکتی که کل ربات انجام می‌دهد را می بینید. توجه دارید که هنگامی که اتومبیلی به جلو حرکت می کند، هنگامی که از بیرون اتومبیل به چرخهای اتومبیل نگاه کنیم، چرخهای سمت چپ در جهت پادساعتگرد می چرخند. اما چرخهای سمت راست در راستای ساعتگرد می چرخند. به همین صورت، ربات نیز هنگامی که به جلو می رود، چرخ چپ در راستای پادساعتگرد و چرخ راست در راستای ساعتگرد می چرخد.

راستای حرکت ربات	جهت چرخش چرخ راست	جهت چرخش چرخ چپ
ایستاده	ایستاده	ایستاده
چرخش به راست همراه با جلو رفتن	ایستاده	پادساعتگرد (جلو)
چرخش به چپ همراه با عقب رفتن	ایستاده	ساعتگرد (عقب)
چرخش به چپ و جلو	ساعتگرد (جلو)	ایستاده
حرکت مستقیما به سمت جلو	ساعتگرد (جلو)	پادساعتگرد (جلو)
چرخش به سمت چپ	ساعتگرد (جلو)	ساعتگرد (عقب)
چرخش به راست و عقب	پادساعتگرد (عقب)	ایستاده
چرخش به راست	پادساعتگرد (عقب)	پادساعتگرد (جلو)
حرکت مستقیما به عقب	پادساعتگرد (عقب)	ساعتگرد (عقب)

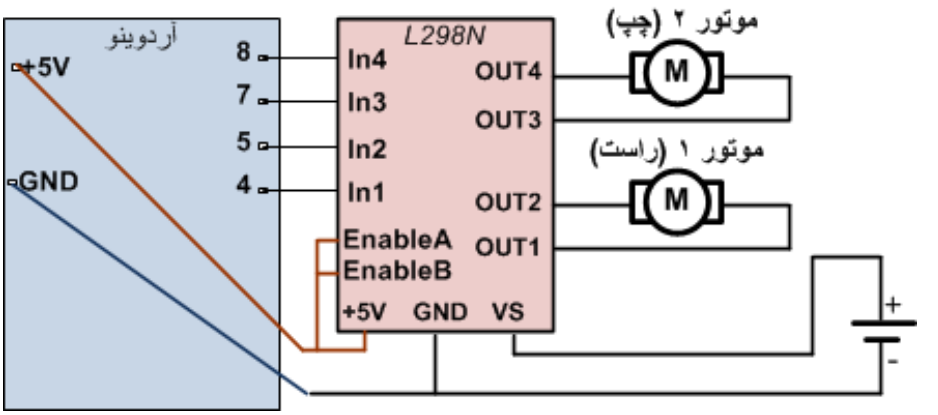
جدول ۱۱-۱: رابطه بین جهت چرخش چرخها با حرکت ربات

در ادامه به نوشتن چندین برنامه برای ربات می پردازیم.

مثال ۱۱-۳: برنامه ای بنویسید که ربات به مدت ۳ ثانیه رو به جلو حرکت کند و سپس به مدت ۳ ثانیه به عقب برود

پاسخ:

همان طوری که در جدول ۱۱-۱ ذکر شده است، برای اینکه ربات رو به جلو حرکت کند، چرخ چپ می بایست به صورت پادساعتگرد و چرخ راست به صورت ساعتگرد بچرخد. بنابراین اگر موتورها را مطابق شکل ۱۰-۱۱ به آردوینو متصل کرده باشیم و موتور یک، موتور راست ربات و موتور دو، موتور چپ ربات باشد، برنامه به صورت زیر می شود:



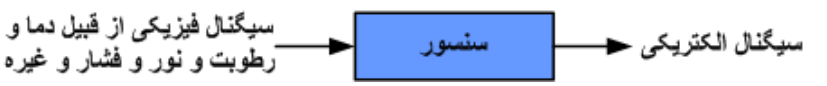

```
void setup() {
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output

    pinMode(7,OUTPUT); // 7(out3) = output
    pinMode(8,OUTPUT); // 8(out4) = output
}

void loop() {
    //---------------
    // move forward
    //---------------
    //Motor 1 (Right): clockwise
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low

    //Motor2 (Left): counter clockwise
    digitalWrite(7,LOW);  //out3 = low
    digitalWrite(8,HIGH); //out4 = high

    digitalRead(3000);      // 3 seconds

    //---------------
    // move backward
    //---------------
```

```
    //Motor 1 (Right): counter clockwise
    digitalWrite(4,LOW);  //out1 = low
    digitalWrite(5,HIGH); //out2 = high

    //Motor 2 (Left): clockwise
    digitalWrite(7, HIGH);  //out3 = high
    digitalWrite(8, LOW); //out4 = low
    digitalRead(3000);      //3 seconds
}
```

بخش ۱۱-۳: سنسورهای تشخیص مانع و دنبال کننده خط

در برنامه قبلی ربات حرکت می کرد بی آنکه تصوری از محیط اطراف خودش داشته باشد و اگر به مانعی برخورد می کرد، متوجه نمی شد. اما خوبست که ربات محیط اطراف خود را درک کند و بر حسب شرایط محیط اطراف تصمیم بگیرد.

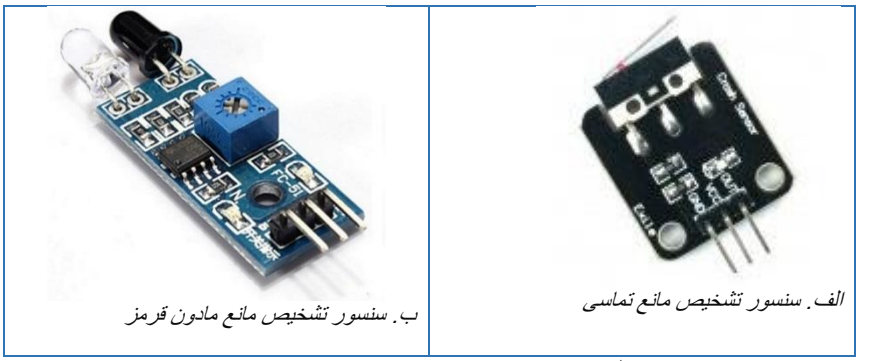
ما دنیای اطراف خودمان را با حواس پنج گانه حس می کنیم. مغز ما قابلیت پردازش سیگنالهای عصبی را دارد و مستقیما نمی‌تواند، رنگ، صدا، بو و غیره را پردازش کند. بنابراین هنگامی که ما صدایی را می‌شنویم گوش، موج صدا را تبدیل به سیگنال عصبی می کند تا مغز بتواند در مورد آن تصمیم بگیرد. در دنیای الکترونیک و کامپیوتر مدارها قادرند که سیگنالهای الکتریکی که به شکل ولتاژ، جریان، مقاومت، خاصیت سلفی و خازنی و غیره هستند را مورد پردازش و تحلیل قرار دهند. سنسورها، برای ما کار تبدیل سیگنالهای محیط اطراف از قبیل دما، رطوبت و غیره را به سیگنالهای الکتریکی انجام می دهند.



شکل ۱۱-۱۳: عملکرد سنسورها

برای شناسایی محیط اطراف، سنسورهای بسیار متنوعی وجود دارند. مثلا برخی از رباتها از دوربین برای شناسایی محیط اطراف استفاده می کنند. با استفاده از رادار نیز می توان تصویر دقیقی از موانع اطراف به دست آورد. اما پردازش تصویر و پردازش سیگنال و مباحث هوش مصنوعی فراتر از موضوع این کتاب هستند و ما در این کتاب به آنها نمی پردازیم. در این فصل ما با برخی از سنسورهای تشخیص مانع آشنا می شویم و در فصل بعد با استفاده از سنسور ماوراء صوت فاصله خود از موانع را اندازه گیری می کنیم. در شکل ۱۱-۱٤ عکس دو تا سنسور تشخیص مانع را می بینید.

سنسورهای تشخیص مانع علاوه بر ربات، در صنعت نیز کاربرد فراوانی دارند. مثلا در هنگام بسته بندی می توانیم تعداد وسایلی که از جلوی سنسور عبور می کنند را بشماریم و زمانی که تعداد وسایلی که داخل جعبه قرار گرفتند به تعداد مشخصی رسید، درب کارتن را ببندیم.



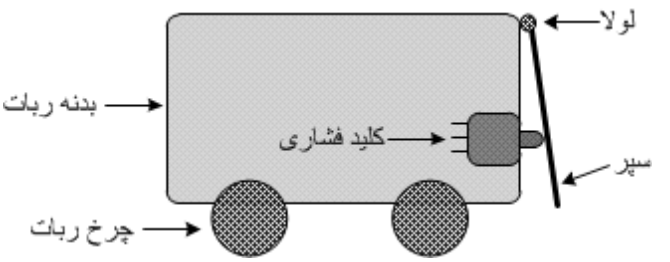
الف. سنسور تشخیص مانع تماسی

ب. سنسور تشخیص مانع مادون قرمز

شکل ۱۱-۱۴: نمونه هایی از سنسورهای تشخیص مانع

سنسور تشخیص مانع تماسی

اگر با استفاده از یک تکه فلز یا پلاستیک، شاخک یا سپر درست کنیم و آن را به یک کلید فشاری متصل کنیم (شکل زیر)، هر موقع شاخک به مانع برخورد کند، کلید فشرده می شود و بنابراین از وضعیت کلید می توانیم تشخیص دهیم که ربات به مانعی برخورد کرده است. سنسورهای تشخیص مانع تماسی با قیمتهای مناسب در بازار وجود دارند که عکس نمونه ای از آن را در شکل ۱۱-۱۴-الف می بینید.



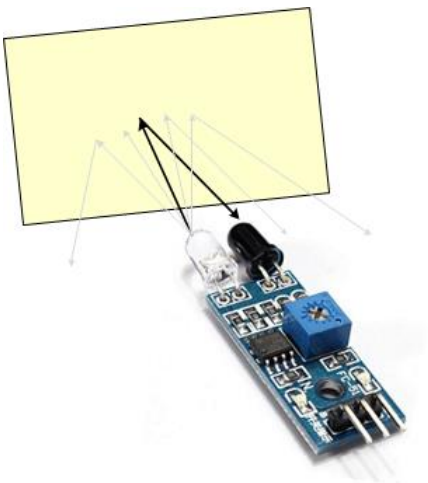
شکل ۱۱-۱۵: استفاده کردن از سپر به عنوان مانع تماسی

سنسور تشخیص مانع مادون قرمز

سنسورهای تشخیص مانع تماسی برای اینکه پی به وجود مانع ببرند لازم است که به آن برخورد کنند. اما بهتر است که قبل از اینکه ربات به مانعی برخورد کند، پی به وجود آن ببرد تا اصلا به مانع برخورد نکند. سنسورهای تشخیص مانع مادون قرمز، دارای یک فرستنده مادون قرمز و یک گیرنده مادون قرمز است. همان طوری که در شکل ۱۱-۱۶ می بینید، اگر نزدیک سنسور تشخیص مانع مادون قرمز، مانعی

وجود داشته باشد، نوری که توسط فرستنده مادون قرمز ایجاد می شود، به مانع می خورد و بازتاب می‌کند و به گیرنده مادون قرمز می رسد و سنسور مادون قرمز متوجه وجود مانع می شود.

در شکل ۱۱-۱۶ و شکل ۱۱-۱٤-ب عکس یک سنسور تشخیص مانع مادون قرمز را می بینید. این سنسور، سه پایه دارد: VCC، GND و خروجی. پایه های VCC و GND را می بایست به ترتیب به پایه های +5V و GND از برد آردوینو متصل کنیم تا برق را برای سنسور فراهم کنند. تا مادامی که مانعی جلوی سنسور وجود نداشته باشد پایه خروجی سنسور HIGH است و در غیر این صورت، LOW می باشد. در مثالهای ۱۱-٤ و ۱۱-٥ سنسور تشخیص مانع را به ربات خود اضافه می کنیم و برای آن برنامه می نویسیم.

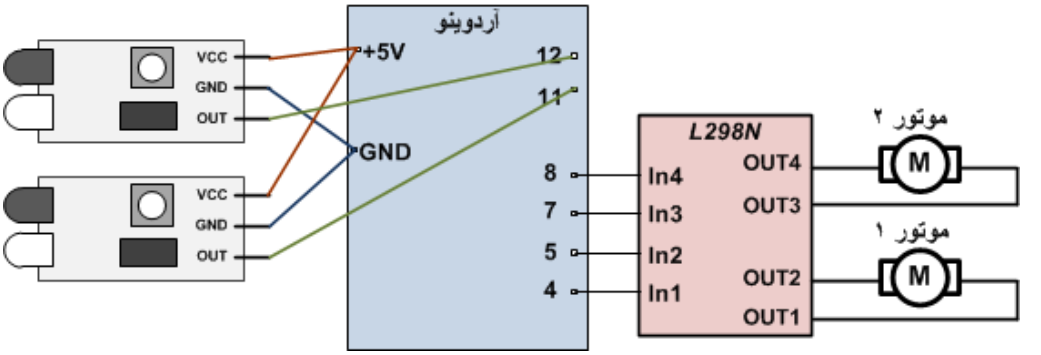


شکل ۱۱-۱۶: چگونگی کار کردن سنسور تشخیص مانع مادون قرمز

مثال ۱۱-۴: یک سنسور تشخیص مانع جلو ربات و یک سنسور تشخیص مانع در عقب ربات نصب کنید. سپس برنامه ای بنویسید که ربات به سمت جلو حرکت کند تا به مانعی برسد. سپس به سمت عقب برود تا به مانعی برسد و این حرکت رفت و برگشت را بین دو مانع دائما تکرار کند.

پاسخ:

مطابق شکل، پایه خروجی سنسور تشخیص مانع جلو را به پایه ۱۱ و سنسور تشخیص مانع عقب را به پایه ۱۲ متصل می کنیم. سپس برنامه ای به شکل زیر می نویسیم:




```
void setup() {
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output

    pinMode(7,OUTPUT); // 7(out3) = output
    pinMode(8,OUTPUT); // 8(out4) = output

    pinMode(11,INPUT_PULLUP);
    pinMode(12,INPUT_PULLUP);
}

void loop() {

    while(digitalRead(11) == HIGH) {
    //---------------
    // move forward
    //---------------
    //Motor 1 (Right): clockwise
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low

//Motor2 (Left): counter clockwise
    digitalWrite(7,LOW);  //out3 = low
    digitalWrite(8,HIGH); //out4 = high
    }
    while(digitalRead(12) == HIGH) {
    //---------------
    // move backward
    //---------------
    //Motor 1 (Right): counter clockwise
    digitalWrite(4,LOW);  //out1 = low
    digitalWrite(5,HIGH); //out2 = high

    //Motor 2 (Left): clockwise
    digitalWrite(7,HIGH);  //out3 = high
    digitalWrite(8,LOW); //out4 = low
    }
}
```

مثال ۱۱-۵: یک سنسور تشخیص مانع جلوی ربات نصب کنید. سپس برنامه ای بنویسید که ربات به جلو برود تا به مانعی برسد. هرگاه به مانعی رسید چرخش درجا به سمت راست کند تا جایی که دیگر مانعی جلوی ربات نباشد و سپس رو به جلو برود.

پاسخ:

```
void setup() {
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output

    pinMode(7,OUTPUT); // 7(out3) = output
    pinMode(8,OUTPUT); // 8(out4) = output

    pinMode(11,INPUT_PULLUP);
}

void loop() {

    if(digitalRead(11) == HIGH) {
    //---------------
    // move forward
    //---------------
    //Motor 1 (Right): clockwise
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low

    //Motor2 (Left): counter clockwise
    digitalWrite(7,LOW);  //out3 = low
    digitalWrite(8,HIGH); //out4 = high
    }
    else{
    //---------------
    // rotate right
    //---------------
    //Motor1 (Right): counter clockwise
    digitalWrite(4,LOW);  //out1 = low
    digitalWrite(5,HIGH); //out2 = high

    //Motor2 (Left): counter clockwise
    digitalWrite(7,LOW);  //out3 = low
    digitalWrite(8,HIGH); //out4 = high
    }
}
```

در نظر داشته باشید که از آنجایی که سنسور مادون قرمز بر حسب بازتاب نور کار می کند، ممکن است اشتباها نور خورشید و یا برخی از نورها را به عنوان بازتاب نور تشخیص دهد و اعلام وجود داشتن

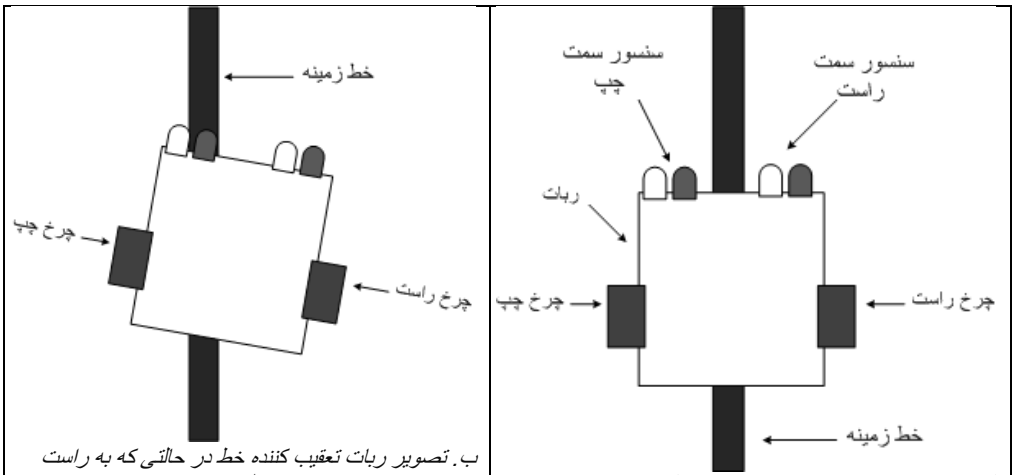
مانع کند. همچنین با توجه به اینکه سطوح مشکی تابش نور کمتری دارند، در فواصل نزدیکتری قابل تشخیص هستند.

بر روی برخی از سنسورهای تشخیص مانع مادون قرمز، ولومی قرار گرفته است که با چرخاندن آن می توانیم میزان حساسیت سنسور مادون قرمز را تنظیم کنیم. هر چه حساسیت سنسور بیشتر باشد، موانع را از فاصله دورتری تشخیص می دهد، اما احتمال اینکه اشتباها نورهای محیطی را به عنوان مانع تشخیص دهد نیز بیشتر می شود. از طرف دیگر اگر حساسیت سنسور کم باشد، ممکن است موانع تیره را تشخیص ندهد و یا در فواصل نزدیک تشخیص دهد. بنابراین می توانیم ولوم را بر حسب نیاز خود به گونه ای تنظیم کنیم که در عین حال که موانع را به خوبی حس می کند، نورهای محیطی را به عنوان مانع تشخیص ندهد.

سنسور دنبال کننده خط

اگر حساسیت سنسور تشخیص مانع مادون قرمز را کم کنیم، اشیاء روشن را تشخیص می دهد و خروجی سنسور LOW می شود، اما اشیاء تیره را تشخیص نمی دهد. بنابراین اگر روی صفحه سفید رنگی، خطی مشکی بکشیم، هنگامی که سنسور روی قسمت سفید قرار گیرد، خروجی آن LOW می شود و هر گاه روی خط مشکی قرار گیرد خروجی آن HIGH می شود. در عین حال سنسورهای مادون قرمزی نیز مخصوص دنبال کردن خط وجود دارند که می توانید از بازار خریداری کنید.

از جمله روشهای مرسوم ساخت ربات دنبال کننده خط این است که دو تا سنسور دنبال کننده خط را در جلوی ربات در دو طرف خط نصب می کنند. در حالت عادی که دو تا سنسور دنبال کننده خط در دو طرف خط قرار می گیرند، ربات رو به جلو می رود. اما هر گاه هر یک از سنسورهای دنبال کننده خط، روی خط مشکی قرار گیرد، خروجی سنسور مربوطه مقدار HIGH را بر می گرداند. در این حالت ربات چرخ همان سمتی که سنسورش روی خط قرار گرفته است را متوقف می کند و چرخ طرف دیگر را می‌چرخاند تا دوباره به وضعیتی قرار گیرد که دو سنسور در دو طرف خط قرار گیرند و ربات راه مستقیم را ادامه دهد.



الف. تصویر ربات تعقیب کننده خط در حالتی که ربات در مسیر است و دو سنسور در دو طرف خط قرار دارند

ب. تصویر ربات تعقیب کننده خط در حالتی که به راست منحرف شده است (در این حالت سنسور سمت چپ تحریک شده و بنابراین برای اینکه ربات به درون پیست برگردد می بایست چرخ چپ متوقف شود و چرخ راست بچرخد.)

شکل ۱۱- ۱۷: طریقه استفاده کردن از سنسورهای مادون قرمز در ربات دنبال کننده خط

مثال ۱۱-۶: بر روی سطح سفیدی مسیر بسته سیاه رنگی را رسم کرده ایم. با استفاده از آردوینو یک ربات تعقیب خط بسازید. یعنی رباتی بسازید که اگر ربات را روی خط بگذاریم و آن را روشن کنیم، بر روی خط مشکی به پیش رود.

پاسخ:

اگر سنسور سمت راست را به پایه ۱۱ و سنسور سمت چپ را به پایه ۱۲ متصل کرده باشیم، برنامه به صورت زیر می شود:

```
void setup() {
    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output

    pinMode(7,OUTPUT); // 7(out3) = output
    pinMode(8,OUTPUT); // 8(out4) = output

    pinMode(11,INPUT_PULLUP);
    pinMode(12,INPUT_PULLUP);
}

void loop() {
    if(digitalRead(11) == HIGH) {
    //Motor 1 (Right): clockwise (forward)
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low
    }
```

```
    if(digitalRead(12) == HIGH) {
//Motor2 (Left): counter clockwise (forward)
    digitalWrite(7,LOW);  //out3 = low
    digitalWrite(8,HIGH); //out4 = high
    }
    else{
    //Motor 1 (Right): clockwise (forward)
    digitalWrite(4,HIGH); //out1 = high
    digitalWrite(5,LOW);  //out2 = low
    }
}
```

توضیح: قسمت else از کد فوق برای اینست که مطمئن باشیم که هیچگاه هر دو موتور متوقف نمی شوند و اگر هر دو سنسور بر روی خط مشکی قرار گرفته بودند، چرخ راست به جلو برود و ربات از این وضعیت خارج شود.

تمرین

۱. برد آردوینو را به کامپیوتر متصل کنید. سپس برنامه ای بنویسید که هرگاه از طریق سریال حرف F را به آردوینو ارسال کردیم، ربات رو به جلو برود. هرگاه حرف B را به آردوینو فرستادیم ربات به عقب برود. هرگاه حرف R را ارسال کردیم ربات به طور درجا به سمت راست بچرخد و هر گاه حرف L را ارسال کردیم ربات به طور درجا به سمت چپ بچرخد. همچنین هر گاه حرف S را ارسال کردیم ربات توقف کند.

۲. در تمرین قبل به جلو و عقب ربات سنسور تشخیص مانع اضافه کنید و این امکان را فراهم کنید که زمانی که ربات به جلو می رود، اگر به مانعی رسید، ربات متوقف شود. همچنین هنگامی که به عقب می رود نیز اگر به مانعی رسید متوقف شود.

۳. یک کارخانه مایل است که تولیدات خود را قبل از عرضه به بازار، شستشو دهد. این محصولات بر روی تسمه نقاله ای جابجا می شوند. پایه ۵ از برد آردوینو به دستگاه شستشو دهنده متصل شده است به گونه ای که هرگاه پایه ۵ را HIGH کنیم، دستگاه شستشو دهنده روشن می شود و با پاشیدن مواد شوینده باعث تمیز شدن محصول تولیدی می شود. با استفاده از سنسور تشخیص مانع و برد آردوینو، دستگاه ساده ای بسازید که هرگاه محصولی روبروی سنسور تشخیص مانع قرار گرفت، پایه ۵ از برد آردوینو به مدت ٤ ثانیه، HIGH شود تا دستگاه شستشو روشن شود و محصول تولیدی را شستشو دهد.

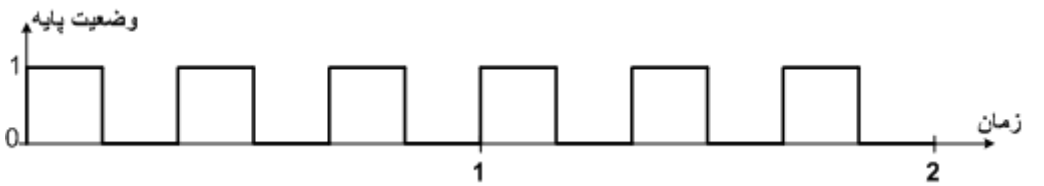
فصل ۱۲: خروجی آنالوگ و کنترل کردن سرعت ربات

بخش ۱۲-۱: آشنایی با اصطلاحات فرکانس و چرخه کاری (Duty Cycle)

فرکانس

احتمالا در مدرسه آزمایش ایجاد کردن صدا با خط کش پلاستیکی را انجام داده اید. هنگامی که خط کشی را در گوشه میز بگذاریم و سر خط کش را به پایین بکشیم، خط کش شروع به نوسان می کند. هرچه طول قسمتی از خط کش که نوسان می کند، کوتاه تر باشد، خط کش سریع تر نوسان می کند و صدای زیرتری دارد. اصطلاحا به تعداد دفعاتی که خط کش در هر ثانیه بالا و پایین رود، فرکانس گفته می شود و واحدی که برای فرکانس استفاده می شود هرتز است.

همانطوری که در فصلهای قبل دیده اید، در دنیای دیجیتال، پایه ها یا صفر (LOW) هستند و یا یک (HIGH). ما اگر پایه آردوینو را پشت سر هم با سرعتی ثابت روشن و خاموش کنیم، موجی به وجود می‌آید که همیشه مقدار آن یا صفر است و یا یک که به چنین موجی، موج مربعی گفته می شود. به تعداد دفعاتی که موج تغییر وضعیت می دهد و سپس به حالت اول خود بازمی گردد، فرکانس موج گفته می‌شود. به طور مثال، در شکل زیر فرکانس موج ۳ هرتز است. زیرا در هر ثانیه ۳ بار، پایه یک شده است و سپس به حالت صفر بازگشته است.



شکل ۱۲-۱: موج مربعی با فرکانس ۳ هرتز

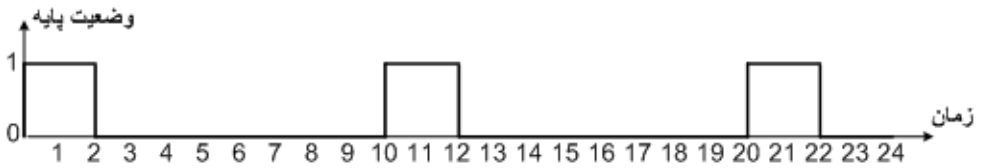
چرخه کاری

هنگامی که پایه ای را صفر و یک می‌کنیم، مدت زمانی که پایه صفر یا یک است الزاما با هم برابر نیستند. برای اینکه مقدار یک بودن موجی را بیان کنند از کمیتی به نام چرخه کاری (duty cycle) استفاده می کنند. برای اینکه چرخه کاری را حساب کنیم، زمانی که یک نوسان کامل طول می کشد را اندازه می‌گیریم و مدت زمانی که موج، یک (HIGH) است را نیز اندازه می گیریم. چرخه کاری برابر است با:

$$\text{چرخه کاری} = \frac{\text{مدت زمان یک بودن}}{\text{مدت زمانی که یک نوسان کامل طول می کشد}}$$

البته مرسوم تر است که چرخه کاری را به صورت درصد بیان کنند. مثلا در مورد وسیله ای که نصف مدت روشن (HIGH) باشد می‌گویند که این وسیله با چرخه کاری 50% کار می کند. به طور نمونه در شکل ۱۲-۲ موجهایی با چرخه کاری مختلف نمایش داده شده است. مثال زیر را ببینید.

مثال ۱۲-۱: اگر وسیله ای ۲ ثانیه روشن و ۸ ثانیه خاموش باشد، چرخه کاری آن را محاسبه کنید.

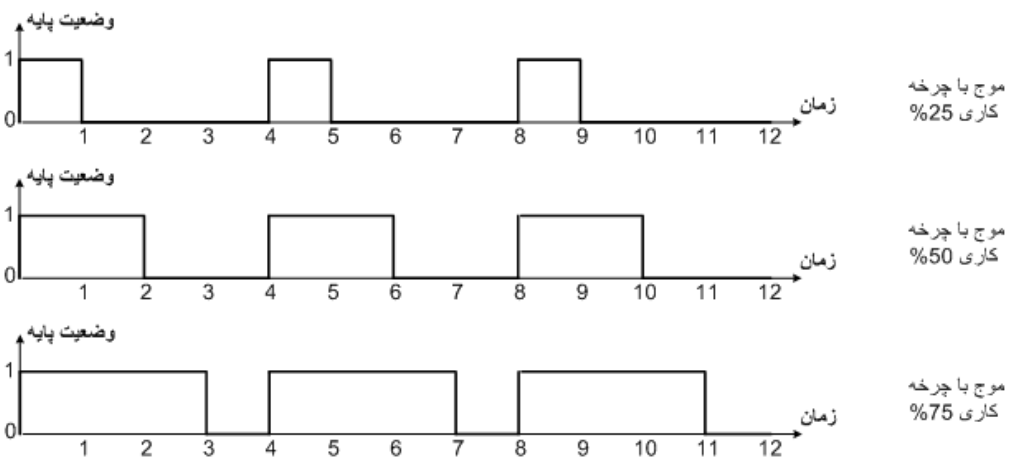


پاسخ:

کل بازه زمانی = مدت روشن بودن + مدت خاموش بودن = ۲ + ۸ = ۱۰

پس چرخه کاری برابر است با:

$$\text{Duty cycle} = \frac{t_{on}}{t_{total}} = \frac{2}{10} = 0.2 = 20\%$$



شکل ۱۲ - ۲: سه موج با چرخه های کاری ۲۵ % و ۵۰% و ۷۵ %

بخش ۱۲-۲:آشنایی با خروجی های آنالوگ (PWM)

یک بخاری برقی را در نظر بگیرید. اگر این بخاری نصف مدت روشن باشد و نصف مدت خاموش باشد، میزان حرارتی که ایجاد می کند نصف حالتی است که تمام این مدت روشن باشد. همچنین اگر این

بخاری یک ثانیه روشن و سه ثانیه خاموش باشد، میزان حرارتی که ایجاد می کند $1/4$ حرارتی است که اگر این بخاری دائما روشن باشد ایجاد می کند. به عبارت دیگر، با کم و زیاد کردن مدت زمانی که این بخاری روشن است می توانیم مقدار حرارتی که ایجاد می کند را کنترل کنیم. برای بیان کردن میزان روشن بودن وسایل برقی از کمیت چرخه کاری (Duty Cycle) استفاده می شود. به طور مثال اگر بخاری با چرخه کاری ۵۰٪ کار کند، حرارتی که ایجاد می کند به اندازه نصف زمانی که دائما روشن باشد (با چرخه کاری ۱۰۰٪ کار کند) حرارت ایجاد می کند.

در مورد بخاری برقی دیدیم که می توانیم با روشن و خاموش کردن بخاری، میزان حرارتی که ایجاد می کند را کنترل کنیم. اما آیا نور لامپ را نیز می توانیم با روشن و خاموش کردن کاهش دهیم؟ در نگاه اول به نظر می رسد که اگر لامپ را ۲ ثانیه روشن و یک ثانیه خاموش کنیم لامپ دائما روشن و خاموش می شود و مطلوب ما نخواهد بود. اما در عمل اگر لامپ با سرعت زیادی روشن و خاموش شود مثلا ۱۰۰ بار در ثانیه روشن و خاموش شود، چشم ما روشن و خاموش شدن لامپ را نخواهد دید و میزان نور لامپ به میزان روشن بودن لامپ وابسته خواهد بود. یعنی هر چه چرخه کاری (مدت روشن بودن نسبت به کل زمان) بیشتر باشد، لامپ روشن تر خواهد بود و هنگامی که تمام مدت لامپ روشن باشد، لامپ کاملا روشن خواهد بود.

پایه های PWM در آردوینو

در بردهای آردوینو پایه هایی که کنار شماره پایه علامت ~ وجود دارد، قابلیت PWM دارند؛ یعنی این پایه ها می توانند با هر چرخه کاری که ما بخواهیم روشن و خاموش شوند و با توجه به اینکه این پایه ها زمانی که در حالت PWM هستند متجاوز از ۴۹۰ بار در ثانیه روشن و خاموش می شوند، بنابراین اگر به لامپ یا وسایل برق دیگری متصل کنیم، روشن و خاموش شدن وسیله برقی قابل تشخیص نخواهد بود. در آردوینو برای اینکه میزان چرخه کاری پایه ای را معین کنیم از دستور زیر استفاده می کنیم:

```
analogWrite(pinNum,value);
```

در دستور فوق، pinNum شماره پایه ای است که می خواهیم میزان چرخه کاری آن را اعلام کنیم و عدد value میزان چرخه کاری را بیان می کند. البته عدد value بر حسب درصد نیست بلکه می بایست عددی بین ۰ تا ۲۵۵ باشد. عدد ۰ به معنای دائما خاموش است و به مرور هر چه میزان value بیشتر باشد چرخه کاری نیز بیشتر می شود تا اینکه به ازای ۲۵۵، پایه دائما روشن خواهد بود و چرخه کاری برابر با ۱۰۰٪ می شود. بین مقدار value و چرخه کاری موج ایجاد شده رابطه زیر وجود دارد:

چرخه کاری = value × 100 / 255

به طور نمونه، مثالهای زیر را ببینید.

مثال ۱۲-۲: دستوری بنویسید که پایه ۳ از برد آردوینو با چرخه کاری ۲۰% روشن و خاموش شود.

پاسخ:

ابتدا حساب می کنیم که به ازای چرخه کاری ۲۰% چه مقداری را می بایست به تابع analogWrite بدهیم:

duty cycle = value×100/255 ➔ value = duty cycle × 255 / 100 = 20×255/100 = 51

پس برای اینکه پایه ۳ با چرخه کاری ۲۰% روشن و خاموش شود، دستور زیر را می نویسیم:

```
analogWrite(3,51);
```

مثال ۱۲-۳: دستوری بنویسید که پایه ۹ از برد آردوینو با چرخه کاری ۹۵% روشن و خاموش شود.

پاسخ:

ابتدا حساب می کنیم که به ازای چرخه کاری ۹۵% چه مقداری را می بایست به تابع analogWrite بدهیم:

duty cycle = value×100/255 ➔ value = duty cycle × 255 / 100 = 95×255/100 = 242.25

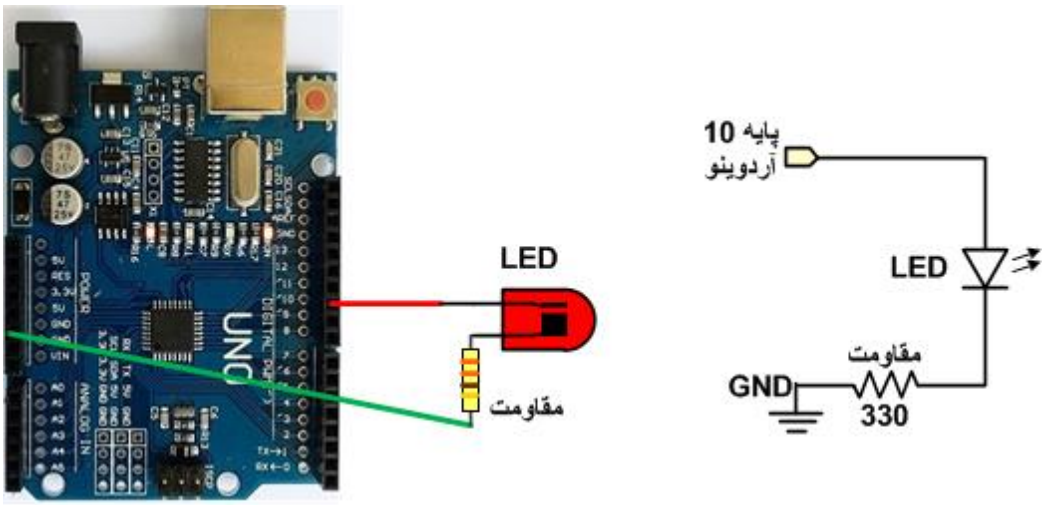
مقدار عددی که هنگام صدا زدن تابع analogWrite استفاده می کنیم می بایست عددی صحیح باشد. بنابراین قسمت اعشاری عدد فوق را حذف می کنیم و آن را به نزدیک ترین عدد روند می کنیم که در این مثال نزدیکترین عدد ۲۴۲ است. پس تابع analogWrite را به صورت زیر صدا می زنیم:

```
analogWrite(9,242);
```

مثال ۱۲-۴: به پایه ۱۰، یک LED قرمز رنگ متصل کنید. سپس برنامه ای بنویسید که نور LED به تدریج از خاموش به روشن تغییر کند و زمانی که LED کاملا روشن شد، نور آن به تدریج کم شود تا اینکه خاموش شود.

پاسخ:

با استفاده از یک مقاومت ۳۳۰ اهم، LED قرمز را به صورت زیر به پایه ۱۰ از برد آردوینو متصل میکنیم.



سپس با استفاده از حلقه for، برنامه ای می نویسیم که به مرور اعداد ۰ تا ۲۵۵ را به تابع analogWrite بدهد. هنگامی که اولین حلقه for به پایان رسید و به عدد ۲۵۵ رسیدیم، با استفاده از حلقه for دیگری به مرور از ۲۵۵ تا ۰ می شماریم تا شدت نور کاهش یابد و در نهایت به ۰ (خاموش) برسد.

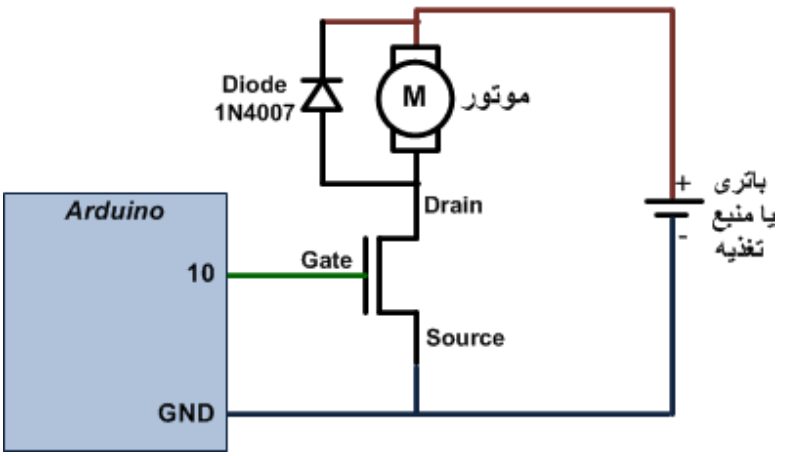
```
void setup() {
}

void loop() {
    int i;
    for(i = 0; i <= 255; i++) //count up from 0 to 255
    {
        analogWrite(10,i); //set duty cycle of pin 10
        delay(15);         //wait 15 millisecond
    }

    for(i = 255; i >= 0; i--) //count down from 255 to 0
    {
        analogWrite(10,i); //set duty cycle of pin 10
        delay(15);         //wait 15 millisecond
    }
}
```

فرکانس موج ایجاد شده در پایه های مختلف (برای علاقه مندان)

در آردوینو UNO، فرکانس پایه های ۵ و ۶، ۹۸۰ هرتز است و فرکانس در سایر پایه ها ۴۹۰ هرتز می‌باشد.

استفاده از PWM برای کنترل سرعت چرخش موتورهای دی سی

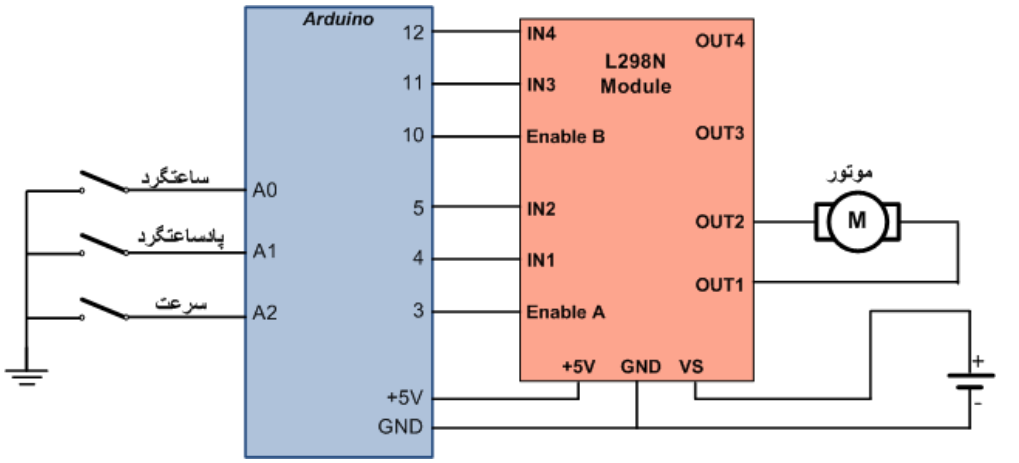
از PWM می توانیم برای کم و زیاد کردن سرعت چرخش موتورهای دی سی نیز استفاده کنیم. مثلا اگر یک موتور دی سی را به صورت زیر به آردوینو متصل کنیم و برنامه مثال ۱۲-٤ را بر روی آن بریزیم، موتور به مرور شروع به چرخش می کند و سرعت چرخش آن زیاد می شود تا اینکه به حداکثر سرعت خود می رسد. سپس سرعت موتور به مرور کاهش می یابد تا اینکه به مرور متوقف می شود.



شکل ۱۲-۳:کنترل دور موتور با PWM

همچنین هنگامی که موتور دی سی را از طریق L298 به آردوینو متصل می کنیم می توانیم با متصل کردن پایه های Enable به پایه های PWM سرعت چرخش موتور را کنترل کنیم. همان طوری که در فصل قبل دیدیم، هرگاه پایه Enable صفر باشد، L298 پایه خروجی خود را نسبت به مدار داخلی جدا می کند و هر موقع پایه Enable را یک کنیم، L298 پایه خروجی خود را به مدار داخلی متصل می کند. بنابراین هر چه چرخه کاری موج PWM بیشتر باشد، مدت زمانی که مدار داخلی L298 به پایه خروجی متصل است طولانی تر است و سرعت چرخش موتور بیشتر می شود. به طور نمونه مثال زیر را ببینید.

مثال ۱۲-۵: یک موتور دی سی را مطابق شکل به کمک ماژول L298 به برد آردوینو متصل کرده ایم. همچنین سه کلید به پایه های A0 و A1 و A2 متصل کرده ایم. برنامه ای بنویسید که اگر کلید A0 پایین باشد موتور ساعتگرد بچرخد. اگر کلید A1 پایین باشد موتور پادساعتگرد بچرخد و اگر کلیدهای A0 و A1 رها بودند، موتور متوقف باشد. همچنین از طریق کلید متصل به پایه A2، سرعت چرخش موتور را مشخص کنیم. اگر A2 فشرده بود، موتور سریع بچرخد و اگر رها بود موتور آهسته بچرخد.



پاسخ:

```
void setup()
{
    pinMode(A0,INPUT_PULLUP); //clockwise switch
    pinMode(A1,INPUT_PULLUP); //counter clock switch
    pinMode(A2,INPUT_PULLUP); //speed switch

    pinMode(4,OUTPUT); // 4(out1) = output
    pinMode(5,OUTPUT); // 5(out2) = output
}

void loop()
{
    if(digitalRead(A2) == 0)
    {
        analogWrite(3,255); //fast
    }
    else
    {
        analogWrite(3,200); //slow
    }
    if(digitalRead(A0) == 0) //clockwise
    {
        digitalWrite(4,HIGH); //out1 = high
        digitalWrite(5,LOW);  //out2 = low
    }
    else
    if(digitalRead(A1) == 0)  //counterclockwise
    {
        digitalWrite(4,LOW);  //out1 = low
        digitalWrite(5,HIGH); //out2 = high
    }
    else //stopped
    {
        digitalWrite(4,LOW);  //out1 = low
```

```
        digitalWrite(5,LOW);  //out2 = low
    }
}
```

تمرین

۱. برنامه ای بنویسید که پایه ۳ از برد آردوینو با چرخه کاری ۲۵٪ روشن و خاموش شود.

۲. اگر در برنامه ای نوشته شده باشد "analogWrite(9,120);" چرخه کاری موج ایجاد شده بر روی پایه ۹ تقریبا چند درصد است؟

فصل ۱۳: اندازه گیری زمان

بخش ۱۳-۱: آشنایی با تابع های micros و millis

هنگامی که آردوینو روشن می شود تایمری شروع به شمارش می کند. مقدار برگشتی تابع micros بیان کننده مدت زمان گذشته از روشن شدن آردوینو بر حسب میکرو ثانیه (میلیونیم ثانیه) است. تابع millis نیز مدت زمان سپری شده از روشن شدن آردوینو را با دقت میلی ثانیه (هزارم ثانیه) به ما می دهد. به طور نمونه برنامه زیر مدت زمانی که از روشن شدن آردوینو می گذرد را با دقت هزارم ثانیه (میلی ثانیه) به کامپیوتر ارسال می کند.

برنامه ۱۳-۱

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    unsigned long t = millis();
    Serial.print("T:");
    Serial.println(t);
    delay(1);
}
```

ممکن است این سوال برای شما مطرح شود که چه اهمیتی دارد که بدانیم که چه مدتی از روشن شدن آردوینو گذشته است و چه کاربردی برای ما دارد؟ دانستن مدتی که از روشن شدن آردوینو گذشته می‌تواند کاربردهای زیر را برای ما داشته باشد:

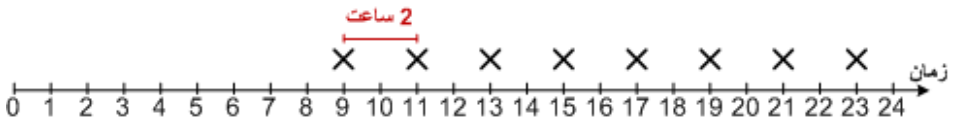
- اندازه گیری مدت زمان بین دو اتفاق
- انجام دادن کارها در بازه های زمانی مشخص

در ادامه به بررسی این کاربردها می پردازیم.

بخش ۱۳-۲: انجام کارها در بازه های زمانی مشخص

فرض کنید که کسی به دکتر مراجعه کرده باشد و دکتر گفته باشد که دارویی را هر ۲ ساعت یکبار مصرف کند و این شخص اولیه بار این دارو را ساعت ۹ صبح خورده باشد. چگونه ساعتهای بعدی که این شخص می‌بایست دارو را مصرف کند حساب می‌کنیم؟ کافی است که به ساعتی که آخرین بار دارو را به بیمار داده‌ایم، مدت بازه دارو دادن را اضافه کنیم تا زمان بعدی دارو خوردن به دست آید. چون که این

شخص آخرین بار ساعت ۹ دارو خورده است ۲ ساعت به ۹ می‌افزاییم که میشود ساعت ۱۱. پس می‌بایست صبر کنیم تا ساعت ۱۱ شود و وعده بعدی دارو را به او بدهیم. همچنین در ساعت ۱۱ نیز ۲ ساعت به زمان اضافه می‌کنیم تا ساعت ۱۳ به دست آید و ما به همین روال دارو دادن به بیمار را ادامه می‌دهیم.



شکل ۱۳-۱: ساعات خوردن دارو

در برنامه نویسی هم اگر قرار باشد که کاری را در بازه های ثابت انجام دهیم کافی است که با استفاده از تابع millis (یا micros) مدتی که از روشن شدن آردوینو گذشته است را بخوانیم و مدت بازه را به آن بیافزاییم تا زمان بعدی که می خواهیم آن کار انجام شود، به دست آید. مثلا در زیر نمونه برنامه‌ای نوشته شده که کاری را هر ۲ ثانیه یکبار انجام می‌دهد.

```
unsigned long nextTime = 0;
void loop() {
    unsigned long t = millis();
    if(t >= nextTime)
    {
        //کار مورد نظر را انجام بده
        nextTime = nextTime + 2000; // دو ثانیه اضافه کن
    }
    delay(1);
}
```

در برنامه فوق، متغیر nextTime حاوی زمان بعدی است که می بایست کار انجام شود. دستور if چک می‌کند که هر گاه زمان مورد نظر فرا رسیده بود، کار مورد نظر انجام شود و سپس به مقدار nextTime به اندازه مدت بازه ای که می خواهیم کاری انجام شود، اضافه می کنیم تا زمان بعدی که می بایست این کار انجام شود، به دست آید.

مثلا برنامه زیر هر ۲ ثانیه یکبار پایه ۱۳ را روشن و خاموش می کند.

برنامه ۱۳-۲

```
void setup() {
    pinMode(13,OUTPUT);
}

unsigned long nextTime = 0;
void loop() {
```

```
    unsigned long t = millis();
    if(t >= nextTime)
    {
        toggle13(); //پایه 13 را روشن و خاموش کن
        nextTime = nextTime + 2000; //دو ثانیه اضافه کن
    }
    delay(1);
}

int ledState = LOW;
void toggle13()
{
    if(ledState == LOW)
    {
        ledState = HIGH;
        digitalWrite(13,HIGH);
    }
    else
    {
        ledState = LOW;
        digitalWrite(13,LOW);
    }
}
```

در برنامه فوق تابع toggle13 نقش روشن و خاموش کردن پایه ۱۳ را برعهده دارد. ما می‌بایست بدانیم که آخرین بار پایه ۱۳ روشن شده یا خاموش شده تا از روی آن تصمیم بگیریم که الان می‌بایست پایه را روشن کنیم یا خاموش کنیم. بنابراین متغیری به نام ledState تعریف کرده ایم که همزمان با روشن و خاموش کردن پایه ۱۳ این متغیر را نیز مقداردهی می‌کنیم و با نگاه کردن به این متغیر پی می‌بریم که آخرین بار پایه را روشن یا خاموش کرده ایم.

البته برنامه فوق را می توانیم با استفاده از delay به صورت زیر نیز بنویسیم و بنابراین ممکن است این سوال برای شما مطرح شود که استفاده کردن از millis چه فایده ای دارد.

```
void loop()
{
    digitalWrite(13,HIGH);
    delay(2000);
    digitalWrite(13,LOW);
    delay(2000);
}
```

مزیت استفاده کردن از millis زمانی آشکار می شود که بخواهیم چندین کار را در بازه های زمانی مختلف انجام دهیم. مثلا اگر قرار باشد که پایه ۱۲ را هر ۲ ثانیه یکبار و پایه ۱۳ را هر ۳ ثانیه یکبار روشن و خاموش کنیم، به سهولت می توانیم برنامه نوشته شده با millis را به صورت زیر گسترش دهیم.

برنامه ۱۳-۳

```
void setup() {
    pinMode(12,OUTPUT);
    pinMode(13,OUTPUT);
}

unsigned long nextTime1 = 0;
unsigned long nextTime2 = 0;
void loop() {
    unsigned long t = millis();
    if(t >= nextTime1)
    {
        toggle12();
        nextTime1 = nextTime1 + 2000; // دو ثانیه اضافه کن
    }

    if(t >= nextTime2)
    {
        toggle13();
        nextTime2 = nextTime2 + 3000; // سه ثانیه اضافه کن
    }
    delay(1);
}

int led12State = LOW;
void toggle12()
{
    if(led12State == LOW)
    {
        led12State = HIGH;
        digitalWrite(12,HIGH);
    }
    else
    {
        led12State = LOW;
        digitalWrite(12,LOW);
    }
}

int led13State = LOW;
void toggle13()
{
    if(led13State == LOW)
    {
```

```
        led13State = HIGH;
        digitalWrite(13,HIGH);
    }
    else
    {
        led13State = LOW;
        digitalWrite(13,LOW);
    }
}
```

در برنامه زیر هر ۲ ثانیه یکبار پیغام Hello را از طریق سریال به کامپیوتر می فرستیم و هر ۵ ثانیه یکبار پایه ۱۳ را روشن و خاموش می‌کنیم.

برنامه ۱۳-۴

```
void setup() {
    Serial.begin(9600);
    pinMode(13,OUTPUT);
}

unsigned long nextTime1 = 0;
unsigned long nextTime2 = 0;
void loop() {
    unsigned long t = millis();
    if(t >= nextTime1)
    {
        Serial.println("Hello");
        nextTime1 = nextTime1 + 2000; // دو ثانیه اضافه کن
    }

    if(t >= nextTime2)
    {
        toggle13();
        nextTime2 = nextTime2 + 5000; // پنج ثانیه اضافه کن
    }
    delay(1);
}

int led13State = LOW;
void toggle13()
{
    if(led13State == LOW)
    {
        led13State = HIGH;
        digitalWrite(13,HIGH);
    }
    else
    {
```

```
        led13State = LOW;
        digitalWrite(13,LOW);
    }
}
```

همان طوری که می بینید این برنامه هایی که با استفاده از millis نوشته ایم دارای ساختار یکسانی هستند و هر موقع بخواهیم کارهایی را در بازه های زمانی مشخصی انجام دهیم می توانیم از این ساختار استفاده کنیم.

بخش ۱۳-۳: اندازه گیری زمان بین دو اتفاق

در این بخش می خواهیم فاصله زمانی بین دو اتفاق را اندازه بگیریم. مثلا دونده ای را در نظر بگیرید که از خط شروع تا پایان می دود و ما می خواهیم اندازه بگیریم که دونده از خط شروع تا پایان را ظرف چند ثانیه دویده است. تصور کنید که در خط شروع یک سنسور چشمی وجود دارد که هرگاه دونده از روبروی آن عبور می کند پایه ٤، زمین (LOW) می شود و سنسوری نیز در خط پایان وجود دارد که هر گاه دونده از آن عبور می کند پایه ٥، زمین (LOW) می شود. اگر بخواهیم مدتی که طول کشیده که دونده از خط شروع تا پایان برود را بر حسب صدم ثانیه اندازه بگیریم، می توانیم برنامه ای به شکل زیر بنویسیم:

برنامه ۱۳-۵: محاسبه فاصله زمانی با استفاده از تابع delay

```
void setup() {
    pinMode(4,INPUT_PULLUP);
    pinMode(5,INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    while(digitalRead(4) == HIGH) //wait here until the pin becomes low
    {
    }

    Serial.println("Started");

    unsigned int t = 0;
    //while pin 5 is HIGH, increase t every 0.01s
    while(digitalRead(5) == HIGH)
    {
        delay(10); //wait 0.01 sec
        t = t + 1; //increase t

        Serial.println(t);
    }
}
```

در برنامه فوق، ابتدا در یک حلقه while صبر می کنیم تا دونده از خط شروع بگذرد. (پایه ٤، زمین شود.) سپس تا مادامی که پایه ٥، زمین نشده است (از خط پایان نگذشته است) مقدار متغیر t را هر 0.01 ثانیه زیاد می کنیم. بنابراین در پایان برنامه متغیر t حاوی مدت زمانی است که طول کشیده تا دونده از خط شروع تا خط پایان برود و از آنجایی که در تابع delay به اندازه 0.01 ثانیه صبر کرده ایم، پس مقدار متغیر t هر 0.01 ثانیه زیاد می شود.

در برنامه فوق، مقداری از وقت CPU برای اجرا شدن دستورات println و while و افزایش t، سپری می شود و بنابراین مدت زمانی که طول می کشد تا یکبار حلقه while اجرا شود، کمی بیشتر از یک صدم ثانیه است و بنابراین زمانی که برنامه فوق اندازه گیری می کند زیاد دقیق نیست. البته مقدار تاخیری که تابع delay ایجاد می کند نیز زیاد دقیق نیست که این موضوع نیز به عدم دقت برنامه فوق می افزاید.

راه حل دیگری که در این موارد وجود دارد استفاده کردن از تابع micros است. اگر بخواهیم برنامه فوق را با استفاده از تابع micros بازنویسی کنیم، می توانیم هنگامی که شروع به دویدن می کند، مقدار micros را بخوانیم و هنگامی که از خط پایان می گذرد نیز مقدار micros را بخوانیم. تفاضل مقادیر خوانده شده از micros مشخص می کند که چند میکرو ثانیه طول کشیده که دونده از خط شروع تا خط پایان برود.

برنامه ۱۳-۶: اندازه گیری فاصله زمانی با استفاده از تابع micros

```
void setup() {
    pinMode(4,INPUT_PULLUP);
    pinMode(5,INPUT_PULLUP);

    Serial.begin(9600);
}

void loop() {
    while(digitalRead(4) == HIGH) //wait here until the pin becomes low
    {
    }

    unsigned long t1 = micros();

    Serial.println("Started");

    while(digitalRead(5) == HIGH) //while pin 5 is HIGH, wait here
    {
    }
```

```
    unsigned long t2 = micros();
    Serial.println(t2 - t1);
}
```

در برنامه فوق، ابتدا صبر می کنیم تا پایه ٤، زمین (LOW) شود. سپس مقدار تابع micros را می‌خوانیم و داخل متغیر t1 می‌ریزیم. سپس صبر می‌کنیم تا پایه ٥، زمین (LOW) شود. هنگامی که پایه ٥، زمین شود، از حلقه while بیرون می‌آید و مقدار تابع micros را درون متغیر t2 می‌ریزد. بنابراین t1 بیان کننده زمان شروع دویدن است و مقدار t2 بیان کننده زمان عبور از خط پایان است و تفاضل این دو زمان بیانگر مدت زمانی است که از خط شروع تا خط پایان برود. در برنامه فوق، اگر بخواهیم زمان نمایش داده شده خواناتر باشد می‌توانیم حاصل تفاضل را بر یک میلیون تقسیم کنیم تا زمان طول کشیده را بر حسب ثانیه نمایش دهیم. برای این کار، می‌توانیم به جای خط آخر برنامه فوق، دستورات زیر را بنویسیم:

```
unsigned long t2 = micros();
float result = (t2 - t1) / 1000000.0;
Serial.println(result);
```

اندازه گیری فاصله بین دو اتفاق کاربردهای فراوانی دارد که از جمله آنها اندازه گیری پهنای پالس و اندازه گیری فرکانس موجها است. همچنین هنگام کار کردن با برخی از سنسورها از جمله سنسور ماوراء صوت و سنسور رطوبت نیز نیاز به اندازه گیری فاصله زمانی داریم.

بخش ۱۳-۴: سنسور اولتراسونیک (ماوراء صوت) و اندازه گیری فاصله

فرض کنید که ما توپی را به سمت دیواری که روبروی ماست پرتاب کنیم. این توپ به دیوار برخورد می‌کند و پس از مدتی به سمت ما باز می‌گردد. هر چه فاصله ما تا دیوار کمتر باشد زودتر توپ به سمت ما باز می‌گردد. پس فاصله ما با دیوار با مدت زمانی که طول می‌کشد تا توپ به ما برسد رابطه مستقیم دارد.

هنگامی که ما صدایی ایجاد کنیم نیز این صدا از ما شروع به دور شدن می کند و اگر در طی راه به مانعی برخورد کند مانع باعث بازتاب صدا می شود و صدا به سمت ما باز می گردد. هر چه مانع به ما نزدیک تر باشد، صدا زودتر به ما می رسد. از همین قاعده، خفاشها در تاریکی برای شناسایی موانع اطراف خود استفاده می کنند. بدین صورت که خفاشها صدایی را از خود ایجاد می کنند و از روی مدتی که طول می کشد تا بازتاب صدا به آنها برسد موانع اطراف خود را شناسایی می کنند.

ما نیز هنگام استفاده از سنسور اولتراسونیک (ماوراء صوت) از همین ایده استفاده می کنیم تا فاصله ربات از موانع را به دست آوریم. سنسور اولتراسونیک از یک بلندگو و یک میکروفن تشکیل شده است. ابتدا با استفاده از بلندگو صدایی را ایجاد می کنیم و سپس صبر می کنیم تا بازتاب صدا به میکروفن برسد. از روی مدت زمانی که طول می کشد تا صدای بازتاب شده شنیده شود. فاصله را محاسبه می کنیم. در زیر عکس یک سنسور اولتراسونیک را مشاهده می کنید.

شکل ۱۳-۲: عکس HC-SR04

طریقه محاسبه فاصله ربات از مانع از روی مدتی که بازتاب صدا شنیده می شود

هنگامی که وسیله ای با سرعت ثابتی در حال حرکت است، از ضرب کردن سرعت در زمان می توان مقدار مسافتی که شی حرکت می‌کند را به دست آورد. مثلا اگر ما به مدت ۲ ساعت، با سرعت ۱۰۰ کیلومتر در جاده ای حرکت کنیم، مقدار مسافتی که ما طی می کنیم ۲۰۰ کیلومتر است. زیرا با ضرب کردن سرعت در مدت زمانی که اتومبیل حرکت می کند خواهیم داشت:

$$2 \times 100 = 200$$

به طور کلی اگر وسیله ای به مدت زمان t، با سرعت V، حرکت کند، مقدار جابجایی شی برابر است با:

$$\text{جابجایی شی} = V \times t$$

سرعت حرکت صدا حدودا ۳٤۰ متر بر ثانیه است و بنابراین مقدار مسافتی که صدا در طی مدت t ثانیه حرکت می کند برابر است با:

340×t = مسافتی که صدا جلو می رود

بنابراین اگر از زمانی که بلندگوی سنسور اولتراسونیک روشن می شود تا زمانی که بلندگو بازتاب صدا را می شنود، t ثانیه طول بکشد، مقدار مسافتی که صدا در این مدت طی کرده است برابر با 340×t متر است.

هنگامی که با استفاده از سنسور اولتراسونیک صدایی ایجاد می کنیم، این صدا به سمت مانع حرکت می کند و پس از برخورد با مانع باز می گردد. به عبارت دیگر، صدا ابتدا فاصله بین سنسور اولتراسونیک تا مانع را می رود و سپس باز می گردد و بنابراین طول مسافتی که صدا طی می کند، دو برابر فاصله بین سنسور اولتراسونیک و مانع است. بنابراین:

340×t/2 = 170×t = فاصله سنسور اولتراسونیک از مانع بر حسب متر

که در رابطه فوق فاصله بر حسب متر و زمان بر حسب ثانیه است. اگر بخواهیم فاصله را بر حسب سانتی متر به دست آوریم با توجه به اینکه هر متر ۱۰۰ سانتی متر است پس می بایست عبارت فوق را در ۱۰۰ ضرب کنیم:

17000×t = فاصله سنسور اولتراسونیک از مانع بر حسب سانتی متر

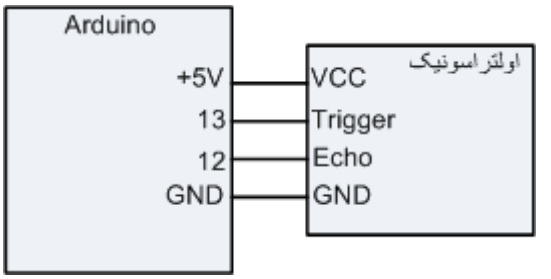
و اگر بخواهیم زمان را بر حسب میکروثانیه بیان کنیم، با توجه به اینکه هر میکرو ثانیه، یک میلیونیم ثانیه است، پس خواهیم داشت:

17000×t /1000000 = 0.017t = فاصله سنسور اولتراسونیک از مانع بر حسب سانتی متر

متصل کردن سنسور اولتراسونیک به آردوینو

سنسور اولتراسونیک ٤ تا پایه دارد: GND، Echo، Trigger و VCC. پایه های GND و VCC به ترتیب به پایه های GND و 5V+ از برد آردوینو متصل می شوند و برق مورد نیاز جهت کار کردن سنسور اولتراسونیک را برایش فراهم می کنند. پایه Trig را هر گاه یک کنیم، بلندگوی سنسور اولتراسونیک صدایی را ایجاد می کند و البته چون فرکانس صدایی که ایجاد می کند بیشتر از ۲۰ کیلوهرتز است گوش ما صدای آن را نمی شنود. هنگامی که صدای سنسور اولتراسونیک به مانع برخورد می کند و بازمی گردد، گیرنده سنسور اولتراسونیک، بازتاب صدا را حس می کند و پایه Echo صفر می شود. به عبارت دیگر پایه Echo در حالت عادی یک است و هنگامی که گیرنده اولتراسونیک بازتاب صدا را حس می کند صفر می شود.

در شکل زیر، یک سنسور اولتراسونیک را به آردوینو متصل کرده ایم و در زیر برنامه ای نوشته ایم که فاصله تا اولین مانع را اندازه می گیرد و فاصله را بر حسب سانتی متر به کامپیوتر ارسال می کند.



شکل ۱۳-۳: متصل کردن سنسور اولتراسونیک به آردوینو

برنامه ۱۳-۷

```
void setup() {
    Serial.begin(9600);
    pinMode(13,OUTPUT); //trigger
    pinMode(12,INPUT_PULLUP); //echo
}

void loop() {
    digitalWrite(13,HIGH);
    delayMicroseconds(10);
    digitalWrite(13,LOW);

    while(digitalRead(12) == LOW);

    unsigned long t0 = micros();

    while(digitalRead(12) == HIGH);

    unsigned long t1 = micros();

    float distance = 0.017*(t1 - t0);
    if(t1-t0 > 25000)
    {
        Serial.println("out of range!");
    }
    else
    {
        Serial.print(distance);
        Serial.println(" cm");
    }
    delay(100);
}
```

در برنامه فوق، در حلقه loop، ابتدا پایه trigger را به مدت ۱۰ میکرو ثانیه، فعال (یک) می کنیم تا صدایی ایجاد شود. سپس صبر می کنیم تا بازتاب صدا حس شود و در انتها فاصله سنسور از مانع را محاسبه می کنیم.

در برنامه فوق ابتدا زمانی که پایه trigger را تحریک کرده ایم مقدار micros را می خوانیم و سپس زمانی که بازتاب صدا حس شد نیز micros را می خوانیم و تفاضل آنها را حساب کنیم تا بدین وسیله مدت زمانی که طول کشیده تا بازتاب صدا حس شود را به دست آوریم.

نکته دیگری که در برنامه فوق وجود دارد، تابع delay(100) موجود در انتهای برنامه است. هنگامی که صدایی در محیط پخش می شود درون محیط به موانع مختلفی برخورد می کند و با زمانهای مختلفی به ما می رسد. ما اولین بازتابی که حس می شود را به عنوان فاصله خود تا نزدیک ترین مانع در نظر می‌گیریم. اما تا مدتی پس از آن ممکن است همچنان بازتاب صدا به سمت سنسور بیاید و آن را تحریک کند. بنابراین پس از هر بار که عمل فاصله سنجی را انجام می دهیم لازم است که دست کم ٦٠ میلی ثانیه صبر کنیم تا بازتاب صداها در محیط تمام شود و سپس می توانیم مجددا عمل فاصله سنجی را انجام دهیم.

نوشتن برنامه اولتراسونیک با استفاده از تابع pulseIn (اختیاری)

در آردوینو تابعی به نام pulseIn وجود دارد که با استفاده از آن می توان مدت زمانی که پایه ای از آردوینو در وضعیت مشخصی قرار دارد را اندازه گرفت. ساختار آرگومانهای تابع pulseIn به صورت زیر است:

```
pulseIn(pinNumber,level,timeout);
```

که در دستور فوق pinNumber، شماره پایه ای از آردوینو است که می خواهیم وضعیت آن زیر نظر قرار گرفته شود. از طریق level اعلام می کنیم که می خواهیم مدت یک بودن (HIGH) پایه را اندازه بگیرد یا اینکه مدت صفر بودن آن را اندازه بگیرد. از طریق timeout اعلام می کنیم که مایلیم حداکثر چقدر صبر کند که پایه تغییر وضعیت دهد. به طور مثال دستور زیر اعلام می کند که پایه ۱۲ را زیر نظر قرار دهد و مدت HIGH بودن پایه را اندازه بگیرد و حداکثر ۲۵۰۰۰ میکروثانیه برای LOW شدن پایه ۱۲ صبر کند:

```
unsigned int t = pulseIn(12,HIGH,25000);
```

بنابراین دستور فوق، صبر می کند تا پایه ۱۲، یک (HIGH) شود. سپس مدت زمانی که طول می کشد تا LOW شود را بر حسب میکروثانیه به عنوان مقدار برگشتی برمی گرداند. اما اگر ظرف ۲۵۰۰۰ میکروثانیه LOW نشد، مقدار ۰ را برمی گرداند.

در برنامه زیر، برنامه اولتراسونیک با استفاده از تابع pulseIn بازنویسی شده است.

برنامه ۱۳-۸

```
void setup() {
    pinMode(13,OUTPUT);
    pinMode(12,INPUT);

    Serial.begin(9600);
}

void loop() {
    digitalWrite(13,HIGH); //enable the trigger pin to make a sound
    delayMicroseconds(10); //wait 10us to make sure the sound is made
    digitalWrite(13,LOW);  //disable the trigger pin (stop making sound)

    unsigned int t = pulseIn(12,HIGH,25000);
    if(t == 0)
        Serial.println("out of range");
    else
    {
        float distance = 0.017*t;
        Serial.print(distance);
        Serial.println("cm");
    }
    delay(100);
}
```

دستور pulseIn که در برنامه فوق استفاده شده است، مدت زمانی که طول می کشد تا پایه echo، صفر شود را بر حسب میکروثانیه اندازه می گیرد و به عنوان مقدار برگشتی بر می گرداند.

در برنامه فوق هرگاه بازتاب صدا توسط سنسور اولتراسونیک حس شود، پایه ۱۲ صفر می شود. بنابراین مقدار برگشتی تابع pulseIn در برنامه فوق به ما می گوید که چه مدت طول کشیده تا بازتاب صدا حس شود.

با توجه به اینکه تابع pulseIn، آرگومان timeOut دارد، بنابراین در برنامه ای که با استفاده از pulseIn نوشته شده است، حداکثر مدتی که سنسور صبر می کند تا بازتاب صدا شنیده شود را می توانیم محدود

کنیم. عملا سنسورهای اولتراسونیک طول برد محدودی دارند (سنسورهای اولتراسونیک موجود در بازار معمولا طول برد ٤ متر را دارند.) بنابراین می توانیم حداکثر مدت زمانی که برای بازتاب صدا صبر می‌کنیم را محدود کنیم تا بدین وسیله سرعت اندازه گیری سنسور اولتراسونیک را افزایش دهیم.

تمرین

۱. یک سنسور تشخیص مانع جلوی ربات نصب کرده ایم. برنامه ای بنویسید که ربات به جلو برود تا به مانعی برسد. سپس به همان مدتی که جلو آمده است به عقب بازگردد تا به نقطه آغاز خود برسد.

۲. سنسور اولتراسونیک را به ربات و برد آردوینو متصل کنید. سپس برنامه ای بنویسید که ربات رو به جلو برود. هر گاه فاصله ربات تا مانع کمتر از ۳۵ سانتی متر بود، به طور درجا به سمت چپ بچرخد تا فاصله اش تا مانع بیشتر از ٤۰ سانتی متر شود و سپس رو به جلو حرکت کند.

۳. با استفاده از تابع millis برنامه ای بنویسید که هر ۲ ثانیه یکبار پایه ۱۳ را روشن و خاموش کند و هر ۵ ثانیه یکبار پیغام Hello را از طریق سریال به کامپیوتر ارسال کند.

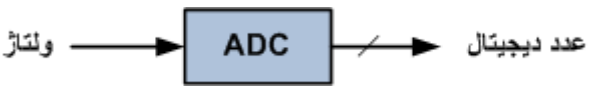
۴. می خواهیم هرگاه دکمه چراغ راه پله را می زنیم، به مدت یک دقیقه چراغ راه پله روشن باشد و اگر در طی مدتی که چراغ روشن است دکمه چراغ را زدیم، اندازه گیری یک دقیقه زمان از ابتدا شروع شود. برای این منظور به یکی از پایه های آردوینو خود کلیدی متصل کنید (اگر کلید ندارید از یک تکه سیم استفاده کنید.) و با استفاده از تابع millis برنامه ای بنویسید که هرگاه کلید زده شد، پایه ۱۳ از آردوینو به مدت یک دقیقه روشن باشد و بعد از یک دقیقه خاموش شود.

فصل ۱۴: مبدل آنالوگ به دیجیتال (ADC) و سنسورهای آنالوگ

بخش ۱۴-۱: آشنایی با ADC (Analog to Digital Converter)

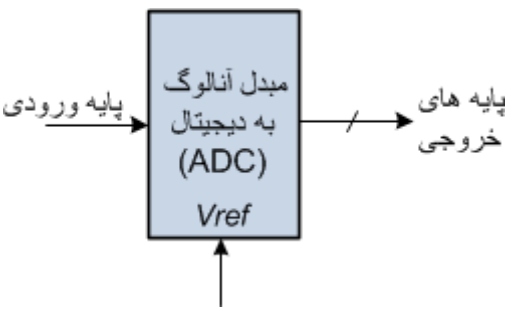
مقدمه

در دنیای واقعی مقادیر ولتاژ می‌تواند هر مقداری باشد. مثلا میکروفنی که در معرض سر و صدا قرار دارد ولتاژی ایجاد می‌کند که مقدار آن دائما تغییر می‌کند. اما در دنیای دیجیتال ولتاژها دو حالت دارند یا صفر منطقی هستند یا یک منطقی و مدارهای دیجیتال فقط قادرند که با ولتاژهای صفر و یک منطقی کار کنند. بنابراین برای اینکه ولتاژهای آنالوگ توسط مدارهای دیجیتال قابل تحلیل باشند، می بایست به اعدادی باینری تبدیل شوند. برای تبدیل کردن ولتاژهای آنالوگ به اعداد باینری از مبدل آنالوگ به دیجیتال (ADC) استفاده می‌کنیم.



شکل ۱۴-۱: مبدل آنالوگ به دیجیتال (ADC)

ADCها پایه ای ورودی دارند که ولتاژ آن را اندازه گیری می‌کنند و به ازای آن عددی باینری در پایه های خروجی خود ایجاد می‌کنند. ADCها همچنین پایه ای به نام Vref دارند که حداکثر ولتاژی که ممکن است پایه ورودی داشته باشند را به آن اعمال می‌کنیم. شکل زیر را ببینید.



شکل ۱۴-۲: پایه های ADC

تعداد پایه‌های خروجی در ADCهای مختلف فرق می‌کند و معمولا ADCها بر حسب تعداد پایه‌های خروجی شان نامگذاری می‌شوند. مثلا اگر در یک ADC هشت تا پایه خروجی وجود داشته باشد، آن را ADC هشت بیتی می‌نامند و هنگامی که ADC، ۱۰ تا پایه خروجی داشته باشد آن را ADC ده بیتی می‌نامند.

هنگامی که تعداد پایه های خروجی n بیت باشد، هر یک از پایه‌های خروجی ممکن است دو حالت مختلف داشته باشند (۰ یا ۱ باشند.) بنابراین پایه های خروجی می توانند 2^n حالت مختلف داشته باشند. شکل زیر را ببینید.

$$\underbrace{2 \times 2 \times 2 \times \ldots \times 2}_{\text{n مرتبه}} = 2^n$$

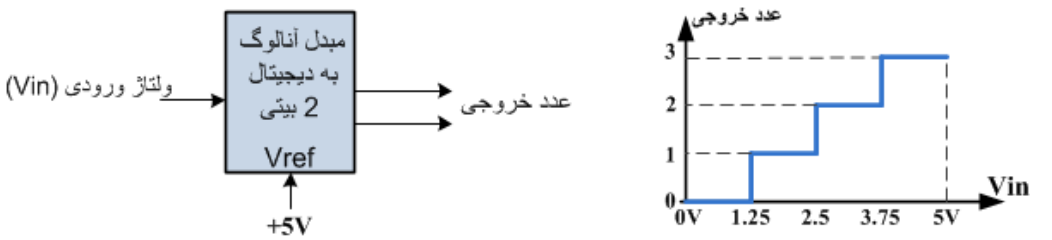
شکل ۱۴-۳: با n تا پایه خروجی 2^n عدد مختلف را می توانیم بیان کنیم

بنابراین ADC ولتاژ بین ۰ تا Vref را به 2^n بازه تقسیم می‌کند و بر حسب اینکه ولتاژ ورودی در چه بازه‌ای قرار بگیرد، عدد متناظر با بازه را در خروجی ایجاد می‌کند.

مثلا اگر خروجی یک ADC، ۲ بیتی باشد، پایه‌های خروجی 2^2 (یعنی ٤) حالت مختلف می‌توانند داشته باشند، بنابراین ولتاژ بین ۰ تا Vref را به ٤ بازه تقسیم می‌کند:

- بازه ۰ تا Vref/4
- بازه Vref/4 تا Vref/2
- بازه Vref/2 تا 3*Vref/4
- بازه 3*Vref/4 تا Vref

اگر ولتاژ ورودی در بازه ۰ تا Vref/4 باشد، مبدل در خروجی خود عدد ۰۰ باینری (که معادل ۰ در مبنای ده است) را ایجاد می‌کند. اگر ولتاژ ورودی در بازه Vref/4 تا Vref/2 باشد، مبدل در خروجی خود عدد ۰۱ باینری (که معادل ۱ در مبنای ده است) را ایجاد می‌کند. اگر ولتاژ ورودی در بازه Vref/2 تا 3*Vref/4 باشد، مبدل در خروجی عدد ۱۰ باینری (که معادل ۲ است) را ایجاد می‌کند. اگر ولتاژ ورودی از 3*Vref/4 به بالا باشد، مبدل در خروجی عدد ۱۱ باینری (که معادل ۳ است) را ایجاد می‌کند. شکل زیر را ببینید.



شکل ۱۴-۴: رابطه بین ولتاژ ورودی و عدد خروجی در یک مبدل ۲بیتی که پایه Vref آن به ۵ ولت متصل شده

مفهوم پله

اگر به شکل ١٤-٤ نگاه کنید می بینید که نمودار رابطه بین ولتاژ ورودی و عدد خروجی به شکل پله است. به همین خاطر به هر بازه اصطلاحا یک پله می‌گویند. هر چه تعداد پایه های خروجی بیشتر باشد، تعداد بازه ها (پله ها) بیشتر می‌شود و اندازه پله ها (بازه‌ها) کوچکتر می‌شود. مبدل آنالوگ به دیجیتال، به ازای جمیع ولتاژهایی که درون یک پله قرار بگیرند، عدد یکسانی را در خروجی خود ایجاد می‌کند.

رابطه ریاضی بین ولتاژ ورودی ADC و عدد خروجی آن

گفتیم که مبدل آنالوگ به دیجیتال فاصله بین ۰ ولت تا Vref را به 2^n بازه مساوی تقسیم می‌کند. بنابراین اندازه هر پله (اندازه بازه) برابر است با:

$$\text{اندازه پله} = \frac{Vref}{2^n}$$

در عمل مبدل آنالوگ به دیجیتال در خروجی خود اعلام می‌کند که ولتاژ ورودی در چندین پله قرار می‌گیرد. بنابراین رابطه بین ولتاژ ورودی و عدد خروجی به صورت زیر است:

$$\text{عدد خروجی} = \frac{\text{ولتاژ ورودی}}{\text{اندازه پله}}$$

توجه دارید که خروجی مبدل آنالوگ به دیجیتال عددی صحیح است. بنابراین در رابطه فوق اگر مقدار عدد خروجی، اعشاری به دست آمد، قسمت اعشاری را دور می‌ریزیم و فقط قسمت صحیح عدد را در نظر می‌گیریم. مثلا اگر عدد خروجی 2.35 در بیاید، می‌دانیم که در بازه دوم قرار دارد و بنابراین عدد خروجی مبدل آنالوگ به دیجیتال ۲ است.

به طور نمونه، مثال زیر را ببینید.

مثال ۱۴-۱

در یک مبدل آنالوگ به دیجیتال ۸ بیتی، پایه Vref به ۲,۵۶ ولت متصل شده است. حساب کنید که:

الف. اندازه پله چقدر است؟

ب. اگر ولتاژ ۰,۲ ولت را به پایه ورودی این ADC متصل کنیم، در خروجی چه عددی را می دهد؟

پ. اگر ولتاژ ورودی ۱,۵ ولت باشد، عدد خروجی چند است؟

ت. اگر ولتاژ ورودی ۵ ولت باشد، چه عددی در خروجی ظاهر می شود؟

ث. اگر این ADC، در خروجی خود، عدد ۶۳ را بیرون بدهد، ولتاژ پایه ورودی چند ولت بوده است؟

پاسخ:

الف. با توجه به اینکه مبدل ۸ بیتی است و Vref به ۲,۵۶ ولت متصل شده است، پس اندازه پله برابر است با:

$$\text{اندازه پله} = \frac{Vref}{2^n} = \frac{2.56}{2^8} = \frac{2.56}{256} = 0.01$$

ب. رابطه زیر بیرون ولتاژ ورودی و عدد خروجی وجود دارد:

$$\text{عدد خروجی} = \text{ولتاژ ورودی} / \text{اندازه پله} = \text{ولتاژ ورودی} / 0.01 \times 100$$

بنابراین زمانی که ولتاژ ورودی ۰,۲ ولت باشد، عدد خروجی برابر است با:

$$\text{عدد خروجی} = \text{ولتاژ ورودی} \times 100 = 0.2 \times 100 = 20$$

پ. عدد خروجی برابر است با:

$$\text{عدد خروجی} = \text{ولتاژ ورودی} \times 100 = 1.5 \times 100 = 150$$

ت. با توجه به اینکه ولتاژ ورودی بزرگتر از Vref است، بنابراین بزرگترین عدد ممکن که همان 255 است را در خروجی خود اعمال می کند.

ث. وقتی که عدد ۶۳ در خروجی ADC ظاهر شود، یعنی ولتاژ در شصت و سومین بازه قرار گرفته است. ولتاژ ابتدای شصت و سومین بازه برابر است با:

$$\text{ولتاژ} = \text{اندازه پله} \times \text{شماره پله} = 0.01 \times 63 = 0.63\ V$$

توضیح: ADC به ازای جمیع ولتاژهایی که در یک پله قرار می‌گیرند، عدد یکسانی را تولید می‌کند. با توجه به اینکه اندازه پله 0.01 است بنابراین به ازای ولتاژهای بین 0.63 ولت تا 0.64 ولت، ADC در خروجی خود عدد ۶۳ را می‌دهد.

بخش ۱۴-۲: مبدل آنالوگ به دیجیتال در برد آردوینو

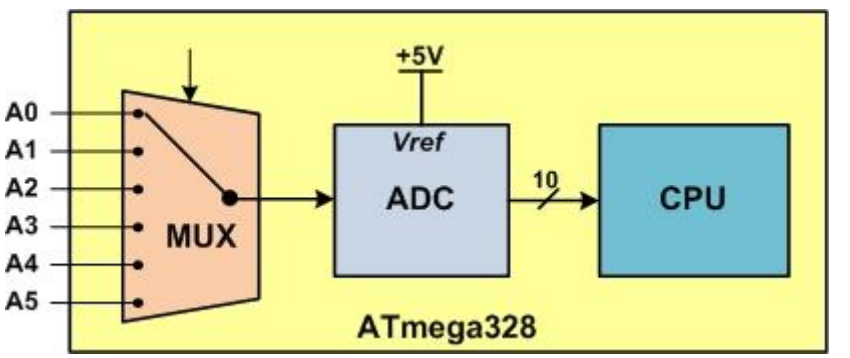
شکل ۱۴-۵: پینهای A0 تا A5

آی سی ATmega328 که بر روی برد آردوینو، به کار رفته است، در دل خود یک ADC دارد که با استفاده از آن می توانیم ولتاژ پایه های A0 تا A5، از برد آردوینو را بخوانیم. در شکل ١٤-٥ پایه های مرتبط با ADC و در شکل ١٤-٦ مدار داخلی آی سی ATmega328 را می بینید. همان طوری که در شکل ١٤-٦ می بینید، ADC به کار رفته در داخل ATmega328، ۱۰ بیتی است و به عنوان Vref در این ADC، ولتاژ ۵ ولت انتخاب شده است. بنابراین اعداد خروجی این ADC، بین ۰ تا ۱۰۲۳ است.

بین پایه های ورودی (A0 تا A5) و پایه ورودی ADC، قسمتی به نام مالتی پلکسر (MUX) وجود دارد که از طریق آن می‌توانیم در هر لحظه یکی از پایه های A0 تا A5 را به ورودی ADC متصل کنیم و مقدار آن را از طریق ADC بخوانیم.

هرگاه بخواهیم ولتاژ پایه ای را اندازه گیری کنیم تابع زیر را صدا می زنیم:

```
analogRead(pinNum)
```



شکل ۱۴-۶: مدار مربوط به ADC داخل ATmega328

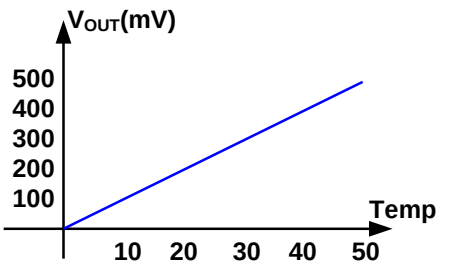
مثلا اگر بخواهیم ولتاژ پایه A2 را بدانیم، دستور analogRead(A2) را استفاده می کنیم. مقدار برگشتی تابع analogRead عددی بین ۰ تا ۱۰۲۳ است. اگر ولتاژ پایه ۰ ولت باشد، عدد صفر برگردانده می شود و اگر ولتاژ پایه ۵ ولت باشد ۱۰۲۳ را باز می گرداند. در حقیقت، بین عدد برگشتی تابع analogRead و ولتاژ پایه ورودی، رابطه ریاضی زیر وجود دارد:

$$number = \frac{1024}{5} V_{IN}$$

بخش ۱۴-۳: اندازه گیری دما با LM35

گاهی اوقات لازم است که دمای محیط اطراف و یا دما محلولی را بدانیم. مثلا اگر ما بخواهیم ربات امدادگری بسازیم که وجود آتش در محیط کمک رسانی خود را تشخیص دهد لازم است که یک سنسور دما داشته باشد که با استفاده از آن دمای اطراف خود را حس کند. از جمله سنسورهای دما LM35 است که دما را به ولتاژ تبدیل می کند. در صفر درجه خروجی این سنسور ۰ ولت است و به ازای هر درجه افزایش دما ولتاژ خروجی آن نیز ۱۰ میلی ولت (یعنی 0.01 ولت) افزایش می یابد. شکل زیر را ببینید. بنابراین رابطه بین دمای محیط، با خروجی آن به صورت زیر است:

$$V = 0.01 \times Temp$$

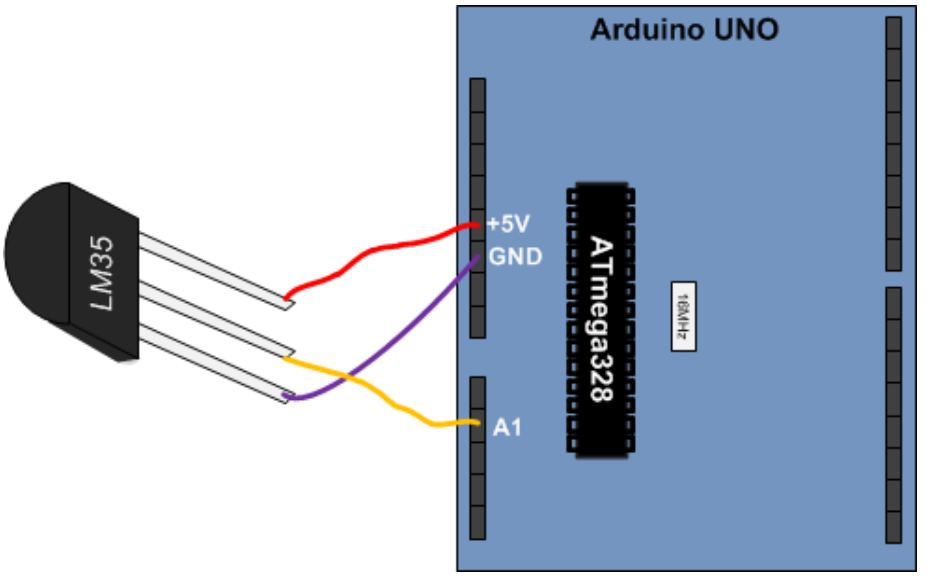


شکل ۱۴-۷: رابطه بین دما و ولتاژ خروجی در LM35

مثلا اگر دمای اطراف این سنسور به ۵۰ درجه سانتیگراد برسد، ولتاژ خروجی آن برابر است با:

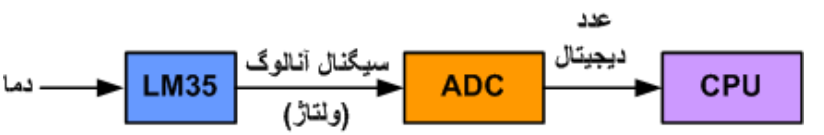
$$0.01 * 50 = 0.5 \text{ V}$$

سنسور LM35 یک عنصر سه پایه است. دو پایه آن به ۵ ولت و زمین متصل می شوند که برق مورد نیاز جهت کار کردن LM35 را برایش فراهم می کند. سنسور LM35 بر روی پایه وسط خود، ولتاژی ایجاد می کند که مقدار ولتاژ آن متناسب با دمای محیط است. همانند شکل زیر، سر وسط سنسور LM35 را به یکی از پایه های A0 تا A5 از برد آردوینو متصل می کنیم و پایه های یک و سه از LM35 را به زمین و ۵ ولت متصل می کنیم.



شکل ۱۴-۸: اتصال سنسور LM35 به برد آردوینو

در عمل، دما مراحل زیر را طی می کند تا مقدار آن به دست CPU برسد.



شکل ۱۴-۹: اتصال خروجی LM35 به ADC

برنامه زیر، دمای محیط را از LM35 می خواند و دمای محیط را از طریق سریال به کامپیوتر ارسال می‌کند. بنابراین به همین سادگی می توانید با استفاده از LM35 یک دماسنج دیجیتال بسازید.

برنامه ۱۳-۱: اندازه گیری دما با استفاده از LM35 و ارسال آن به کامپیوتر

```
void setup() {
    Serial.begin(9600);
    Serial.println("Temperature Sensor Project");
}

void loop() {
    Serial.print("Temp is:");
    long t = analogRead(A1);
    t = t * 500 / 1024;
    Serial.println(t);
    delay(1000);
}
```

ممکن است از خود بپرسید که چرا در برنامه فوق مقدار تابع analogRead را در ۵۰۰ ضرب کرده ایم و بر ۱۰۲٤ تقسیم کرده ایم. در فوق دیدیم که رابطه بین ولتاژی که پایه آردوینو دارد با مقدار برگشتی تابع analogRead به صورت زیر است:

$$number = \frac{1024}{5} V_{IN}$$

همچنین دیدیم که رابطه دما با ولتاژ ایجاد شده توسط LM35 به صورت زیر است:

$$V = 0.01 \times Temp$$

بنابراین رابطه بین عددی که تابع analogRead می دهد با دما به صورت زیر است:

$$number = \frac{1024}{5} V_{IN} = \frac{1024}{5} \times 0.01 \times Temp = \frac{1024}{500} \times Temp$$

لذا اگر بخواهیم دمای محیط را به دست آوریم خواهیم داشت:

$$Temp = \frac{500}{1024} \times number$$

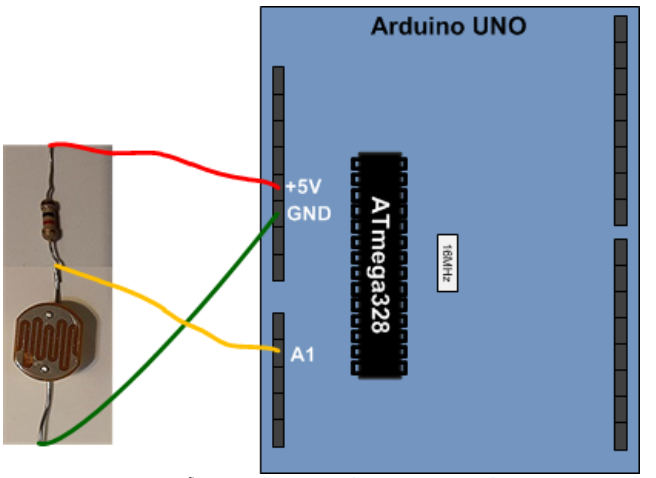
به همین خاطر است که در برنامه مقدار بازگشتی تابع analogRead را در ۵۰۰ ضرب و بر ۱۰۲٤ تقسیم کرده ایم.

بخش ۱۴-۴: اتصال سنسور نور به آردوینو

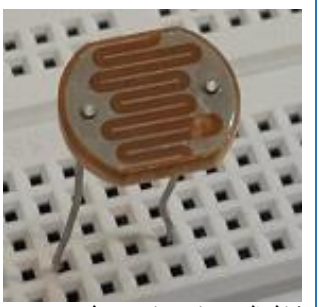
از جمله سنسورهایی که برای اندازه گیری شدت نور وجود دارد، سنسور فتوسل (photo-resistor) است که عکس آن را در شکل ۱٤-۱۰-الف می بینید. این سنسور، دو سر دارد که مقاومت آن بر حسب شدت نوری که به آن می تابد تغییر می کند. زمانی که در محیط تاریک قرار گیرد مقاومت آن زیاد است و هنگامی که در معرض نور قرار گیرد، مقاومت آن کم می‌شود.

برای اینکه بتوانیم مقاومت سنسور فتوسل را اندازه بگیریم، می توانیم مداری به شکل ۱۰-۱٤-ب ببندیم و آن را به برد آردوینو متصل کنیم.

برنامه زیر، ولتاژ پایه A1 را می خواند و به صورت سریال برای کامپیوتر ارسال می کند. هر چه محیط روشن تر باشد، عددی که به کامپیوتر ارسال می شود کوچکتر خواهد بود.



شکل ب: طریقه اتصال یک فتوسل به برد آردوینو



شکل الف. عکس یک فتوسل روی برد برد

شکل ۱۴-۱۰: فتوسل (LDR)

برنامه ۱۴-۲: اندازه گیری شدت نور با فتوسل

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.println(analogRead(A1));
    delay(1000);
}
```

تمرین

۱. با استفاده از LCD و LM35 یک دماسنج دیجیتال درست کنید. برای این منظور LM35 را به یکی از ورودیهای آنالوگ آردوینو متصل کنید و برنامه ای بنویسید که هر ثانیه یکبار مقدار دما را از سنسور بخواند و بر روی LCD نمایش دهد.

۲. می خواهیم هرگاه دمای اتاق از ۲۶ درجه بیشتر بود، به طور اتوماتیک کولر روشن شود. قبلا با استفاده از رله مداری را به پایه ۱۳ از آردوینو متصل کرده ایم که هرگاه پایه ۱۳ را روشن کنیم، کولر روشن می شود. سنسور LM35 را به یکی از پایه های ورودی آنالوگ متصل کنید و برنامه ای بنویسید که هرگاه دمای محیط گرمتر از ۲۶ درجه بود پایه ۱۳ روشن شود و در غیر این صورت پایه ۱۳ خاموش شود.

فصل ۱۵: استفاده کردن از Keypad

بخش ۱۵-۱: آشنایی با دستور do while

در فصل شش با دستور while آشنا شدید. در این بخش با دستور do while آشنا می شوید. ساختار دستور do while به صورت زیر است:

```
do{
        //مجموعه دستورات داخل حلقه
}while(شرط);
```

در دستور do while مجموعه دستورات داخل حلقه اجرا می شوند و زمانی که به انتهای حلقه رسید، شرط جلوی while را چک می‌کند. به طور مثال، در کد زیر، ابتدا محتوای متغیر a توسط سریال ارسال می‌شود و سپس یکی از مقدار a کم می‌شود و در انتها چک می‌شود که آیا مقدار a بزرگ تر از صفر هست یا نه.

```
do{
    Serial.print(a);
    a = a - 1;
}while(a > 0);
```

تفاوت بین while و do while

در دستور while قبل از اینکه وارد حلقه شود، شرط جلوی while را چک می‌کند و اگر شرط درست نبود وارد حلقه نمی‌شود. اما در دستور do while ابتدا دستورات داخل حلقه را اجرا می‌کند و هنگامی که به انتهای حلقه رسید، شرط جلوی while را چک می‌کند. به عبارت دیگر در دستور do while دستورات داخل حلقه دست کم یکبار اجرا می شوند. اما در دستور while اگر شرط برقرار نباشد دستورات داخل حلقه اجرا نمی‌شوند. در زیر فلوچارت مربوط به while و do while در کنار هم رسم شده‌اند تا بتوانید آنها را با هم مقایسه کنید.

مثال ۱۵-۱: مثالی برای do while

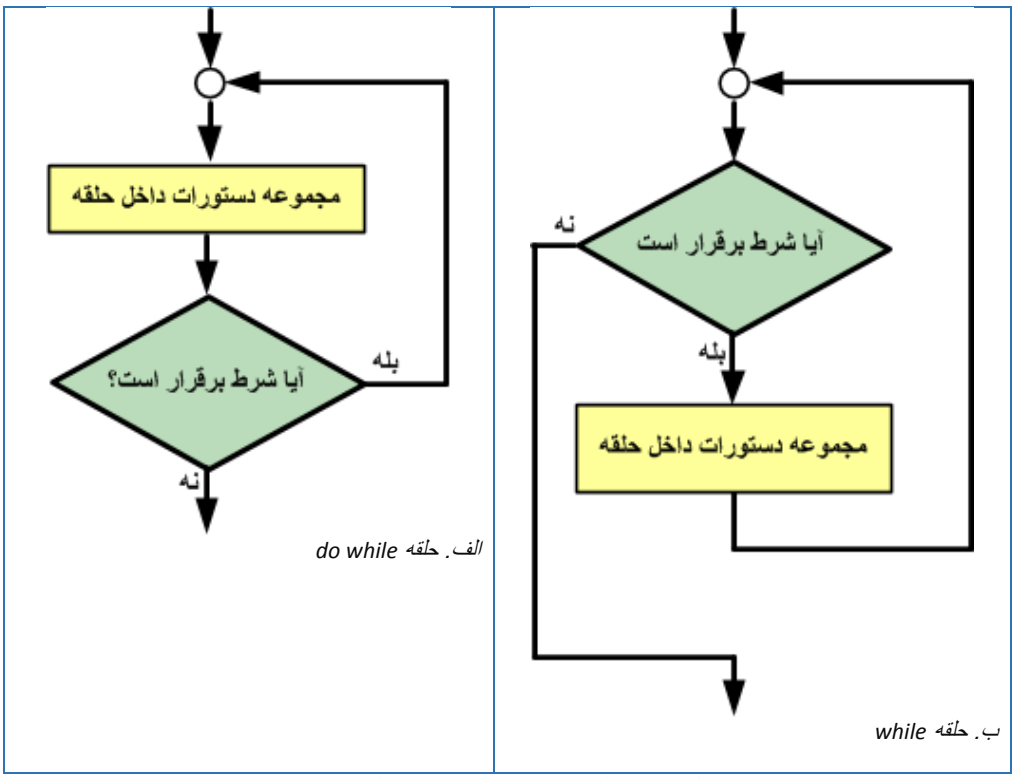
تابعی بنویسید که از طریق سریال هر کاراکتری که دریافت می‌کند را از طریق سریال پس بفرستد. تا مادامی که کاراکتر دریافتی q نبود، به این کار ادامه دهد.

پاسخ:

```
void sendBackFunc()
{
    char c;
    do{
```

```
        c = Serial.read();
        Serial.print(c);
    }while(c != 'q');
}
```

توضیح: در تابع فوق ابتدا می بایست دریافت کردن کاراکتر انجام پذیرد و سپس شرط q نبودن کاراکتر دریافتی چک شود. به همین خاطر از دستور do while استفاده کرده ایم.



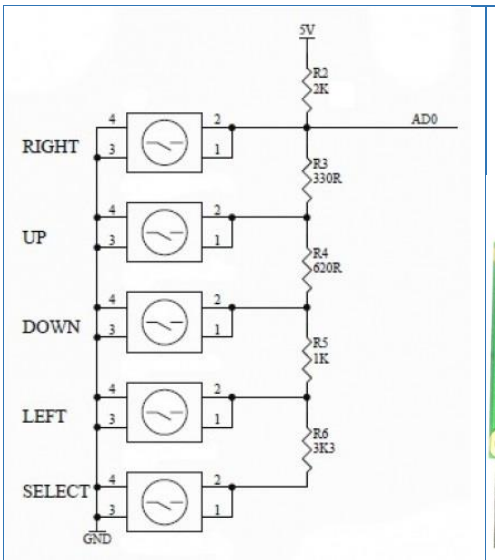
شکل ۱۵-۱: حلقه do while در مقایسه با حلقه while

بخش ۱۵-۲: خواندن کلیدهای کی بورد

در برخی بردهای ال سی دی و کی بورد که برای آردوینو آماده شده است، برای اینکه چندین کلید را با استفاده از یک پایه آردوینو بخوانند، مداری بسته اند که اگر جمیع کلیدها رها باشند ولتاژ پایه آردوینو ۵ ولت است. هنگامی که هر یک از کلیدها را فشار می دهیم، ولتاژ پایه آردوینو کاهش می یابد. میزان ولتاژ پایه آردوینو بیانگر کلیدی است که فشرده شده است. نمونه ای از این بردهای ال سی دی و کی بورد (LCD Keypad Shield) را در عکس ۱۵-۲-الف می بینید. نقشه مدار مربوط به کلیدهای این برد در شکل ۱۵-۲-ب نشان داده شده است و جدول ۱۵-۱ بیانگر ولتاژ پایه آردوینو زمانی که هر یک از کلیدها فشرده‌اند می‌باشد.

وضعیت کلیدها	ولتاژ ایجاد شده	عدد تقریبی خروجی ADC
آزاد بودن همه کلیدها	۵ ولت	۱۰۲۳
فشرده بودن کلید right	۰ ولت	۰
فشرده بودن کلید up	۰,۷ ولت	۱۴۳
فشرده بودن کلید down	۱,۶ ولت	۳۲۷
فشرده بودن کلید left	۲,۴۷ ولت	۵۰۵
فشرده بودن کلید select	۳,۶ ولت	۷۳۶

جدول ۱۵-۱: کلیدها برای شکل ۱۵-۲



شکل ب. شماتیک مدار کلیدها

شکل الف. عکسی از برد Lcd Keypad Shield

شکل ۱۵-۲: برد LCD Keypad Shield

در برنامه ۱۵-۱، هنگامی که هر یک از کلیدهای بالا، پایین، چپ، راست و select فشار داده شوند، نام کلید فشار داده شده بر روی LCD نمایش داده می شود. مثلا اگر کلید راست فشار داده شود، بر روی LCD عبارت RIGHT نمایش داده می شود.

برنامه ۱۵-۱: تشخیص کلید فشار داده شده

```
 #include <LiquidCrystal.h>

LiquidCrystal lcd(8,9,4,5,6,7);
void setup() {
    lcd.begin(16, 2);
}

void loop() {
    lcd.setCursor(0,0);
```

```
    int a = analogRead(A0);
    if(a < 70)
    {
        lcd.print("RIGHT ");
    }
    else
    {
        if(a < 235)
        {
            lcd.print("UP    ");
        }
        else
        {
            if(a < 415)
            {
                lcd.print("DOWN  ");
            }
            else
            {
                if(a < 620)
                {
                    lcd.print("LEFT  ");
                }
                else
                {
                    if(a < 880)
                    {
                           lcd.print("SELECT");
                    }
                }
            }
        }
    }
}
```

در برنامه ۱۵-۲، بر روی LCD عددی نمایش داده می شود و هنگامی که دکمه های بالا و پایین را می‌زنیم مقدار این عدد زیاد و کم می شود و هنگامی که دکمه Select را فشار دهیم، مقداری که بر روی LCD نمایش داده شده است، از طریق سریال با سرعت ۹۶۰۰ به کامپیوتر ارسال می شود که اگر پنجره Serial Monitor باز باشد عدد ارسال شده نمایش داده می شود. در این برنامه، تابع getAKey() صبر می‌کند تا کلید جدیدی فشار داده شود. سپس کد کلید فشار داده شده را باز می‌گرداند.

برنامه ۱۵-۲: دریافت عدد از کاربر با استفاده از LCD و Keypad

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8,9,4,5,6,7);
```

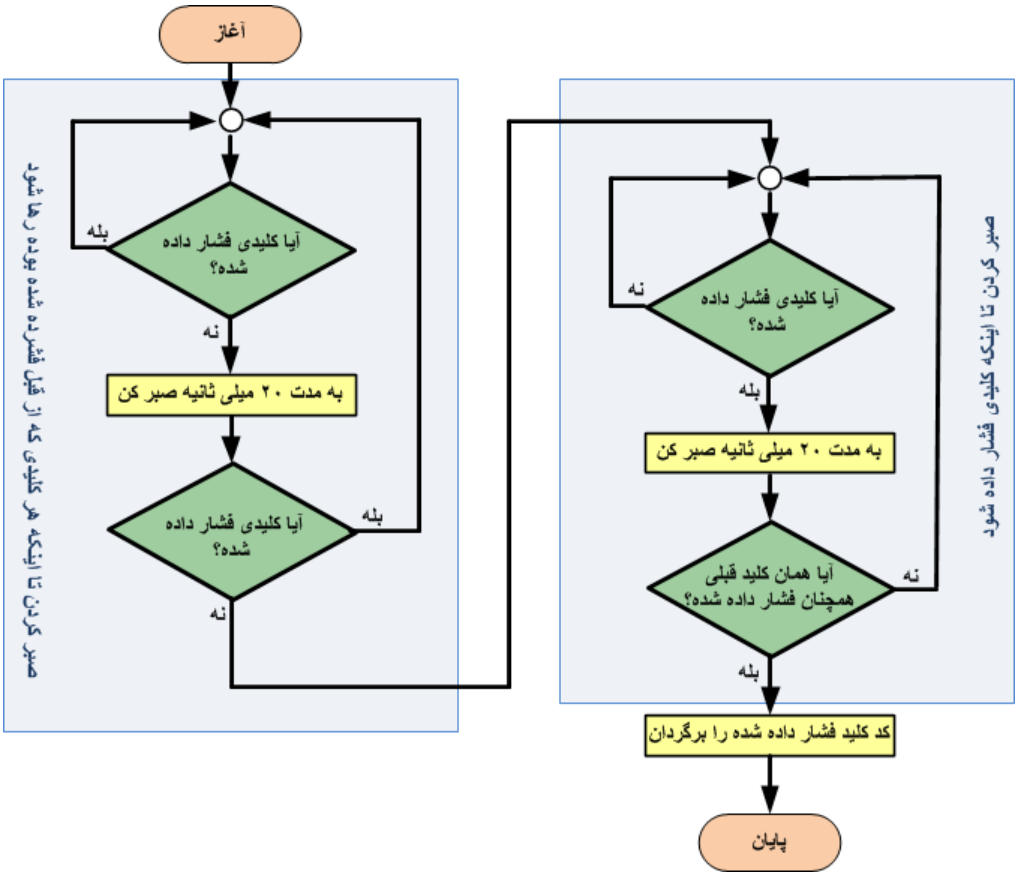
```
void setup() {
    lcd.begin(16, 2);   //init LCD
    Serial.begin(9600); //init Serial
}

//============================================================
// This function returns the state of keys.
// 1: right is press, 2: up is pressed, 3: down is pressed
// 4: left is pressed, 5: select is pressed, 0: no key is pressed
//============================================================
char getKeyState()
{
    int a = analogRead(A0);
    if(a < 70)
    {
        return 1; //right
    }
    if(a < 235)
    {
        return 2; //up
    }
    if(a < 415)
    {
        return 3; //down
    }
    if(a < 620)
    {
        return 4; //left
    }
    if(a < 880)
    {
        return 5; //select
    }
    return 0; //none
}

char getAKey()
{
    // wait until the keys are released
    do{
        while(getKeyState() != 0) //while a key is pressed
        {}
        delay(20); //wait 20 ms
    }while(getKeyState() != 0); //while a key is pressed

    char key1, key2;
    // wait until a key is pressed
    do{
        while(getKeyState() == 0) //while no key is pressed
        {}
        key1 = getKeyState();  //key1 = the key state
        delay(20);  //wait 20 ms
```

```
        key2 = getKeyState(); //newKey = the key state
    }while((key2 == 0)||(key2 != key1)); //repeat if no key is pressed
//or the key is changed

    return key2; //return the pressed key
}

int value = 0;
void loop() {
    lcd.setCursor(1,1); //move cursor to 1,1
    lcd.print(value);   //show value
    lcd.print("    ");

    char key = getAKey(); //get a new key

    if(key == 2) //if the pressed key is up
    {
        value = value + 1; //increase value
    }
    else
    if(key == 3) //if down is pressed
    {
        value = value - 1; //decrease value
    }
    else
    if(key == 5) //if select is pressed
    {
        Serial.print("Value is ");
        Serial.println(value);  //send value to the PC
    }
}
```

در شکل ۱۵-۳، فلوچارت مربوط به تابع getAKey() را می بینید. همان طوری که می بینید، تابع getAKey از دو قسمت تشکیل شده است. ابتدا صبر می کند تا اگر کلیدی از قبل فشرده شده است، رها شود. سپس صبر می کند تا کلیدی فشرده شود و کد کلید فشرده شده را باز می گرداند.

مفهوم bounce و debounce کردن

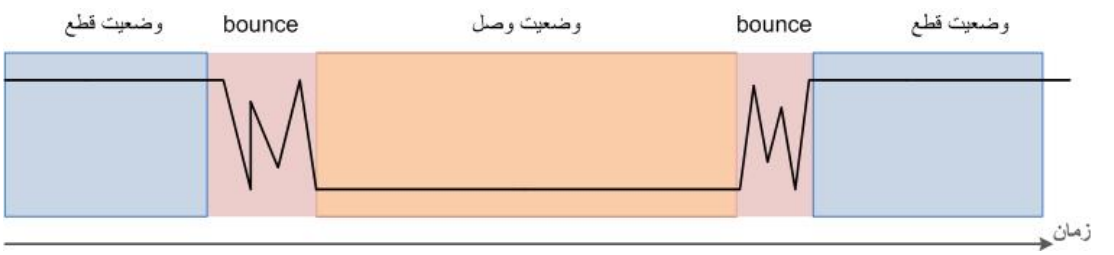
همان طوری که در الگوریتم فوق می بینید، در قسمت اول، ما صبر می کنیم که کلید رها شود، بعد ۲۰ میلی ثانیه صبر می‌کنیم و دوباره چک می کنیم که کلید رها شده باشد. همچنین در قسمت دوم، ما صبر می کنیم که کلیدی فشار داده شود. بعد ۲۰ میلی ثانیه صبر می کنیم و دوباره چک می کنیم که کلید فشرده شده باشد. اما علت این کار چیست؟



شکل ۱۵-۳: الگوریتم تابع getAKey() به صورت فلوچارت

زمانی که کلیدی را به پایین فشار می دهیم، قبل از اینکه کلید کاملا متصل شود، در مدت بسیار کوچکی، یک تماس ضعیفی برقرار می شود. در موقعی که این اتصال ضعیف برقرار می شود اگر وضعیت کلید را بخوانیم، وضعیت کلید دائما بین حالت قطع و وصل در نوسان است. انگار که کسی با سرعت بسیار زیاد، دارد کلید را وصل و قطع می کند. اصطلاحا به این قطع و وصل شدن سریع، bounce می گویند. در شکل۱۵-٤ مفهوم bounce نمایش داده شده است. فرق قطع و وصلی که در اثر bounce به وجود می‌آید با قطع و وصلی که انسان انجام می دهد، در این است که حتی سریع ترین انسانها نیز قادر نیستند که کلیدی را بیشتر از ۱۰ بار در ثانیه قطع و وصل کنند. در حالی که هنگامی که در اثر bounce کلیدی قطع و وصل می‌شود، در هر میلی ثانیه چندین بار قطع و وصل می‌شود. بنابراین اگر کلیدی قطع شود و به مدت دست کم ۲۰ میلی ثانیه نیز قطع بماند مطمئن هستیم که واقعا انسانی کلید را رها کرده است. اما اگر بعد از ۲۰ میلی ثانیه دیدیم که مجددا کلید وصل است، می‌دانیم که اثر bounce بوده و انسان کلید را رها

نکرده. بنابراین مجددا صبر می کنیم تا کلید قطع شود. هرگاه به مدت ۲۰ میلی ثانیه پشت سر هم کلید رها بود یا فشرده بود، مطمئن هستیم که واقعا کلید توسط انسان فشرده یا رها شده است.



شکل ۱۵ - ۴: نوسانات حاصل از bounce در زمان فشرده شدن و نیز رها شدن کلید

بخش ۱۵-۳: آشنایی با دستور switch case

گاهی لازم است مقدار متغیری را با اعداد مختلفی مقایسه کنیم و بر حسب اینکه برابر با چه عددی هست کارهای مختلفی انجام می‌دهیم. در چنین مواردی می‌توانیم از دستور switch case استفاده کنیم. ساختار کلی دستور switch case به صورت زیر است:

```
switch(متغیر)
{
        case value1: فلان کارها را انجام بده; break;
        case value2: فلان کارها را انجام بده; break;
        ...
}
```

در جلوی switch متغیری که می‌خواهیم مقدار آن چک شود را می‌نویسیم و در جلوی هر دستور case یکی از مقادیری که می‌خواهیم با آن مقایسه شود را ذکر می‌کنیم و بعد از case مجموعه کارهایی که می‌خواهیم انجام شود را ذکر می‌کنیم و در انتها کلمه break را می نویسیم. به طور نمونه مثال زیر را ببینید.

مثال ۱۵-۲

برنامه ای بنویسید که کاراکتری را از طریق سریال دریافت کند. اگر کاراکتر دریافتی h بود، کلمه hello را از طریق سریال پس بفرستد. اگر b بود، کلمه bye را پس بفرستد. اگر n بود، کلمه nice را پس بفرستد.

پاسخ:

```
void setup() {
    Serial.begin(9600);
}
```

```
void loop() {
    char c = Serial.read();
    switch(c)
    {
        case 'h': Serial.println("Hello"); break;
        case 'b': Serial.println("Bye"); break;
        case 'n': Serial.println("Nice"); break;
    }
}
```

هنگامی که مقدار متغیر ذکر شده در جلوی switch با مقدار یک case برابر باشد، دستورات بعد از case اجرا می شوند و آن قدر دستورات را اجرا می‌کند تا به کلمه break برسد و اگر پس از یک case، کلمه break ذکر نشده باشد، دستورات بعد از case بعدی نیز اجرا می‌شود. از این خصوصیت می‌توانیم در مواقعی که بخواهیم به ازای چندین مقدار مختلف، کار یکسانی انجام شود، استفاده کنیم. به طور نمونه، مجموعه دستورات زیر مقدار متغیر a را مورد بررسی قرار می‌دهد؛ اگر مقدار a برابر با ۲ یا ٦ باشد، مقدار ۱ را در متغیر b می‌ریزد. اگر مقدار a برابر با ۱ یا ۳ باشد، مقدار ۲ را درون b می‌ریزد و اگر مقدار a برابر با ۵ بود، مقدار ۳ را درون b می‌ریزد.

```
    switch(a)
    {
        case 2:
        case 6: b = 1; break;
        case 3:
        case 1: b = 2; break;
        case 5: b = 3; break;
    }
```

همچنین هنگام نوشتن switch case ممکن است تمایل داشته باشیم که اگر مقدار متغیر جلوی switch برابر با هیچ یک از caseهای ذکر شده نبود، کار مشخصی انجام شود. در چنین مواردی از کلمه کلیدی default استفاده می‌کنیم. مثلا در کد فوق اگر مایل باشیم که در صورتی که مقدار متغیر a هیچ یک از مقادیر ۲، ٦، ۱، ۳ و ۵ نبود، مقدار ٤ را درون b بریزد، می‌توانیم برنامه زیر را بنویسیم.

```
    switch(a)
    {
        case 2:
        case 6: b = 1; break;
        case 3:
        case 1: b = 2; break;
        case 5: b = 3; break;
        default: b = 4; break;
    }
```

در زیر نمونه کدهای بیشتری برای switch case را می‌بینید.

مثال ۱۵-۳

برنامه نوشته شده در مثال ۱۵-۲ را به گونه ای بازنویسی کنید که چه حروف کوچک تایپ شوند و چه حروف بزرگ تایپ شوند، کارهای ذکر شده در مثال ۱۵-۲ را انجام دهد.

پاسخ:

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    char c = Serial.read();
    switch(c)
    {
        case 'H':
        case 'h': Serial.println("Hello"); break;
        case 'B':
        case 'b': Serial.println("Bye"); break;
        case 'N':
        case 'n': Serial.println("Nice"); break;
    }
}
```

مثال ۱۵-۴

در برنامه ۱۵-۲، در داخل حلقه loop برای اینکه مقدار key را با مقادیر مختلف مقایسه کنیم از دستور if استفاده کرده ایم. این مقایسه را با استفاده از دستور switch case انجام دهید.

پاسخ:

```
    switch(key)
    {
        case 2: value = value + 1; break; //increase value
        case 3: value = value - 1; break; //decrease value
        case 5: //if select is pressed
                Serial.print("Value is ");
                Serial.println(value);  //send value to the PC
                break;
    }
```

تمرین

۱. برنامه ای بنویسید که هر ۱۰ میلی ثانیه یکبار وضعیت کلیدها را چک کند. سپس از طریق سریال نام کلید فشرده شده را به کامپیوتر ارسال کند. مثلا اگر کلید بالا فشرده شده بود، عبارت Up را به کامپیوتر ارسال کند. همچنین اگر هیچ کلیدی فشرده نبود، عبارت None را به کامپیوتر بفرستد.

۲. می‌خواهیم با استفاده از LCD و کی بورد یک منو بسازیم. گزینه های منو عبارتند از: Settings، Demo، Timer و Exit. با استفاده از دکمه های بالا و پایین کاربر می‌تواند بین گزینه های منو جابجا شود. (راهنمایی: متغیری از جنس int به نام currentChoice تعریف کنید و با مقدار ۰ مقداردهی اولیه کنید. هنگامی که دکمه پایین زده می شود، مقدار این متغیر را یکی زیاد کنید و هنگامی که دکمه بالا زده می‌شود، مقدار این متغیر را کم کنید. اگر مقدار این متغیر ۰ بود بر روی LCD عبارت Settings را نمایش دهید، اگر ۱ بود، عبارت Demo را نمایش دهید و اگر ۲ بود، Timer را بر روی LCD نمایش دهید و اگر ۳ بود عبارت Exit را نمایش دهید.)

۳. گاهی اوقات در دستگاهی که تولید می شود لازم است تنظیماتی در داخل کارخانه انجام شود. اما هنگامی که محصول به بازار عرضه می شود کاربر عادی از وجود این تنظیمات اطلاع نداشته باشد چرا که تغییر دادن این تنظیمات موجب اختلال در کار دستگاه می شود. در این موارد در برنامه دستگاه چک می شود که اگر کلیدهای خاصی فشرده شدند وارد تنظیمات خاص شود. فرض کنید می خواهیم در صفحه دستگاه هرگاه ۳ مرتبه کلید بالا فشرده شد و سپس کلید پایین فشرده شد وارد صفحه خاصی شود. شما ماژول ال سی دی را به آردوینو متصل کنید و برنامه‌ای بنویسید که هرگاه ۳ مرتبه کلید بالا فشرده شد و سپس کلید پایین فشرده شد، عبارت Factory menu بر روی LCD نمایش داده شود و هر گاه کلید چپ فشار داده شد، نوشته از روی LCD پاک شود. (راهنمایی: یک متغیر تعریف کنید و هرگاه کلید بالا فشرده شد، مقدار آن را یکی زیاد کنید. اگر کلیدی به غیر از بالا و پایین فشرده شد، مقدار این متغیر صفر شود و اگر کلید پایین فشرده شد، مقدار این متغیر چک شود که آیا ۳ است (کلید بالا ۳ مرتبه فشار داده شده) یا نه. اگر برابر با ۳ بود پیغام مربوطه نمایش داده شود.)

فصل ۱۶: آشنایی با آرایه ها، رشته ها و struct

بخش ۱۶-۱: آشنایی با آرایه ها

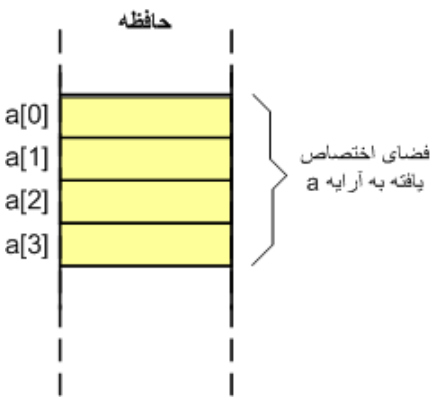
در فصل پنج با تعریف کردن متغیرها آشنا شدید. در این فصل با آرایه ها آشنا می‌شوید. آرایه ها چندین متغیر همسان را در خود جای می‌دهند. طریقه تعریف کردن آرایه ها به صورت زیر است:

```
جنس خانه های آرایه  نام متغیر [تعداد خانه های آرایه];
```

مثلا دستور زیر آرایه ای از جنس int تعریف می‌کند که ٦ خانه دارد:

```
int myArray[6];
```

همان طوری که می‌بینید، طریقه تعریف کردن آرایه مشابه متغیر است. با این تفاوت که جلوی اسمی که به آرایه اختصاص می‌دهیم، تعداد خانه‌هایی که مایلیم آرایه داشته باشد را ذکر می‌کنیم. هنگامی که آرایه‌ای را تعریف می‌کنیم، درون حافظه خانه‌هایی همسان از جنس اعلام شده و به تعداد خانه های گفته شده، برای آرایه اختصاص می‌یابد. به طور مثال، هنگامی که دستور int a[4]; را می‌نویسیم، درون حافظه ٤ خانه پشت سر هم برای آرایه a در نظر گرفته می‌شود که اندازه هر خانه به اندازه متغیر int (یعنی ۲ بایت) است. شکل ١٦-١ را ببینید.



شکل ۱۶-۱: فضای اختصاص یافته توسط دستور int a[4];

خانه‌های آرایه با عدد شماره گذاری می‌شوند و اولین خانه آرایه ۰ نامیده می‌شود، دومین خانه ۱ نامیده می‌شود و به همین ترتیب nمین خانه از آرایه n-1 نامیده می‌شود. برای اینکه به هر یک از خانه های آرایه دسترسی داشته باشیم، نام آرایه را می‌نویسیم و جلوی آن در داخل کروشه ([]) شماره خانه را ذکر می‌کنیم. مثلا اگر بخواهیم در اولین خانه از آرایه a، مقدار ٥ را بریزیم، عبارت زیر را می‌نویسیم:

```
a[0] = 5;
```

همچنین دستور زیر مقدار اولین خانه از متغیر a را با ۹ جمع می‌کند و در سومین خانه از متغیر a می‌ریزد:

```
a[2] = a[0] + 9;
```

به منظور ممارست بیشتر با آرایه مثالهای زیر را ببینید.

مثال ۱۶-۱

برنامه‌ای بنویسید که آرایه‌ای ۱۰ خانه‌ای تعریف کند و اعداد ۱ تا ۱۰ را درون خانه‌های آرایه بریزد.

پاسخ:

```
int numbers[10];
int i;
for(i = 0; i < 10; i = i + 1)
{
    numbers[i] = i + 1;
}
```

توضیح: با توجه به اینکه می‌خواهیم داخل اولین خانه (خانه شماره ۰) عدد ۱ را بریزیم، پس در برنامه فوق عددی که می‌خواهیم داخل آرایه بریزیم یکی بیشتر از شماره خانه آرایه است و به همین خاطر است که مقدار i+1 درون خانه i از آرایه ریخته شده.

مثال ۱۶-۲

برنامه ای بنویسید که آرایه‌ای به طول ۶ خانه تعریف کند (به تعداد خانه‌های آرایه اصطلاحا طول آرایه می‌گویند.) و سپس آن را با اعداد فرد بین ۷ تا ۱۷ پر کند.

پاسخ:

```
int oddNums[6];
int n = 7;
for(i = 0; i < 6; i=i+1)
{
    oddNums[i] = n;
    n = n + 2;
}
```

توضیح: در برنامه فوق n با ۷ مقداردهی اولیه شده است و هر دفعه ۲ تا به مقدار آن افزوده شده است تا اعداد فرد بین ۷ تا ۱۷ را ایجاد کند.

مثال ۱۶-۳

فرض کنید که ۶ تا سنسور دمای LM35 به پایه های A0 تا A5 از برد آردوینو متصل شده اند. برنامه ای بنویسید که با استفاده از دستور analogRead مقادیر پایه های آنالوگ را بخواند و درون آرایه‌ای بریزد. سپس مقادیر ریخته شده در آرایه را از طریق سریال به کامپیوتر بفرستد.

پاسخ:

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    int adcValues[6];
    int i;
    for(i = 0; i < 6; i = i + 1)
    {
        adcValues[i] = analogRead(A0+i);
    }

    //sending values to the PC
    for(i = 0; i < 6; i = i + 1)
    {
        Serial.print(adcValues[i]);
        Serial.print(" ");
    }
    Serial.println("");

    delay(10);
}
```

مقداردهی اولیه آرایه ها

هنگامی که آرایه‌ای را تعریف می‌کنیم، می‌توانیم با استفاده از = مقادیری را درون خانه‌های آن بریزیم. به طور نمونه دستور زیر آرایه‌ای به نام a از جنس int تعریف می‌کند و به ترتیب اعداد ۲، ۳، ۵، ۷و ۱۱ را درون خانه های حافظه می‌ریزد:

```
int a[5]={2,3,5,7,11};
```

همچنین دستور زیر آرایه ای از جنس float تعریف می کند و اعداد 0 و 3.2 و 4.6 را درون خانه‌های آن می‌ریزد.

```
float f[3] = {0.0,3.2,4.6};
```

هنگامی که آرایه ای را در حین تعریف کردن مقداردهی اولیه می‌کنیم، اگر تعداد اعدادی که می‌نویسیم کمتر از تعداد خانه های آرایه باشد، به ترتیب مقادیر ذکر شده را در خانه های آرایه می‌ریزد و مابقی خانه‌ها را با صفر پر می‌کند. مثلا دستور زیر آرایه‌ای به طول ۵ خانه تعریف می‌کند و مقدار ۲ را در خانه صفر از آرایه می‌ریزد و مابقی خانه ها را با ۰ مقداردهی می‌کند.

```
int a[5]={2};
```

توجه داشته باشید که اگر آرایه ها را مقداردهی اولیه نکنیم، درون خانه های آن، هر مقداری می‌تواند باشد. بنابراین اگر می‌خواهید حتما خانه های آرایه ای صفر باشد، می‌بایست مقدار ۰ را درون خانه های آن بریزد. دستور زیر آرایه ای به طول ۹ تعریف می‌کند و جمیع خانه های آن را با مقدار ۰ مقداردهی اولیه می‌کند.

```
int anArray[9]={0};
```

توجه دارید که در دستور فوق، تعداد مقادیری که برای مقداردهی اولیه ذکر کرده‌ایم از تعداد خانه ها کمتر است و بنابراین مابقی خانه ها را با عدد ۰ پر می‌کند.

ارسال کردن آرایه ای به درون تابع

برای اینکه در تعریف تابعی اعلام کنیم که قرار است آرایه ای را به عنوان آرگومان دریافت کنیم، ابتدا جنس آرایه و سپس اسم آرایه و علامت کروشه باز و بسته را می نویسیم. همچنین در هنگام صدا زدن تابع، اسم آرایه را ذکر می‌کنیم. به طور نمونه در برنامه زیر تابعی به نام sumOfArray نوشته شده است که آرایه ای از جنس int به عنوان آرگومان دریافت می‌کند و حاصل جمع مقادیر داخل آرایه را به دست می‌آورد و به عنوان مقدار برگشتی بر می‌گرداند. در تابع sumOfArray، متغیر arrayLen، تعداد خانه های آرایه را بیان می‌کند. درون تابع loop صدا زده ایم تا حاصل جمع مقدارهای داخل آرایه ای را به دست آوریم.

برنامه ۱۶-۱

```
void setup()
{
}
void loop()
{
    int sum;
    int myArray[6] = {25,4,14,9,5,19};
    sum = sumOfArray(myArray,6);
}
```

```
int sumOfArray(int ar[], int arrayLen)
{
    int sum = 0;
    for(i = 0; i < arrayLen; i=i+1)
    {
        sum = sum + ar[i];
    }
    return i;
}
```

مثال ۱۶-۴

تابعی بنویسید که آرایه ای از int را به عنوان آرگومان دریافت کند و بزرگ ترین عدد موجود در آرایه را به عنوان مقدار برگشتی بازگرداند.

پاسخ:

```
int maximum(int ar[], int arrayLen)
{
    int m = ar[0];
    for(i = 1; i < arrayLen; i=i+1)
    {
        if(ar[i] > m)
        {
            m = ar[i];
        }
    }
    return m;
}
```

توضیح: در برنامه فوق، متغیر m را با مقدار اولین خانه از آرایه مقداردهی می‌کنیم. سپس مقدار m را با یکایک خانه‌های آرایه مقایسه می‌کند و اگر مقدار هر یک از خانه ها بیشتر از m بود، m را با محتوای آن خانه مقداردهی می‌کند. در انتها m حاوی بزرگ ترین عدد موجود در آرایه است که به عنوان مقدار برگشتی آن را بر می‌گردانیم.

بخش ۱۶-۲: آشنایی با رشته ها (آرایه ای از کاراکترها)

در فصل ۵ دیدیم که متغیرهای char می‌توانند یک حرف را درون خود نگهداری کنند. پس اگر ما بخواهیم یک کلمه و یا یک جمله را نگهداری کنیم از چه متغیری می‌بایست استفاده کنیم؟

برای نگهداری کلمات و یا جملات می‌توانیم آرایه ای از جنس char تعریف کنیم که طول آرایه را به اندازه حداکثر تعداد حروفی که می‌خواهیم درون آن جای دهیم، تعریف می‌کنیم. مثلا اگر بخواهیم در

برنامه ای یک نام کاربری را نگهداری کنیم و طول نام کاربری حداکثر ۳۰ حرف باشد، می‌توانیم آرایه ای به صورت زیر تعریف کنیم:

```
char userName[30];
```

چنانچه بخواهیم userName را در لحظه تعریف کردن، مقداردهی اولیه کنیم، می‌توانیم از علامت = استفاده کنیم. مثلا دستور زیر علاوه بر تعریف کردن آرایه ای به نام userName، کلمه Sobhan را نیز درون آن می‌ریزد:

```
char username[30]="Sobhan";
```

ممکن است این سوال برای شما مطرح شود که طول کلمه Sobhan کمتر از ۳۰ حرف است، پس چگونه انتهای نام کاربری مشخص شود؟ در زبان سی برای اینکه انتهای رشته ای را مشخص کنند از کد اسکی 0 استفاده می‌کنند. به طور نمونه مثال زیر را ببینید.

مثال ۵-۱۶

هنگامی که دستور زیر را بنویسیم، درون حافظه چه اتفاقی می‌افتد؟

```
char name[15] = "Koorosh1";
```

پاسخ:

هنگامی که دستور فوق را بنویسیم ۱۵ خانه از حافظه جهت نگهداری از آرایه name اختصاص می‌یابد و سپس حروف کلمه Koorosh1 درون آرایه ریخته می‌شود. با توجه به جدول ۷-۱، کد اسکی حرف 'K'، برابر با ۷۵ است و کد اسکی حروف o و r و s و h به ترتیب ۱۱۱ و ۱۱۴ و ۱۱۵ و ۱۰۴ است. همچنین کد اسکی مربوط به عدد '1' برابر با ۴۹ است. بنابراین محتوای خانه های آرایه به صورت زیر می‌شود:

مثال ۱۶-۵

تابعی بنویسید که آرایه‌ای از کاراکترها را به عنوان آرگومان ورودی دریافت کند. سپس طول نوشته موجود در آرایه را به عنوان مقدار برگشتی بازگرداند.

پاسخ:

همان طوری که در فوق ذکر شد، برای این که انتهای نوشته ای را علامتگذاری کنند، کد اسکی ۰ را قرار می‌دهند. بنابراین برای اینکه طول نوشته موجود در آرایه ای را به دست آوریم، از یک حلقه for استفاده می کنیم و شماره اولین خانه ای از آرایه را که حاوی کد اسکی ۰ است را به عنوان طول نوشته پس می-فرستیم. به طور مثال اگر کد اسکی ۰ را اولین بار در خانه ۲ مشاهده کنیم، خانه های ۰ و ۱ حاوی نوشته هستند و بنابراین طول نوشته برابر با ۲ است که برابر با شماره خانه حاوی کد اسکی ۰ می‌باشد.

```
int getStrLength(char str[])
{
    int i = 0;
    while(str[i] != 0)
    {
        i = i + 1;
    }
    return i;
}
```

توضیح: در زبان سی، تابعی به نام strlen از قبل نوشته شده است که کار آن اندازه گیری طول نوشته موجود در آرایه است. هدف ما از نوشتن این تابع، صرفا این بوده که درک کنیم که چگونه می‌توانیم با آرایه ای از نوشته ها کار کنیم.

بخش ۱۶-۳: آشنایی با توابع آماده مربوط به رشته ها

در زبان سی تعدادی توابع جهت کار کردن بر روی رشته ها از قبل نوشته شده اند که ما می‌توانیم از آنها در برنامه‌های خود استفاده کنیم. در جدول زیر لیست برخی از این توابع به همراه کاربرد آنها ذکر شده است.

تابع	کاربرد
strcpy(char str1[],char str2[])	محتوای str2 را به درون str1 کپی می‌کند.
strlen(char str[])	تعداد حروف موجود در str را برمی‌گرداند. (طول نوشته موجود در str را برمی‌گرداند.)
strcat(char str1,char str2[])	نوشته موجود در str2 را به انتهای str1 اضافه می‌کند.

strcmp(char str1[],char str2[])	محتوای موجود در str1 و str2 را با هم مقایسه می‌کند. اگر آنها دقیقا با هم برابر بودند مقدار ۰ را برمی‌گرداند. در غیر این صورت اگر محتوای str1 از لحاظ ترتیب الفبایی قبل از str2 قرار بگیرد، مقدار 1- را برمی‌گرداند و اگر بعد از str2 قرار بگیرد مقدار ۱ را برمی‌گرداند.
strstr(char str1[],char str2[])	درون str1 به دنبال اولین محلی که در آن str2 وجود داشته باشد، می‌گردد. اگر پیدا کند، شماره خانه اولین محلی که پیدا کرده است را باز می‌گرداند و اگر پیدا نکند، مقدار 1- را برمیگرداند.

جدول ۱۶-۱: برخی توابع آماده مربوط به رشته ها

برای اینکه درک بهتری از عملکرد توابع فوق به دست آورید، در زیر نمونه برنامه هایی با استفاده از توابع فوق نوشته شده‌اند.

مثال ۱۶-۶

آرایه ای از جنس کاراکتر با طول ۲۰ حرف به نام password تعریف کنید. سپس با استفاده از تابع strcpy به درون password عبارت myPassword را کپی کنید و با استفاده از تابع println از طریق سریال محتوای password را به کامپیوتر بفرستید.

پاسخ:

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    char password[20];
    strcpy(password,"myPassword");
    Serial.println(password);
    delay(100);
}
```

مثال ۱۶-۷

دو آرایه به نامهای s1 و s2 تعریف کنید که طول s1، ۲۵ کاراکتر و طول s2، ۱۰ کاراکتر باشد. سپس به درون آرایه s1 کلمه Hello و به درون s2 کلمه World را بریزد. سپس با استفاده از تابع strcat، آرایه s2 را به انتهای آرایه s1 اضافه کنید و محتوای نهایی آرایه s1 را از طریق سریال به کامپیوتر بفرستید.

پاسخ:

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    char s1[25],s2[10];
    strcpy(s1,"Hello");
    strcpy(s2,"World");
    strcat(s1,s2);
    Serial.println(s1);
    delay(100);
}
```

توضیح: با توجه به اینکه s1 حاوی Hello و s2 حاوی World است، هنگامی که تابع strcat صدا زده می شود، کلمه World به انتهای Hello افزوده می شود که عبارت HelloWorld به دست می آید و آن را از طریق سریال به کامپیوتر می فرستد. اگر بخواهیم بین Hello و World فاصله ای قرار بگیرد و این دو کلمه به هم نچسبند، می توانیم ابتدا یک رشته که حاوی یک کاراکتر فاصله است را به انتهای s1 اضافه کنیم و سپس s2 را به انتهای s1 اضافه کنیم. یعنی در برنامه فوق قبل از ;strcat(s1,s2) دستور زیر را اضافه کنیم:

```
strcat(s1," ");
```

بخش ۱۶-۴: آشنایی با struct

گاهی اوقات متغیرهایی که می‌خواهیم تعریف کنیم به هم مرتبط هستند. به طور مثال اگر قرار باشد برنامه‌ای را برای مدیریت کارکنان یک شرکت بنویسیم، نام و شماره پرسنلی و شماره تلفن پارامترهای به هم مرتبطی هستند که با هم مشخصات یک فرد را تشکیل می‌دهد. همچنین اگر بخواهیم زمان را ذخیره کنیم، لازم است که مقادیر ساعت و دقیقه و روز و ماه و سال را ذخیره کنیم. در چنین مواردی می‌توانیم با استفاده از struct انواع متغیر مورد نیاز را کنار هم بگذاریم و نوع جدیدی متغیر بسازیم که توانایی ذخیره کردن پارامترهای مورد نیاز را داشته باشد. به طور مثال، دستور زیر نوع جدیدی متغیر به نام Time تعریف می‌کند که توانایی نگهداری از ساعت و دقیقه و ثانیه را دارد:

```
struct Time{
    int hour;
    int min;
    int sec;
};
```

تعریف کردن متغیر از جنس Time همانند تعریف کردن متغیرهای دیگر است. به طور مثال اگر بخواهیم از جنس Time متغیرهایی به نامهای t1 و t2 را تعریف کنیم، می‌توانیم دستور زیر را بنویسیم:

```
Time t1, t2;
```

توجه داشته باشید که هنگامی که Time را تعریف می‌کنیم، هیچ فضایی از حافظه به Time اختصاص نمی‌یابد. بلکه فقط به کامپایلر می‌گوید که Time دارای چه ساختاری است. اما هنگامی که متغیری را از جنس Time تعریف می‌کنیم و مثلا می‌نویسیم "Time t1;" کامپایلر فضایی را برای ذخیره کردن t1 اختصاص می‌دهد که برای ذخیره کردن جمیع اجزایی که داخل Time تعریف کرده‌ایم جا دارد.

برای دسترسی به اجزای متغیری که از جنس یک struct تعریف کرده‌ایم از علامت نقطه استفاده می‌کنیم. به طور مثال دستور زیر عدد ۱۰ را درون قسمت ساعت از متغیر t1 می‌ریزد:

```
t1.hour = 10;
```

همچنین دستور زیر مقدار دقیقه از متغیر t2 را درون متغیر a می‌ریزد:

```
int a = t2.min;
```

همچنین دستور زیر مقدار ثانیه از متغیر t2 را درون متغیر ثانیه از متغیر t1 می‌ریزد:

```
t1.sec = t2.sec;
```

برای اینکه کل مقادیر متغیر t1 را در متغیر t2 کپی کنیم، می‌توانیم همانند سایر متغیرها از علامت = استفاده کنیم. به طور مثال اگر دستور زیر را بنویسیم مقدار t1 به درون t2 کپی می‌شود:

```
t2 = t1;
```

مثلا اگر t1 حاوی ساعت 9:12:15 باشد، بعد از اجرا شدن دستور فوق t2 نیز حاوی زمان 9:12:15 خواهد شد.

در زیر برنامه‌ای نوشته شده که درون متغیر Time ساعت جاری را نگه می‌دارد و هر ثانیه یکبار از طریق سریال به کامپیوتر می‌فرستد.

برنامه ۱۶-۲

```
struct Time
{
  int hour;
  int minute;
  int second;
};

Time t;

void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  t.second = t.second + 1;

  if(t.second >= 60)
  {
    t.second = 0;
    t.minute = t.minute + 1;
    if(t.minute >= 60)
    {
      t.minute = 0;
      t.hour = t.hour + 1;
      if(t.hour >= 24)
      {
        t.hour = 0;
      }
    }
  }

  Serial.print(t.hour);
  Serial.print(":");
  Serial.print(t.minute);
  Serial.print(":");
  Serial.println(t.second);

  delay(1000);
}
```

در برنامه فوق تمام توانایی برد آردوینو صرف نگهداشتن زمان جاری شده، در عین حال اگر برق آردوینو قطع شود، زمان جاری از دست می‌رود و دوباره از ساعت ۰ شروع به شمارش می‌کند. در فصل ۱۸ با آی سی های RTC آشنا می‌شوید که ساعت و تاریخ جاری را به ما می‌دهند.

ارسال متغیری از جنس struct به درون تابع

طریقه ارسال متغیرهایی که با استفاده از struct تعریف شده باشند درست همانند سایر متغیرهاست. به طور نمونه در زیر تابعی به نام isAfter تعریف شده که اگر زمان t1 بعد از t2 بود، مقدار ۱ را برمی‌گرداند و در غیر این صورت مقدار ۰ را برمی‌گرداند:

```
int isAfter(Time t1, Time t2)
{
    if(t1.hour > t2.hour){
        return 1;
    }
    if(t1.hour < t2.hour){
        return 0;
    }
```

```
    if(t1.minute > t2.minute) {
        return 1;
    }
    if(t1.minute < t2.minute) {
        return 0;
    }
    if(t1.second > t2.second) {
        return 1;
    }
    else {
        return 0;
    }
}
```

تمرین

۱. آرایه ای به طول ۱۰ تعریف کنید. سپس برنامه ای بنویسید که محتوای خانه های ۰ تا ۹ از آرایه را با مضربهای ۳ که بین ۳ تا ۳۰ هستند پر کند.

۲. برنامه ای بنویسید که منتظر شود تا ۸ تا کاراکتر از طریق سریال دریافت کند، سپس هر ۸ تا کاراکتر را یکجا برای کامپیوتر پس بفرستد. (توضیح: برای این منظور آرایه ای به طول ۸ خانه از جنس char تعریف کنید و یک متغیر int به عنوان شمارنده تعریف کنید. هر کاراکتری که دریافت می کنید را به ترتیب در خانه های آرایه قرار دهید و مقدار شمارنده را یکی زیاد کنید. هنگامی که مقدار شمارنده به ۸ رسید محتوای جمیع خانه های آرایه را پس بفرستید.)

۳. می‌خواهیم یک سیستم نظرسنجی اتوماتیک درست کنیم که کاربران می‌توانند از طریق سریال یکی از گزینه های a، b، c یا d را انتخاب کنند و در نهایت هرگاه حرف r تایپ شد، تعداد دفعاتی که به هر یک از گزینه‌ها رای داده شده است از طریق سریال به کامپیوتر ارسال می‌شود. یک آرایه به طول ٤ از جنس int تعریف کنید. سپس برنامه ای بنویسید که هر گاه از طریق سریال حرف a را دریافت کردید، محتوای خانه صفرم از آرایه را یکی زیاد کند، اگر b را دریافت کردید، محتوای خانه یکم را زیاد کند و به همین ترتیب الی آخر. هرگاه حرف r را دریافت کردید، مقادیر جمیع خانه های آرایه را به ترتیب به کامپیوتر ارسال کنید.

فصل ۱۷: دسترسی به EEPROM

بخش ۱۷-۱: حافظه های موجود در آردوینو

همان طوری که در فصل یک ذکر شد، حافظه ها بر دو قسم هستند: فَرّار و غیر فرّار. حافظه های فرار با قطع شدن برق اطلاعات خود را از دست می‌دهند. در حالی که حافظه های غیرفرار با قطع شدن برق اطلاعات خود را از دست نمی‌دهند.

درون برد آردوینو یک آی سی میکروکنترلر وجود دارد که برنامه هایی که برای آردوینو می‌نویسیم توسط این آی سی اجرا می‌شود. در دل میکروکنترلر سه نوع حافظه وجود دارند:

- حافظه رَم (RAM): متغیرهایی که در هنگام برنامه نویسی با آردوینو تعریف می‌کنیم درون حافظه RAM قرار می‌گیرد و با توجه به اینکه حافظه RAM فرار است، هنگامی که برق برد آردوینو قطع شود و یا دکمه ریسِت (Reset) که روی برد آردوینو است را فشار دهیم، مقادیر داخل متغیرها پاک می‌شوند.
- حافظه فِلَش رام (Flash ROM): حافظه Flash ROM از جنس ROM است و غیرفرار می‌باشد. برنامه‌هایی که بر روی برد آردوینو می‌ریزیم، درون حافظه Flash ROM قرار می‌گیرد و بنابراین با قطع شدن برق، برنامه محفوظ می‌ماند و پاک نمی‌شود.
- حافظه ای دو پی رام (EEPROM): حافظه EEPROM از جنس ROM است و بنابراین اطلاعاتی که درون آن بریزیم با قطع شدن برق پاک نمی‌شوند. گاهی اوقات ما لازم داریم که مقادیر متغیرهایی را در محلی ذخیره کنیم تا با قطع شدن برق مقادیر آنها محفوظ بماند. به طور مثال دستگاهی که دمای منزل را کنترل می‌کند را در نظر بگیرید. ما این دستگاه را تنظیم کرده ایم که دمای منزل را روی ۲۵ درجه نگه دارد. هنگامی که برق شهر قطع می‌شود، خوبست که تنظیمات این دستگاه به هم نخورد و هنگامی که برق می‌آید دستگاه دما را بر روی ۲۵ درجه نگه دارد. بنابراین تنظیماتی که انجام می‌دهیم درون EEPROM دستگاه ذخیره می‌شود. ما در این فصل طریقه ذخیره کردن اطلاعات در داخل EEPROM و چگونگی دریافت اطلاعات از آن را فرا می‌گیریم.

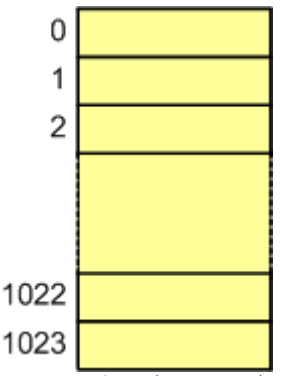
بخش ۱۷-۲: طریقه دسترسی به اطلاعات موجود در EEPROM

حافظه EEPROM که درون آردوینو UNO وجود دارد، دارای ۱۰۲٤ خانه است که هر خانه آن ظرفیت ذخیره کردن یک بایت دیتا را دارد. خانه‌های EEPROM با شماره های ۰ تا ۱۰۲۳ شماره گذاری شده‌اند. در شکل زیر، حافظه EEPROM را می‌بینید. در سایر انواع بردهای آردوینو، تعداد خانه‌های موجود در

EEPROM ممکن است فرق کند. در جدول ۱۷-۱، مشخصات EEPROM موجود در برخی از بردهای آردوینو ذکر شده است.

برد آردوینو	اندازه حافظه EEPROM
آردوینو UNO	۱۰۲۴ بایت
آردوینو مینی	۱۰۲۴ بایت
آردوینو نانو	۱۰۲۴ بایت
آردوینو پیکو	۱۰۲۴ بایت
آردوینو لی لی پد	۱۰۲۴ بایت
آردوینو تاینی لی لی پد	۵۱۲ بایت
آردوینو لئوناردو	۱۰۲۴ بایت
آردوینو Esplora	۱۰۲۴ بایت
آردوینو Mega	۴۰۹۶ بایت

جدول ۱۷- ۱: تعداد خانه های EEPROM در بعضی از بردها



شکل ۱۷-۱: شکل حافظه EEPROM

برای اینکه درون هر یک از خانه‌های EEPROM مقداری را بریزیم از دستور EEPROM.write استفاده می‌کنیم:

```
EEPROM.write(شماره خانه حافظه, مقداری که می‌خواهیم ذخیره شود);
```

به طور مثال، دستور زیر عدد ۲۳ را درون خانه ٤ می‌ریزد:

```
EEPROM.write(4,23);
```

همچنین دستور زیر مقدار متغیر a را درون خانه ٦ می‌ریزد:

```
EEPROM.write(6,a);
```

توابع EEPROM.write و EEPROM.read در داخل فایلی به نام EEPROM.h تعریف شده اند. بنابراین هنگامی که در برنامه خود از این توابع استفاده می‌کنیم می‌بایست فایل EEPROM.h را در بالای برنامه خود include کنیم.

به طور نمونه مثال زیر را ببینید.

مثال ۱۷-۱

برنامه ای بنویسید که اعداد ۰ تا ۲۰ را درون خانه های ۰ تا ۲۰ از حافظه EEPROM ذخیره کند.

پاسخ:

```
#include <EEPROM.h>
void setup()
{
}
```

```
void loop()
{
    byte i;
    for(i = 0; i <= 20; i = i + 1)
    {
        EEPROM.write(i, i);
    }
}
```

برای اینکه مقداری که درون خانه‌های EEPROM است را بخوانیم از تابع EEPROM.read استفاده می‌کنیم:

```
EEPROM.read(شماره خانه);
```

به طور مثال، در زیر متغیری از جنس byte تعریف کرده ایم. سپس با استفاده از EEPROM.read مقدار خانه ۲۰ از EEPROM را خوانده ایم و درون متغیر c ریخته ایم:

```
byte c;
c = EEPROM.read(20);
```

در زیر مثالهای بیشتری را با استفاده از EEPROM می‌بینید.

مثال ۱۷-۲

برنامه ای بنویسید که پایه ۱۳ چشمک بزند و سرعت چشمک زدن آن از طریق سریال قابل کنترل باشد. اگر از طریق سریال عدد ۱ را به آردوینو بفرستیم، پایه ۱۳ هر ۱ ثانیه یکبار چشمک بزند، اگر ۲ بفرستیم هر ۲ ثانیه یکبار چشمک بزند و به همین ترتیب اگر اعداد ۳، ۴ یا ۵ را بفرستیم، سرعت چشمک زدن پایه ۱۳ به هر ۳ ثانیه یکبار و هر ۴ ثانیه یکبار و هر ۵ ثانیه یکبار تغییر کند. همچنین مقدار زمان چشمک زدن در EEPROM ذخیره شود تا در صورت خاموش شدن آردوینو، دفعه بعد که آردوینو روشن می‌شود با همان سرعت قبلی چشمک بزند.

پاسخ:

```
#include <EEPROM.h>
byte t;
void setup()
{
    Serial.begin(9600);
    pinMode(13,OUTPUT);

    t = EEPROM.read(0);     //read location 0
    if((t == 0)||(t > 5)) //if an invalid value
    {
        t = 1;
    }
}
void loop()
```

```
{
    digitalWrite(13,HIGH);
    delay(t*1000);   //wait t seconds
    digitalWrite(13,LOW);
    delay(t*1000);   //wait t seconds

    if(Serial.available())
    {
        char c = Serial.read();
        switch(c)
        {
            case '1': t = 1; break;
            case '2': t = 2; break;
            case '3': t = 3; break;
            case '4': t = 4; break;
            case '5': t = 5; break;
        }
        EEPROM.write(0,t); //store t in location 0
    }
}
```

توضیح درباره برنامه فوق:

۱. هنگامی که دستگاه روشن می‌شود، مقدار قبلی پارامتر t را از داخل EEPROM می‌خواند و درون متغیر t می‌ریزد. اگر از قبل مقدار معتبری در حافظه نباشد (اگر هنوز مقدار t را از طریق سریال تنظیم نکرده باشیم، مقدار داخل خانه EEPROM، مقدار غیر معتبری است.) مقدار t را با ۱ مقداردهی می‌کند تا هر ۱ ثانیه یکبار چشمک بزند.

۲. درون loop، پایه ۱۳ را روشن و خاموش می‌کنیم و اگر از طریق سریال کاراکتری دریافت شده باشد، کاراکتر دریافتی را بررسی می‌کنیم تا اگر کاراکترهای ۱ تا ۵ بودند، سرعت چشمک زدن را تنظیم کنیم و پس از اینکه t را مقداردهی کردیم، مقدار آن را در EEPROM ذخیره می‌کنیم.

همانند مثال فوق، اگر بخواهیم مقدار هر پارامتر دیگری در زمان خاموش شدن برد آردوینو محفوظ بماند، در زمان تنظیم شدن و تغییر کردن، مقدار متغیر را در EEPROM ذخیره می‌کنیم و موقعی که آردوینو روشن می‌شود در تابع setup مقدار متغیر را از درون EEPROM می‌خوانیم.

آشنایی با تابع update

تعداد دفعاتی که می‌توانیم داخل خانه‌های حافظه‌های ROM بنویسیم محدود است. به طور مثال، حافظه EEPROM به کار رفته در آردوینو، به گونه‌ای ساخته شده است که حداکثر صد هزار مرتبه می‌توانیم در هر خانه آن، مقادیری را بنویسیم و بعد از آن، آن خانه از EEPROM خراب می‌شود و قادر به نگهداری

مقادیر نخواهد بود. بنابراین برای اینکه عمر EEPROM بیشتر شود، خوبست که تعداد دفعاتی که درون EEPROM می‌نویسیم را تا حد امکان کمتر کنیم.

به طور مثال در برنامه ۱۷-۲، هر بار که کاراکتری از طریق سریال دریافت می‌شود، مقدار t درون حافظه ذخیره می‌شود. حال آنکه ممکن است کاراکتر دریافتی از کاراکترهای ۱ تا ۵ نبوده باشد یا اصلا کاراکتر دریافتی همان مقدار قبلی بوده باشد. بنابراین در آردوینو علاوه بر تابع write تابع دیگری به نام update وجود دارد که چک می‌کند که اگر مقداری که می‌خواهیم داخل EEPROM ذخیره کنیم با مقدار قبلی آن خانه متفاوت بود، مقدار جدید را درون EEPROM می‌ریزد. طریقه استفاده از دستور update همانند write است و به صورت زیر می‌باشد:

```
EEPROM.update(شماره خانه حافظه, مقداری که می‌خواهیم ذخیره شود);
```

در مثال زیر، برنامه مثال قبل را با استفاده از تابع update بازنویسی کرده‌ایم.

مثال ۱۷-۳

برنامه مثال ۱۷-۲ را با استفاده از تابع update باز نویسی کنید.

پاسخ:

```
#include <EEPROM.h>
void setup()
{
    Serial.begin(9600);
    pinMode(13,OUTPUT);

    t = EEPROM.read(0);     //read location 0
    if((t == 0)||(t > 5)) //if an invalid value
    {
        t = 1;
    }
}
void loop()
{
    digitalWrite(13,HIGH);
    delay(t*1000);   //wait t seconds
    digitalWrite(13,LOW);
    delay(t*1000);   //wait t seconds

    if(Serial.available())
    {
        char c = Serial.read();
        switch(c)
        {
            case '1': t = 1; break;
```

```
            case '2': t = 2; break;
            case '3': t = 3; break;
            case '4': t = 4; break;
            case '5': t = 5; break;
        }
        EEPROM.update(0,t); //store t in location 0 if needed
    }
}
```

مثال ۱۷-۴ (برای علاقه مندان)

برنامه ای که در مثال ۷-۳ نوشته اید را با استفاده از تابع millis که در فصل ۱۳ آموخته اید بازنویسی کنید.

پاسخ:

```
#include <EEPROM.h>

byte t;
void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);

  t = EEPROM.read(0); //read location 0
  if((t == 0)||(t > 5)) //if an invalid value
  {
    t = 1;
  }
}

unsigned long lastTime = 0;
byte lastState = LOW;

void loop()
{
  if(millis() > lastTime + (t*1000) )
  {
    lastTime = millis();
    if(lastState == HIGH) //if 13 is HIGH
    {
      lastState = LOW;
      digitalWrite(13,LOW); //make 13 LOW
    }
    else
    {
      lastState = HIGH;
      digitalWrite(13,HIGH); //otherwise, make 13 HIGH
```

```
    }
  }

  if(Serial.available())
  {
    char c = Serial.read();
    switch(c)
    {
      case '1': t = 1; break;
      case '2': t = 2; break;
      case '3': t = 3; break;
      case '4': t = 4; break;
      case '5': t = 5; break;
    }
    EEPROM.update(0,t); //store t in location 0 if needed
  }
}
```

توضیح: مزیت برنامه فوق نسبت به برنامه مثال ۷-۳، این است که در برنامه مثال ۷-۳ در طی مدتی که در حال اجرا کردن دستور delay است نمی‌تواند وضعیت سریال را چک کند و اگر از طریق سریال، کاراکتری را ارسال کنیم، می‌بایست اول delay ها بر حسب مدتی که از قبل اعلام کرده بودیم، سپری شود و سپس کاراکتر دریافتی مورد بررسی قرار گیرد. اما در برنامه فوق، در طی مدتی که منتظر سپری شدن زمان جهت روشن و خاموش کردن پایه ۱۳ هستیم، دائما وضعیت سریال را زیر نظر داریم و به مجرد اینکه کاراکتری دریافت شود، زمان بندی چشمک زدن پایه ۱۳ بر حسب کاراکتر دریافتی تغییر می‌کند.

بخش ۱۷-۳: ذخیره کردن متغیرهای int در حافظه EEPROM

همان طوری که در بخش قبل گفته شد، هر خانه از EEPROM ظرفیت ذخیره کردن یک بایت را دارد. در این قسمت می‌بینیم می‌خواهیم متغیرهای int را درون EEPROM ذخیره کنیم. متغیر int برای ذخیره شدن به ۲ بایت احتیاج دارد. بنابراین می‌بایست آن را به ۲ تکه یک بایتی خرد کنیم و هر قسمت آن را در یک خانه از EEPROM ذخیره کنیم.

در برنامه نویسی آردوینو برای جدا کردن بایت با ارزش و کم ارزش متغیرهای int می‌توانیم از highByte() و lowByte() استفاده کنیم. به طور مثال، در زیر، بایت با ارزش از متغیر i را در متغیر a و بایت کم ارزش از متغیر i را در متغیر b ریخته ایم:

```
int i = 15326;
byte a, b;
```

```
a = highByte(i);
b = lowByte(i);
```

همچنین در برنامه زیر، قسمت کم ارزش از متغیر temp را در خانه ۵ از حافظه EEPROM، و قسمت با ارزش از متغیر temp را در خانه ٦ از حافظه EEPROM ذخیره کرده‌ایم:

```
EEPROM.write(5,lowByte(temp));
EEPROM.write(6,highByte(temp));
```

برای اینکه دو تا بایت را به هم بچسبانیم می‌توانیم از word(,) استفاده کنیم:

```
word(بایت با ارزش ,بایت کم ارزش);
```

به طور نمونه، در زیر محتوای متغیرهای a و b را به هم می‌چسبانیم و حاصل را در متغیر i می‌ریزیم:

```
byte a = 95;
byte b = 123;
int i;
i = word(a,b); // i = a b
```

همچنین در زیر محتوای خانه‌های ۵ و ٦ از حافظه EEPROM را می‌خوانیم و آنها را به هم می‌چسبانیم تا یک متغیر از جنس int به دست آید:

```
byte l, h;
l = EEPROM.read(5);
h = EEPROM.read(6);
int i = word(h,l);
```

تمرین

۱. برنامه ای بنویسید که خانه های ۰ تا ۱۰ از حافظه EEPROM را با اعداد زوج بین ۲ تا ۲۲ پر کند.

۲. برنامه ای بنویسید که هر ثانیه یکبار، کاراکتری را از طریق سریال به کامپیوتر بفرستد. کاراکتری که به کامپیوتر می‌فرستد، آخرین کاراکتری است که از کامپیوتر دریافت کرده است. مایلیم که اگر برد آردوینو خاموش شد، پس از روشن شدن همان کاراکتری را که قبل از خاموش شدن می‌فرستاد را به کامپیوتر بفرستد. بنابراین کاراکتر ارسالی می‌بایست درون EEPROM ذخیره شود.

۳. می‌خواهیم با استفاده از آردوینو یک ثانیه شمار بسازیم. به گونه ای که ابتدا از طریق سریال عدد ۰ را به کامپیوتر بفرستد و بعد از یک ثانیه عدد ۱ را بفرستد و سپس ۲ را بفرستد و الی آخر. همچنین اگر برد آردوینو خاموش شد، شمارش را از آخرین عددی که به کامپیوتر فرستاده‌ایم ادامه دهد.

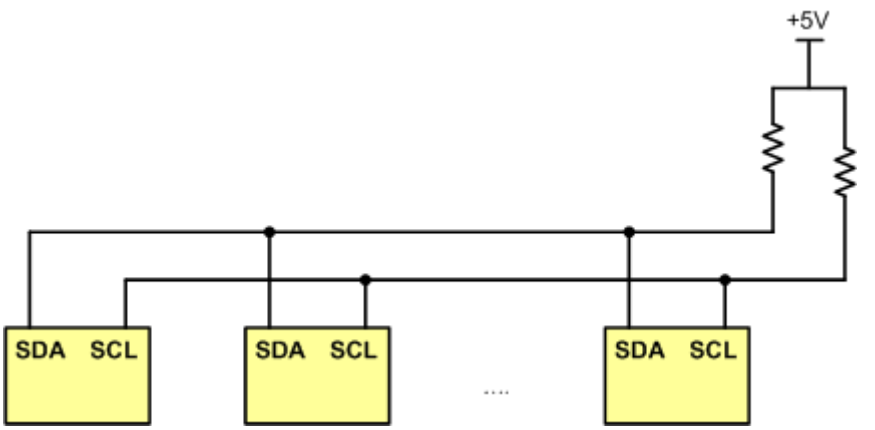
فصل ۱۸: ساعت و پروتکل I2C

بخش ۱۸-۱: آشنایی با پروتکل I2C

برای اینکه یک سیستم سخت افزاری درست کنیم، ممکن است لازم شود تعداد زیادی وسایل مختلف از قبیل نمایشگرها، سنسورهای مختلف، ورودیها و خروجیها، آی سی ساعت و غیره را به برد آردوینو خود متصل کنیم و اگر قرار باشد برای اتصال هر وسیله تعدادی سیم به برد آردوینو متصل کنیم، ممکن است لازم باشد تعداد زیادی سیم به برد آردوینو متصل کنیم و ممکن است در مواقعی پایه آردوینو کم بیاید.

کمپانی فیلیپس با در نظر گرفتن این مشکلات در سال ۱۹۸۲ پروتکلی به نام I2C را ایجاد کرد، با این هدف که بتوانیم از طریق فقط دو رشته سیم تعداد زیادی وسایل را با هم مرتبط کنیم. بعدها شرکتهای دیگری نیز محصولاتی تولید کردند که از پروتکل I2C پشتیبانی می‌کنند.

در عکس زیر طریقه متصل کردن وسایل از طریق پروتکل I2C به هم را ملاحظه می‌کنید. همان طوری که می‌بینید همه وسایل از طریق دو رشته سیم به هم مرتبط شده اند که این سیمها SDA (Serial Data line) و SCL (Serial Clock line) نامیده شده‌اند. این دو رشته سیم از طریق دو تا مقاومت به VCC متصل شده‌اند. بنابراین زمانی که وسایل از این سیمها برای تبادل اطلاعات استفاده نمی‌کنند، این سیمها از طریق مقاومتها به ۵ ولت متصل می‌شوند و ولتاژ این سیمها ۵ ولت است.



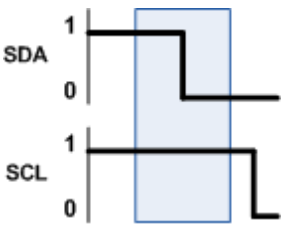
شکل ۱۸-۱: متصل کردن وسایل از طریق I2C

زمانی که وسیله‌ای بخواهد با وسیله دیگری تبادل اطلاعات کند، بر روی سیم SCL موجی مربعی ایجاد می‌کند و از سیم SDA برای مشخص کردن مخاطب خود و تبادل اطلاعات استفاده می‌کند. به وسیله ای

که آغاز کننده تبادل اطلاعات است، اصطلاحا رئیس (master) و به وسیله‌ای که منتظر می‌ماند تا تبادل اطلاعات با او صورت گیرد فرمانبر (slave) می‌گویند. معمولا برد آردوینو در نقش رئیس است و سایر وسایلی که به برد آردوینو متصل می‌شوند در نقش فرمانبر هستند.

تبادل اطلاعات طی مراحل زیر بین رئیس و فرمانبر صورت می‌گیرد:

۱. ***آغاز تبادل اطلاعات (Start Condition)***: ابتدا رئیس پایه SDA را پایین می‌کشد. زمانی که وسیله‌ای سیم SDA را پایین بکشد، سایر وسایلی که به خطوط I2C متصل هستند، متوجه می‌شوند که سیم SDA پایین کشیده شده و بنابراین وسیله‌ای قصد تبادل اطلاعات دارد. بنابراین به اطلاعاتی که روی سیمهای I2C گذاشته می‌شود، توجه می‌کنند تا در گام بعدی ببینند که رئیس با چه کسی کار دارد.



شکل ۱۸-۲: اعلام آغاز تبادل

۲. ***اعلام آدرس مخاطب:*** در گام دوم، رئیس آدرسِ مخاطبِ خود را بر روی سیم SDA می‌گذارد. به هر یک از وسایلی که به خط I2C متصل شده، عدد منحصر به فردی اختصاص یافته است که برای اینکه مخاطب خود را مشخص کنیم از این عدد استفاده می‌کنیم. اصطلاحا به این عدد آدرس می‌گویند. آدرس وسایل یک عدد ۷ بیتی است و می‌تواند بین ۱ تا ۱۲۰ باشد. هنگامی که رئیس آدرس مخاطب خود را روی خط SDA می‌گذارد، جمیع وسایلی که به I2C متصل هستند، آدرس را دریافت می‌کنند و وسیله ای که آدرسش بر روی خط I2C قرار گرفته است، می‌داند که رئیس می‌خواهد با او تبادل اطلاعات داشته باشد. ممکن است این سوال برای شما مطرح شود که چگونه عددی ۷ بیتی را بر روی یک رشته سیم SDA قرار می‌دهیم. هر عددی که قرار است بر روی سیم تبادل شود، ابتدا به باینری تبدیل می‌شود و سپس یک بیت یک بیت بر روی سیم SDA قرار می‌گیرد و هم زمان با قرار گرفتن هر بیت جدید، رئیس یک کِلاک بر روی پایه SCL ایجاد می‌کند. (یعنی پایه SCL را ابتدا یک می‌کند و سپس صفر می‌کند.) هنگامی که سیم SCL از یک به صفر تغییر می‌کند،

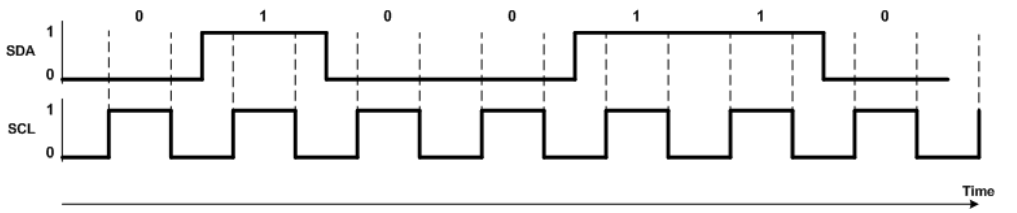
سایر وسایلی که به I2C متصل هستند، سیم SDA را می‌خوانند و به این ترتیب بیتها یکی یکی پشت سر هم بر روی خط SDA منتقل می‌شود.

مثال ۱۸-۱

تصور کنید که یک رئیس بخواهد با وسیله ای با آدرس ۳۸ تبادل اطلاعات داشته باشد. مقادیر اعمال شده به سیمهای SCL و SDA را رسم کنید.

پاسخ:

عدد ۳۸ را که به باینری تبدیل کنیم، عدد ۰۱۰۰۱۱۰ به دست می‌آید. (با توجه به اینکه آدرس می‌بایست ۷ بیتی باشد، در سمت چپ عدد آن قدر ۰ می‌گذاریم تا تعداد ارقام ۷ بیت شود.) رئیس، بیتهای آدرس را به ترتیب از پرارزش به کم ارزش بر روی سیم می‌گذارد و بعد از قرار دادن هر بیت، پایه SCL را بالا می-کشد و سپس پایین می‌کشد.



۳. ***اعلام ارسال یا دریافت کردن:*** در گام بعدی رئیس اعلام می‌کند که می‌خواهد به وسیله مخاطب اطلاعاتی بفرستد یا اینکه می‌خواهد از آن اطلاعاتی دریافت کند. اگر بخواهد بفرستد، خط SDA را صفر می‌کند و بر روی پایه SCL، یک کلاک ایجاد می‌کند. اما اگر بخواهد دریافت کند، خط SDA را یک می‌کند و بر روی پایه SCL، کلاک ایجاد می‌کند.

٤. ***موافقت با تبادل اطلاعات:*** اگر وسیله ای با آدرسی که رئیس اعلام کرده است بر روی خط I2C باشد، پایه SDA را به مدت یک کِلاک (پایه SCL، یکبار، بالا برود و مجددا پایین بیاید.) پایین می‌کشد. اما اگر هیچ وسیله ای با این آدرس آماده تبادل اطلاعات نباشد، خط SDA، بالا (یک) باقی می‌ماند. اگر خط SDA پایین کشیده شده باشد، رئیس می‌داند که وسیله مخاطب آماده تبادل اطلاعات است و بنابراین به تبادل اطلاعات ادامه می‌دهد. اما اگر خط SDA بالا باقی بماند، رئیس می‌داند که چنین وسیله‌ای وجود ندارد یا آمادگی تبادل اطلاعات ندارد.

۵. ***انجام تبادل اطلاعات:*** در این مرحله یک بایت دیتا بین رئیس و فرمانبر تبادل می‌شود.

۶. ***تایید دریافت:*** پس از اینکه فرستنده یک بایت را ارسال کرد، خط SDA را رها می‌کند. در این حالت اگر گیرنده بخواهد تایید (ACK) بفرستد، پایه SDA را پایین می‌کشد و در غیر این صورت به SDA دست نمی‌زند تا به معنی عدم تایید (NACK) تلقی شود. مفهوم بیت تایید دریافت در آی سی های مختلف متفاوت است. ممکن است یک وسیله دریافت کننده هنگامی که دیتا را به درستی دریافت کند و مورد قبولش باشد، تایید (ACK) بفرستد و در غیر این صورت عدم تایید (NACK) بفرستد. در حالی که وسیله گیرنده دیگری ممکن است زمانی که بخواهد بایت دیگری دریافت کند تایید بفرستد و هنگامی که نخواهد بایت دیگری دریافت کند، عدم تایید (NACK) بفرستد.

۷. ***اعلام اتمام تبادل اطلاعات (Stop):*** هنگامی که کار تبادل اطلاعات به پایان می‌رسد، ابتدا سیم SCL را رها می‌کند تا یک شود و سپس سیم SDA را از ۰ به یک تغییر می‌دهد.

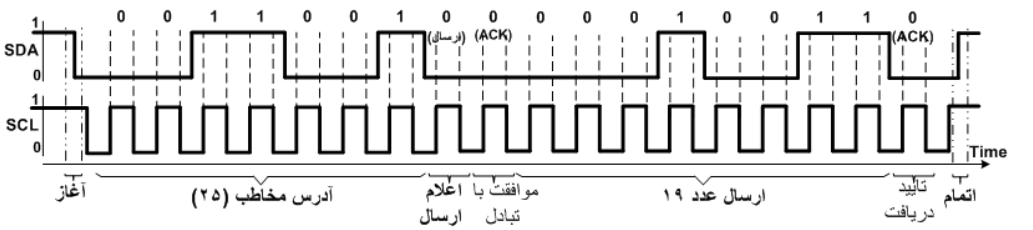
مثال ۱۸-۱

تصور کنید که یک رئیس بخواهد به وسیله ای با آدرس ۲۵ عدد ۱۹ را بفرستد. مقادیر اعمال شده به سیمهای SCL و SDA را رسم کنید. فرض کنید که وسیله ای با آدرس ۲۵ بر روی خط وجود دارد.

پاسخ:

عدد ۲۵ را که به باینری تبدیل کنیم، عدد ۰۰۱۱۰۰۱ به دست می‌آید. همچنین عدد ۱۹ را که به باینری تبدیل کنیم عدد ۰۰۰۱۰۰۱۱ به دست می‌آید.

رئیس ابتدا بر روی خط آغاز تبادل اطلاعات را اعلام می‌کند و سپس آدرس مخاطب (25) را اعلام می‌کند و مشخص می‌کند که می‌خواهد اطلاعاتی را برای مخاطب بفرستد و سپس به اندازه یک کلاک خط SDA را رها می‌کند. وسیله ای که آدرس آن ۲۵ است خط SDA را به اندازه یک کلاک پایین می‌کشد. در گام بعد رئیس عدد ۱۹ را به فرمانبر ارسال می‌کند و در جوابش فرمانبر تایید دریافت (ACK) می‌فرستد و در انتها، رئیس اعلام اتمام تبادل اطلاعات می‌کند.



برای اینکه بتوانیم از I2C استفاده کنیم لزومی ندارد که تمام توضیحات فوق را با جزئیات به خاطر بسپاریم. بلکه همین قدر کافی است که بدانیم که:

- برای ارتباط وسایل از طریق پروتکل I2C می‌بایست پایه های SDA جمیع وسایل را به هم متصل کنیم و یک مقاومت ۱۰ کیلو اهمی بین پایه SDA و ولتاژ ۵ ولت قرار دهیم. پایه های SCL جمیع وسایل را نیز می‌بایست به هم متصل کنیم و یک مقاومت ۱۰ کیلو اهمی نیز بین پایه SCL و ولتاژ ۵ ولت قرار دهیم.

- جمیع وسایلی که قرار است از طریق I2C با هم مرتبط شوند، دارای آدرسی هستند. این آدرس در برخی از وسایل ثابت است. مثلا در آی سی های DS1307 و DS3231 آدرس آنها همیشه ۱۰٤ است. اما در برخی دیگر از وسایل آدرس قابل تغییر است. مثلا در آی سی AT2432 که یک آی سی EEPROM است، آدرس آن قابل انتخاب است.

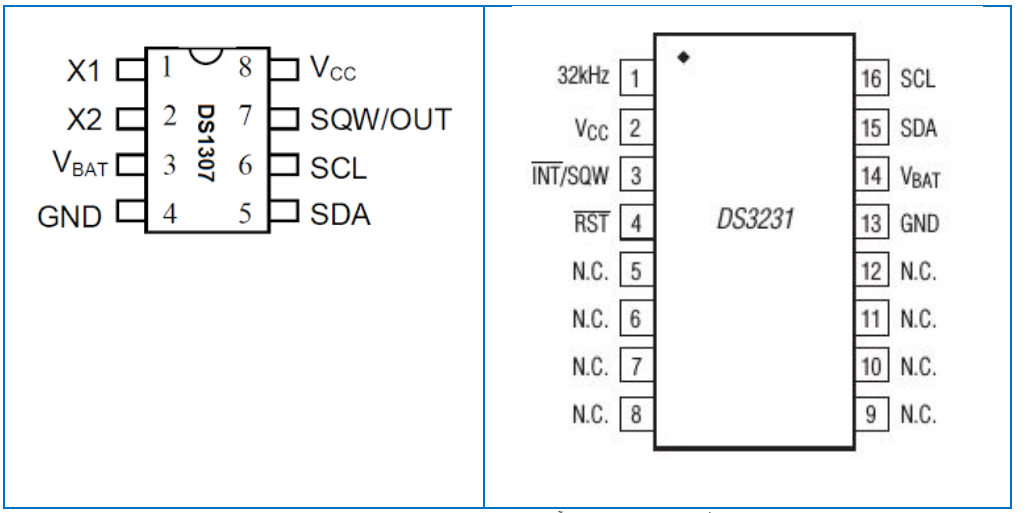
بخش ۱۸-۲: آشنایی با آی سی ساعت DS3231

در بعضی از وسایلی که می‌سازیم ممکن است نیاز باشد که ساعت و تاریخ جاری را بدانیم. مثلا ممکن است بخواهیم سر ساعت مشخصی وسیله‌ای برقی را روشن کنیم. برای اینکه ساعت و تاریخ جاری را بدانیم می‌توانیم از آی سی هایی که به آی سی های ساعت (RTC) معروف هستند استفاده کنیم. (RTC مخفف Real-Time Clock است.) از جمله آی سی های ساعت رایج در بازار DS1307 و DS3231 هستند. طریقه استفاده کردن از DS3231 و DS1307 درست مثل هم است، به گونه ای که اگر برنامه ای را برای DS3231 نوشته باشیم می‌توانیم دقیقا همان برنامه را برای DS1307 نیز استفاده کنیم. اما DS3231 جدیدتر است و دقت آن نیز نسبت به DS1307 بیشتر است و بنابراین برای تولیدات جدید بهتر است که از DS3231 استفاده کنیم.

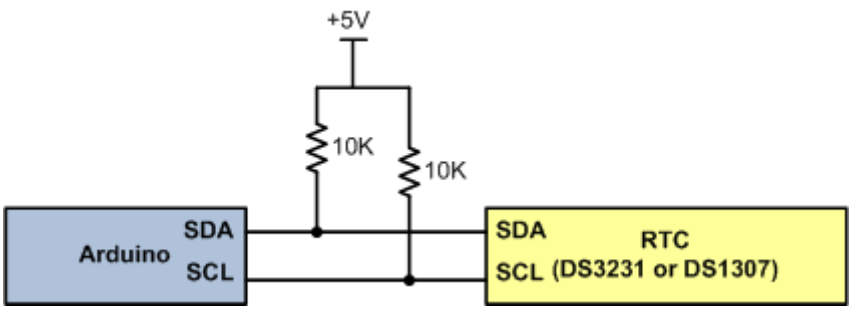
پایه های DS1307 و DS3231

در شکل ۱۸-۳ پایه‌های آی‌سی‌های DS1307 و DS3231 را می‌بینید. پایه‌هایی که برای نگهداری تاریخ و ساعت کاربرد دارند به قرار زیرند:

- پایه های SCL و SDA: از طریق این پایه‌ها آی سی ساعت را به برد آردوینو متصل می-کنیم. همان طوری که در بخش پیش دیدیم، پایه‌های SCL و SDA می‌بایست نظیر به نظیر به هم متصل شوند. یعنی مطابق شکل ۱۸-٤.

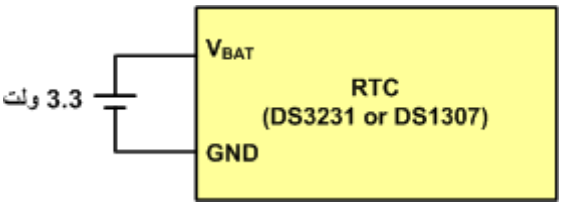


شکل ۱۸-۳: پایه های آی سی های DS1307 و DS3231



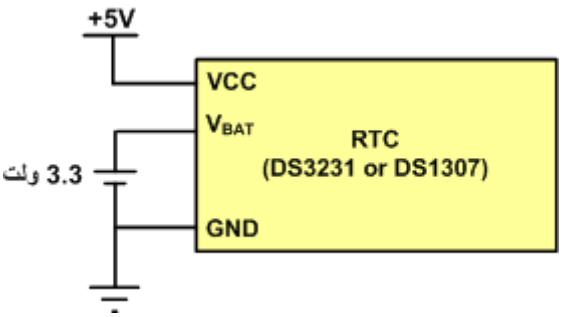
شکل ۱۸-۴: متصل کردن پایه های SDA و SCL به آردوینو

- پایه VBAT: زمانی که برق سیستم قطع می‌شود و آردوینو خاموش می‌شود، آی سی ساعت باید به کار خودش ادامه دهد و ساعت و تاریخ را نگه دارد. بنابراین بر روی آی سی های ساعت پایه VBAT در نظر گرفته شده است که آن را به باتری 3.3 ولت متصل می‌کنیم تا برق مورد نیاز جهت کار کردن آی سی ساعت در زمانی که سیستم خاموش است را فراهم کند. در شکل زیر طریقه متصل کردن باتری به پایه VBAT را می‌بینید. با توجه به اینکه مصرف برق آی سی های ساعت خیلی کم است، یک باتری ساعت ساده می‌تواند برق مورد نیاز آی سی ساعت را برای چندین سال فراهم کند. به همین خاطر در اکثر کاربردها در کنار آی سی ساعت یک جا باتری ساعتی تعبیه می‌شود.



شکل ۱۸-۵: متصل کردن باتری به RTC

- پایه‌های VCC و GND: این پایه‌ها را می‌بایست به ترتیب به پایه‌های 5+ ولت و GND از برد آردوینو متصل کنیم تا برق مورد نیاز جهت کار کردن آی سی ساعت را فراهم کند. هنگامی که برق این پایه ها فراهم شود، می‌توانیم از طریق پروتکل I2C با آی سی ساعت مرتبط شویم. اما اگر برق این پایه‌ها قطع باشد، برای اینکه مصرف برق آی سی ساعت کاهش یابد، اکثر قسمتهای آی سی ساعت (بجز قسمتی که کار شمارش ساعت و تاریخ را برعهده دارد) خاموش می‌شوند و بنابراین در این حالت آی سی ساعت به پایه های I2C خود واکنشی نشان نمی‌دهد.



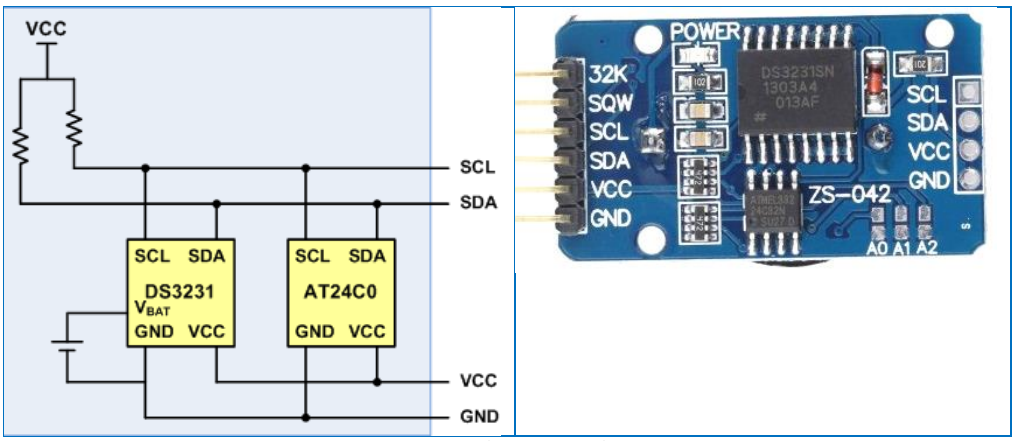
شکل ۱۸-۶: فراهم کردن برق برای RTC

- پایه های کریستال در DS1307: ساعتها برای اینکه هر ثانیه یکبار، مقدار ثانیه شمار را زیاد کنند، نیاز به کریستالی دارند که موجی مربعی ایجاد کند. آی سی DS1307، دو تا پایه به نامهای X1 و X2 دارد که می‌بایست یک کریستال با فرکانس 32,768 هرتز بین این دو پایه قرار دهیم. اما در آی سی DS3231، کریستال در دل خود آی سی جاسازی شده است و نیازی به کریستال ندارد.

ماژولهای ساعت

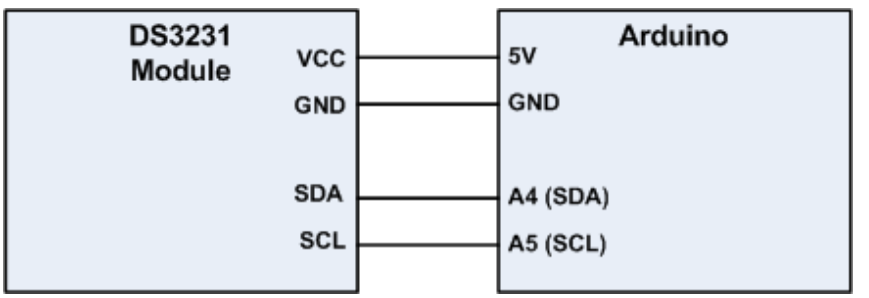
در زیر عکس یک ماژول آماده DS3231 و نقشه شماتیک آن را می‌بینید. با توجه به اینکه فاصله بین پایه‌های DS3231 کم است، متصل کردن سیم به پایه های DS3231 مشکل است. بر روی ماژول DS3231

یک جا باتری برای قرار دادن باتری ساعت وجود دارد و پایه‌های مورد نیاز را بر روی سوکت در اختیار ما قرار داده است به گونه‌ای که به راحتی بتوانیم آنها را به برد آردوینو متصل کنیم.



شکل ۱۸-۷: ماژول DS3231

برای اینکه از ماژول DS3231 فوق استفاده کنیم، کافی است که یک باتری عدسی ۳ ولت در جا باتری آن قرار دهیم و پایه‌های SCL و SDA و VCC و GND ماژول را نظیر به نظیر به صورت زیر به برد آردوینو متصل کنیم.



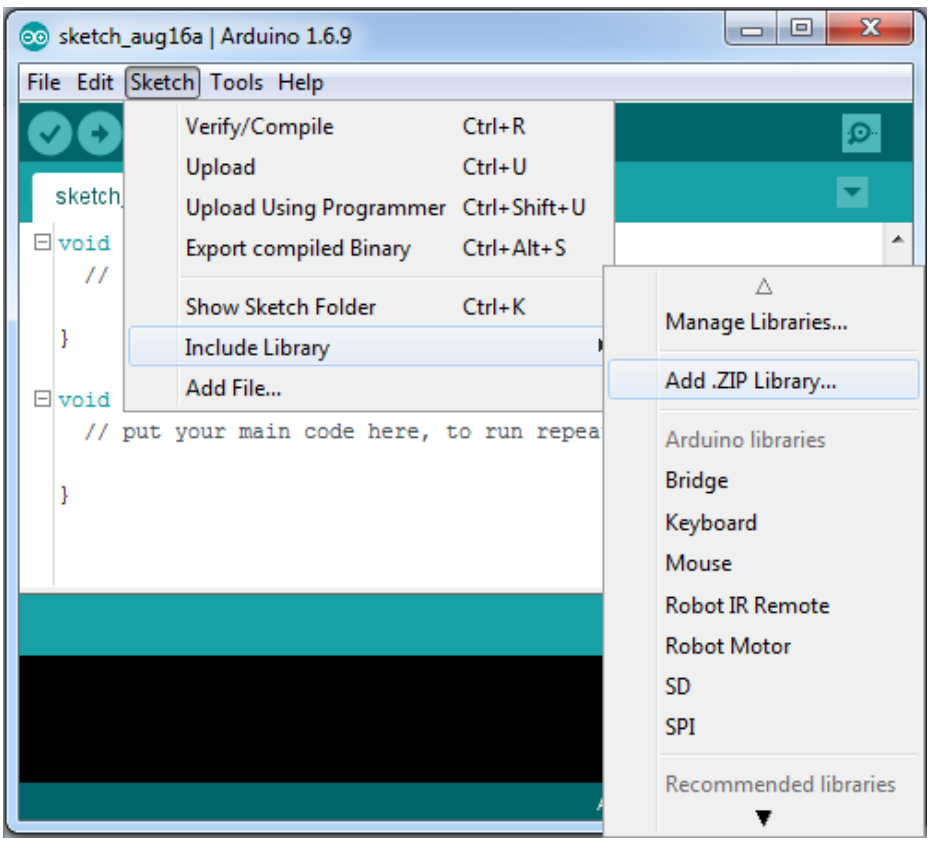
شکل ۱۸-۸: متصل کردن ماژول DS3231 به برد آرودینو

بخش ۱۸-۳: برنامه نویسی برای آی سی های DS1307 و DS3231

برای اینکه بتوانیم به سهولت از آی سی های ساعت در محیط آردوینو استفاده کنیم، کتابخانه هایی را قبلا افراد مختلفی آماده کرده اند که ما در این بخش از کتابخانه DS3231 که توسط رینکی دینک نوشته شده است، استفاده می‌کنیم.

جهت دانلود کردن کتابخانه مربوط به DS3231 به سایت www.RinkyDinkElectronics.com بروید و در قسمت Libraries، بر روی DS3231 کلیک کنید و در صفحه‌ای که باز می‌شود بر روی فایل

DS3231.zip کلیک کنید تا دانلود شود. (محض اطمینان، یک کپی از این فایل بر روی سایت کتاب با آدرس www.NicerLand.ir قرار گرفته است که در صورت تمایل می‌توانید این فایل را از روی سایت کتاب نیز دانلود کنید.) سپس به محیط آردوینو بروید و در منوی Sketch بر روی Include Library کلیک کنید و سپس بر روی Add .Zip Library کلیک کنید. در پنجره‌ای که باز می‌شود، به محلی که فایل DS3231.zip در آن دانلود شده بروید (در محیط ویندوز معمولا فایلها در دایرکتوری Downloads دانلود می‌شوند.) و فایل DS3231.zip را انتخاب کنید و دکمه Open را بزنید تا کتابخانه DS3231 نصب شود.



شکل ۱۸ - ۹: افزودن کتابخانه از طریق فایل zip

برنامه نویسی با کتابخانه DS3231

برای استفاده کردن از کتابخانه DS3231 در بالای برنامه فایل DS3231.h را اضافه (include) می‌کنیم و سپس متغیری از جنس DS3231 تعریف می‌کنیم. هنگام تعریف کردن این متغیر، شماره پایه‌هایی از برد آردوینو که به پایه‌های SDA و SCL از ماژول DS3231 متصل شده‌اند را اعلام می‌کنیم. مثلا اگر پایه‌های ٤ و ٥ از آردوینو به ترتیب به پایه‌های SDA و SCL متصل باشند، عبارت زیر را می‌نویسیم:

```
DS3231 rtc(4,5);
```

سپس جهت آماده به کار شدن کتابخانه تابع begin را صدا می‌زنیم:

```
rtc.begin();
```

تنظیم کردن ساعت و تاریخ

این کتابخانه تابعی به نام setTime دارد که هرگاه مایل باشیم، با استفاده از آن می‌توانیم ساعت آی‌سی DS3231 را تنظیم کنیم. آرگومانهایی که به این تابع می‌دهیم، به ترتیب ساعت و دقیقه و ثانیه هستند. به طور مثال، دستور زیر ساعت DS3231 را به ساعت ۱۰ و ۱۵ دقیقه و ۱۹ ثانیه (10:15:19) تنظیم می‌کند:

```
rtc.setTime(10,15,19);
```

همچنین با استفاده از تابع setDate می‌توانیم تاریخ جاری را تنظیم کنیم. به طور مثال دستور زیر تاریخ را بر روی هفتم ژانویه (اولین ماه میلادی) سال ۲۰۱۷ تنظیم می‌کند:

```
rtc.setDate(7,1,2017);
```

برنامه زیر ساعت و تاریخ DS3231 را به ٤ ژانویه ۲۰۱۸ ساعت ۱۹ و ۱۵ دقیقه و ۸ ثانیه مقداردهی می‌کند. توجه داشته باشید که زمانی که DS3231 را از بازار خریداری می‌کنیم، باتری به آن متصل نیست و ساعت و تاریخ آن تنظیم نمی‌باشد. بنابراین بعد از اینکه باتری را درون جا باتری قرار می‌دهیم لازم است که ساعت و تاریخ آن را تنظیم کنیم. برای تنظیم کردن ساعت و تاریخ می‌توانید از برنامه زیر استفاده کنید و البته می‌بایست مقادیر ساعت و تاریخ را بر حسب زمانی که دارید، DS3231 را تنظیم می‌کنید، تغییر دهید.

برنامه ۱۸-۱: تنظیم کردن ساعت و تاریخ

```
#include <DS3231.h>
DS3231 rtc(A4,A5);   //SDA and SCL are connected to pins A4 and A5.
void setup()
{
    rtc.begin();
    rtc.setTime(19,15,8);
    rtc.setDate(4,1,2018);
}
void loop()
{
}
```

دریافت ساعت و تاریخ جاری

برای دریافت تاریخ و ساعت جاری تابع getTime را صدا می‌زنیم. این تابع به عنوان مقدار برگشتی متغیری از جنس Time برمی‌گرداند که حاوی تاریخ و ساعت است. متغیر Time دارای اجزای year، mon (مخفف month)، date، hour، min (مخفف minute)، sec (مخفف second) و dow (مخفف day of week) است. در زیر برنامه‌ای نوشته‌ایم که زمان و تاریخ جاری را هر یک ثانیه یکبار از DS3231 می‌خواند و از طریق سریال به کامپیوتر می‌فرستد:

برنامه ۱۸-۲: دریافت ساعت و تاریخ

```
#include <DS3231.h>
DS3231 rtc(A4,A5);   //SDA and SCL are connected to pins A4 and A5.

void setup()
{
    rtc.begin();
    Serial.begin(9600);
}
void loop()
{
    Time t = rtc.getTime();
    // day/month/year
    Serial.print(t.date);
    Serial.print("/");
    Serial.print(t.mon);
    Serial.print("/");
    Serial.print(t.year);
    Serial.print(" ");
    // hour:min:sec
    Serial.print(t.hour);
    Serial.print(":");
    Serial.print(t.min);
    Serial.print(":");
    Serial.println(t.sec);

    delay(1000);
}
```

تمرین

۱. برنامه‌ای بنویسید که اگر ساعت جاری بین ۹:۳۰:۰۰ تا ۱۰:۲۰:۰۰ بود، پایه ۱۳ را روشن کند تا باغچه آبیاری شود و در غیر این صورت پایه ۱۳ خاموش باشد.

۲. برنامه‌ای بنویسید که ساعت و تقویم جاری را بر روی LCD نمایش دهد. همچنین از کلیدهایی که روی ماژول LCD قرار دارد، برای تنظیم کردن ساعت استفاده کنید. بدین صورت که اگر دکمه بالا زده شد، مقدار ساعت یکی زیاد شود و اگر دکمه راست زده شد، مقدار دقیقه یکی زیاد شود.

۳. برنامه ای بنویسید که در طی روزهای هفته پایه ۱۳ روشن باشد و جمعه‌ها پایه ۱۳ خاموش باشد.

٤. برنامه‌ای بنویسید که روز اول ماه پایه ۱۳ روشن باشد و سایر روزهای ماه پایه ۱۳ خاموش باشد.

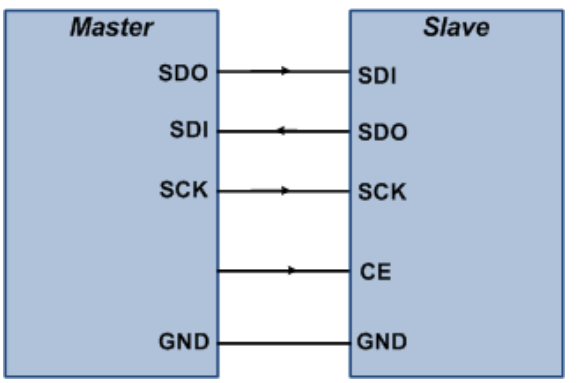
۵. در گذشته ساعتها سر هر ساعتی به تعداد آن ساعت، صدا ایجاد می‌کردند. مثلا سر ساعت ۱۱، به تعداد ۱۱ مرتبه صدا ایجاد می‌کردند. برنامه‌ای بنویسید که سر هر ساعت پایه ۱۳ به تعداد ساعت، چشمک بزند. مثلا سر ساعت ۹ به تعداد ۹ مرتبه چشمک بزند.

فصل ۱۹: پروتکل SPI، دسترسی به حافظه SD و مدیریت فایل

بخش ۱۹-۱: آشنایی با پروتکل SPI

پروتکل SPI از طریق چهار رشته سیم دو تا وسیله را با هم مرتبط می‌کند که از بین این دو وسیله یکی در نقش رئیس (master) و دیگری در نقش فرمانبر (slave) است. رئیس تصمیم می‌گیرد که در چه زمانی تبادل اطلاعات صورت گیرد و فرمانبر هرگاه دید که رئیس مایل به تبادل اطلاعات است بلافاصله به رئیس پاسخ می‌دهد.

وسایلی که می‌خواهند از طریق SPI تبادل اطلاعات کنند ٤ پایه دارند که عبارتند از: SDO، SDI، SCK و CE. شکل زیر را ببینید. از طریق پایه SDO (Serial Data Out)، وسایل اطلاعات را بیت به بیت ارسال می‌کنند و از طریق پایه SDI (Serial Data In)، وسایل اطلاعات را دریافت می‌کنند. رئیس هرگاه مایل باشد با فرمانبر تبادل اطلاعات کند، پایه CE از فرمانبر را پایین می‌کشد و فرمانبر هرگاه دید که پایه CE پایین کشیده شده است، می‌داند که می‌بایست آماده تبادل اطلاعات با رئیس باشد. پایه SCK پایه Clock (موج مربعی) است که طی تبادل اطلاعات، رئیس این پایه را یک و صفر (بالا و پایین) می‌کند و با هر بار بالا و پایین کردن این پایه، یک بیت رئیس برای فرمانبر می‌فرستد و همزمان یک بیت فرمانبر برای رئیس می‌فرستد. به عبارت دیگر در پروتکل SPI، به طور همزمان هم رئیس برای فرمانبر اطلاعات می‌فرستد و هم فرمانبر برای رئیس.

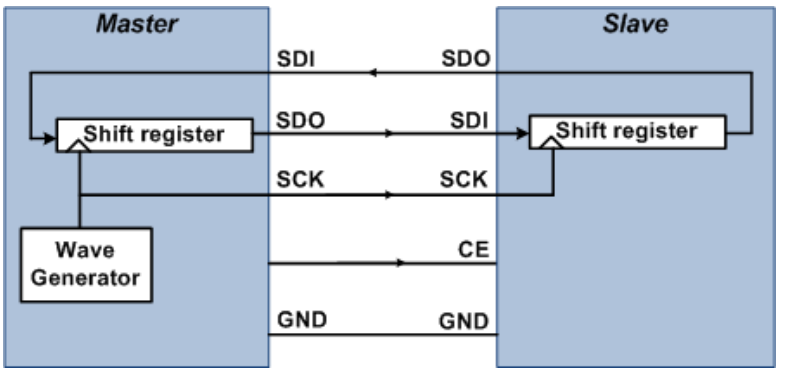


شکل ۱۹-۱: ارتباط دو وسیله از طریق پروتکل SPI

ساختار پروتکل SPI تا حدود زیادی شبیه تله کابین است. در تله کابین یک پایگاه در پایین کوه داریم و یک پایگاه در بالای کوه که یکی از پایگاه ها موتوری برای به حرکت آوردن تله کابین دارد. در تله کابین به طور همزمان که یک کابین از بالا به پایین می‌آید، یک کابین نیز از پایین به بالا می‌رود. پایگاه‌های بالا و پایین در حکم دو وسیله تبادل کننده اطلاعات هستند و کابین‌ها در حکم بیتهای داده هستند. همچنین

پایگاهی که موتور دارد در حکم رئیس است که هرگاه موتور خود را روشن کند (پایه SCK را بالا و پایین ببرد) کابین‌ها به حرکت درمی‌آیند و بین پایگاه بالا و پایین تبادل می‌شوند.

اگر بعدها درس مدارهای منطقی را بخوانید و با شیفت رجیستر آشنا شوید، خواهید دید که در عمل یک شیفت رجیستر در هر یک از وسایلی که می‌خواهند به صورت SPI تبادل اطلاعات داشته باشند وجود دارد که با هر بار بالا و پایین بردن کلاک، یک بیت از شیفت رجیستر خارج می‌شود و یک بیت وارد شیفت رجیستر می‌شود. (شکل زیر)



شکل ۱۹-۲: ساختار داخلی SPI

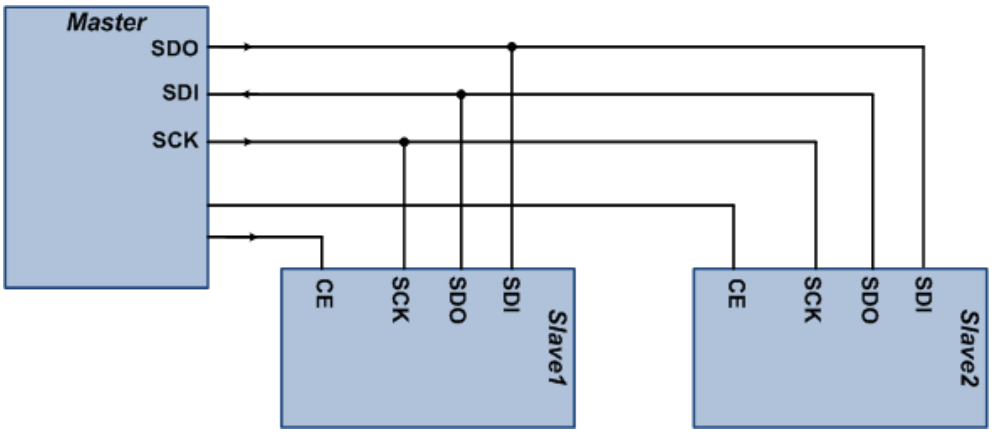
نکاتی درباره پروتکل SPI (برای علاقه مندان)

مرتبط کردن بیش از دو وسیله از طریق پروتکل SPI

در صورت تمایل می‌توانیم چندین وسیله را از طریق SPI به یک رئیس متصل کنیم که پایه CE از فرمانبرها به پایه‌های مختلفی از رئیس متصل می‌شود. (شکل ۱۹-۳) رئیس هرگاه مایل بود با هر یک از فرمانبرها تبادل اطلاعات کند، پایه CE از آن وسیله را پایین می‌کشد و پایه CE مابقی وسایل را غیرفعال نگه می‌دارد. وسیله‌ای که پایه CE آن فعال شده می‌داند که رئیس می‌خواهد با او تبادل داده کند و مابقی وسایل ساکت می‌نشینند.

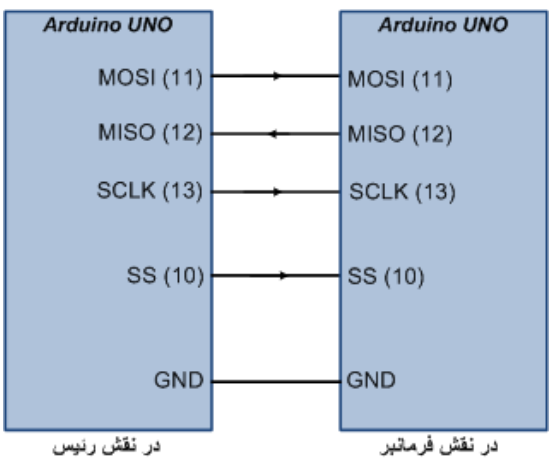
جابجا کردن نقش رئیس

نکته دیگر اینکه، هنگامی که دو تا وسیله را از طریق پروتکل SPI، به هم متصل می‌کنیم، می‌تواند نقش رئیس و فرمانبر بودن قابل جابجایی باشد. بدین معنی که هرگاه وسیله‌ای مایل بود با وسیله دیگر تبادل اطلاع کند، پایه CE از وسیله دیگر را پایین بکشد و شروع به بالا و پایین کردن پایه کلاک نماید.



شکل ۱۹-۳: مرتبط کردن بیش از یک فرمانبر به یک رئیس

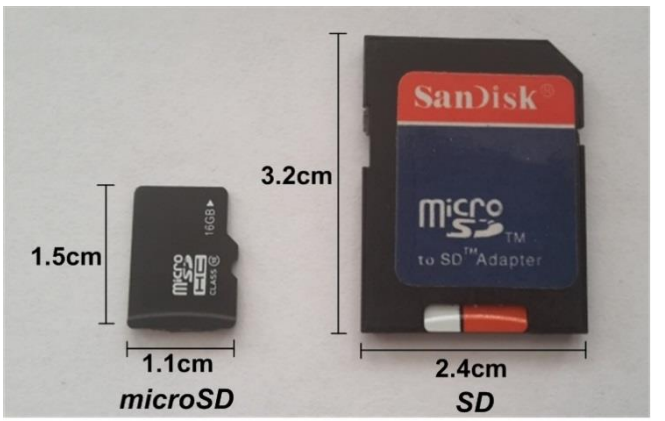
بردهای آردوینو از جمله وسایلی هستند که هم قابلیت تبادل اطلاعات به صورت رئیس را دارند و هم قابلیت تبادل اطلاعات به صورت فرمانبر را دارند. در بردهای آردوینو که با استفاده از میکروکنترلرهای AVR ساخته شده‌اند، مرسوم است که پایه‌های SPI را با نامهای MOSI، MISO، SCK و SS نامگذاری کنند. پایه MOSI مخفف Master Out Slave In است. یعنی هرگاه وسیله در نقش رئیس (master) باشد، این پایه در حکم SDO عمل می‌کند و هرگاه در نقش فرمانبر (slave) باشد در حکم SDI عمل می‌کند. همچنین پایه MISO مخفف Master In Slave Out است. بدین معنی که هرگاه وسیله در نقش رئیس (master) باشد این پایه در حکم SDI عمل می‌کند و هرگاه در نقش فرمانبر (slave) عمل کند، این پایه در حکم SDO عمل می‌کند. در شکل زیر دو تا برد آردوینو به هم متصل شده اند و جهت تبادل اطلاعات در لحظه ای که یکی از آنها رئیس و دیگری فرمانبر است مشخص شده است.



شکل ۱۹-۴ : مرتبط کردن دو برد آردوینو از طریق SPI

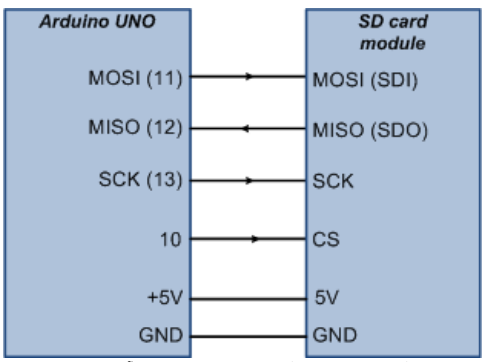
بخش ۱۹-۲: متصل کردن ماژول SD به برد آردوینو از طریق SPI

امروزه استفاده کردن از حافظه‌های SD، بسیار رواج یافته است و بسیاری از وسایل دیجیتال از قبیل گوشی‌های موبایل، لپ تاپها، دوربینها، وسایل صوتی و تصویری و غیره، سوکتی را برای اتصال حافظه‌های SD در نظر گرفته‌اند تا بتوانیم از این حافظه‌ها اطلاعاتی را دریافت کنیم یا بر روی آن اطلاعاتی را بنویسیم.

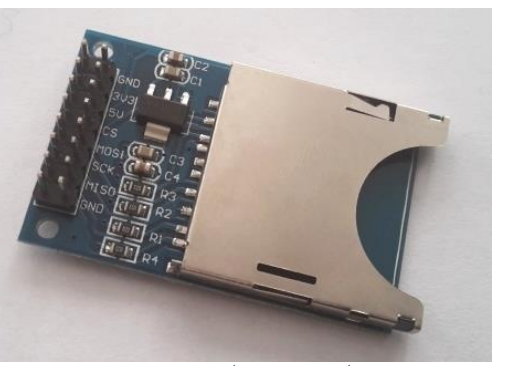


شکل ۱۹-۵: عکس میکرو SD و SD در کنار هم

برای متصل کردن حافظه‌های SD به برد آردوینو ماژولهای سوکت SD به طور آماده وجود دارند که نمونه‌ای از آن را در عکس زیر می‌بینید. نقشه متصل کردن ماژول سوکت SD به برد آردوینو در شکل زیر نمایش داده شده است.



شکل ۱۹-۷: اتصال سوکت SD به برد آردوینو



شکل ۱۹-۶: ماژول سوکت SD

در هنگام تبادل اطلاعات حافظه‌های SD همیشه در نقش فرمانبر هستند و برد آردوینو رئیس است.

بخش ۱۹-۳: نوشتن برنامه برای استفاده از حافظه SD

کتابخانه مربوط به کار کردن با حافظه SD، همراه با محیط برنامه نویسی آردوینو وجود دارد. برای اینکه از حافظه SD استفاده کنیم، می بایست فایل SD.h را در بالای برنامه خود اضافه (include) کنیم.

برای اینکه کتابخانه SD آماده استفاده کردن شود می‌بایست تابع SD.init را صدا بزنیم و هنگام صدا زدن این تابع شماره پایه‌ای از آردوینو که به پایه CS از برد SD متصل کرده‌ایم را اعلام کنیم. مثلا اگر پایه CS را به پایه ۱۰ از برد آردوینو متصل کرده‌ایم می‌بایست دستور زیر را بنویسیم:

```
SD.init(10);
```

تابع init اگر با موفقیت آماده کار کردن با حافظه SD شود، به عنوان مقدار برگشتی، مقدار true را برمی‌گرداند. اما اگر مشکلی در دسترسی به SD وجود داشته باشد، مقدار false را بازمی‌گرداند. توجه داشته باشید که برای استفاده کردن از کتابخانه SD، می‌بایست که حافظه SD به صورت FAT16 یا FAT32 فرمت شده باشد. در غیر این صورت کتابخانه SD قادر به کار کردن با حافظه SD نیست.

اگر حافظه شما به صورت FAT فرمت نشده است، آن را به کامپیوتر متصل کنید و در محیط ویندوز به درون my computer بروید و بر روی حافظه SD کلیک راست کنید و گزینه ...Format را انتخاب کنید و در لیست پایین افتادنی File system، گزینه FAT را انتخاب کنید و حافظه را فرمت کنید. توجه داشته باشید که در صورت فرمت کردن حافظه، اطلاعات درون آن پاک می‌شود، پس اگر درون حافظه SD از قبل فایلی دارید، ابتدا آن فایلها را بر روی کامپیوتر خود و یا محل دیگری کپی کنید و سپس اقدام به فرمت کردن حافظه نمایید.

باز کردن فایل

برای اینکه بتوانیم به درون فایلی بنویسیم یا از آن بخوانیم می‌بایست آن فایل را باز کنیم. برای باز کردن فایل از تابع SD.open استفاده می‌کنیم. مثلا دستور زیر فایل myfile.txt را برای نوشتن باز می‌کند:

```
File f = SD.open("myfile.txt",FILE_WRITE);
```

تابع SD.open دو تا آرگومان دریافت می‌کند. اولین آرگومان اسم فایلی است که می‌خواهیم باز کنیم و از طریق آرگومان دوم اعلام می‌کنیم که آیا می‌خواهیم به درون فایل بنویسیم یا اینکه از آن بخوانیم. اگر می‌خواهیم درون فایل بنویسیم دومین آرگومان را FILE_WRITE می‌دهیم و هرگاه بخواهیم از فایل بخوانیم آرگومان دوم را FILE_READ می‌دهیم. تابع SD.open به عنوان مقدار برگشتی متغیری از جنس File باز

می‌گرداند که با استفاده از آن می‌توانیم درون فایل بنویسیم یا از فایل بخوانیم. اگر در باز کردن فایل مشکلی پیش آمده باشد، مقدار برگشتی تابع open برابر با صفر است.

نکته دیگری که درباره باز کردن فایل وجود دارد، آدرس فایل است. اگر فایلی که می‌خواهیم باز کنیم در شاخه اصلی حافظه SD است، فقط کافی است که اسم فایل را ذکر کنیم. اما اگر در داخل زیر شاخه‌ای قرار دارد، می‌بایست آدرس دقیق فایل را نسبت به شاخه اصلی ذکر کنیم. مثلا اگر بخواهیم فایل log.txt که در دایرکتوری docs از شاخه اصلی قرار دارد را برای نوشتن باز کنیم می‌بایست دستور زیر را بنویسیم:

```
File myFile = SD.open("docs/log.txt", FILE_WRITE);
```

توجه داشته باشید که هنگامی که می‌خواهیم آدرسی را بیان کنیم اسم دایرکتوری‌ها و نیز اسم فایل را با علامت / از هم جدا می‌کنیم که در انگلیسی به آن slash می‌گویند.

نوشتن به درون فایل

برای نوشتن به درون فایل می‌توانیم از توابع println، print و write استفاده کنیم. به طور نمونه، برنامه زیر فایل mydata.txt را برای نوشتن باز می‌کند و درون آن پیغام Hello World را ذخیره می‌کند.

برنامه ۱۹-۱: نوشتن به درون فایل

```
#include <SPI.h>
#include <SD.h>

void setup()
{
    Serial.begin(9600);
    if(SD.begin(10) == false)
    {
        Serial.println("Initialization Error!");
    }
}

void loop()
{
    File myFile = SD.open("mydata.txt",FILE_WRITE);
    if(myFile != 0)
    {
        Serial.println("Writing to file");
        myFile.println("Hello World!");
        myFile.close();
    }
    else
    {
        Serial.println("Error opening file!");
    }
```

```
    delay(5000);
}
```

هنگامی که کار ما با فایلی تمام می‌شود با استفاده از تابع close فایل را می‌بندیم تا حافظه‌ای که جهت دسترسی به فایل استفاده شده است آزاد شود و در عین حال تغییراتی که بر روی فایل داده‌ایم بر روی SD اعمال شود. به عبارت دیگر تا مادامی که فایل را نبندیم تغییرات بر روی SD منتقل نمی‌شود و اگر برق آردوینو قطع شود، مطالبی که مایل بودیم بر روی SD ثبت شود، ذخیره نمی‌شود. بنابراین زمانی که کار ما با فایل تمام می‌شود، می‌بایست حتما فایل را ببندیم.

در عین حال گاهی اوقات ممکن است مطالبی که می‌خواهیم درون فایل بنویسیم هنوز ادامه داشته باشد، اما مایل باشیم که هر آنچه که تاکنون نوشته‌ایم درون SD ذخیره شود تا اگر برق قطع شد از دست نرود. در چنین مواردی از تابع flush استفاده می‌کنیم. تابع flush اگر چه فایل را نمی‌بندد، اما هر آنچه که تاکنون به درون فایل نوشته‌ایم را درون SD ذخیره می‌کند. به طور نمونه برنامه زیر مقدار ورودی یک از مبدل آنالوگ به دیجیتال را هر ثانیه یکبار می‌خواند و به درون فایل می‌ریزد. اما برای اینکه مقادیری که تاکنون به درون فایل نوشته‌ایم در اثر قطع شدن برق از دست نرود بعد از هر نوشتن دستور flush را نیز صدا زده‌ایم.

برنامه ۱۹-۲: استفاده کردن از flush

```
#include <SPI.h>
#include <SD.h>

File logFile;
void setup()
{
    SD.begin(10);
    logFile = SD.open("log.txt",FILE_WRITE);
}

void loop()
{
    if(logFile != 0)
    {
        logFile.println(analogRead(1));
        logFile.flush();
    }
    delay(1000);
}
```

خواندن از درون فایل

برای اینکه محتوای فایلی را بخوانیم می‌بایست ابتدا آن فایل را توسط تابع open باز کنیم. به طور مثال دستور زیر فایل mydata.txt را باز می‌کند و به ابتدای فایل می‌رود تا از آن بخوانیم:

```
File f = SD.open("mydata.txt",FILE_WRITE);
```

برای خواندن از فایل می‌توانیم از دستور read استفاده کنیم. هنگامی که فایلی را باز می‌کنیم اشاره‌گری به ابتدای متن فایل اشاره می‌کند و هنگامی که از فایل می‌خوانیم این اشاره‌گر به جلو می‌رود تا اینکه به انتهای فایل برسیم. از طریق تابع available می‌توانیم تشخیص دهیم که آیا درون فایل چیز دیگری برای خواندن وجود دارد یا اینکه به انتهای فایل رسیده‌ایم.

برنامه ۱۹-۳: برنامه‌ای که محتوای فایل mydata را می‌خواند و از طریق سریال به کامپیوتر می‌فرستد.

```
#include <SPI.h>
#include <SD.h>

void setup()
{
    SD.begin(10);
    Serial.begin(9600);

    File inputFile = SD.open("mydata.txt",FILE_READ);
    if(inputFile != 0)
    {
        while(inputFile.available())
        {
            char c = inputFile.read();
            Serial.print(c);
        }
        inputFile.close();
    }
}

void loop()
{
}
```

توجه داشته باشید که فایلی که برای خواندن باز می‌کنید، از قبل وجود داشته باشد. اگر از قبل بر روی حافظه SD فایلی وجود ندارد، می‌توانید در محیط ویندوز، نرم افزار notepad را باز کنید و مطالبی را تایپ کنید و سپس با زدن دکمه save مطالبی که تایپ کرده‌اید را save کنید و سپس فایلی که درست کرده‌اید را به درون حافظه SD کپی کنید.

تمرین

۱. متغیری از جنس int تعریف کنید و آن را با ۰ مقداردهی اولیه کنید و برنامه‌ای بنویسید که هر ثانیه یکبار مقدار آن را یکی زیاد کند و مقدار آن را در فایل value.txt ذخیره کند.

۲. برنامه‌ای بنویسید که فایل data.txt را باز کند و تعداد دفعاتی که حرف a در آن وجود دارد را بشمارد. (راهنمایی: یک متغیر از جنس int تعریف کنید. سپس از درون فایل، بخوانید و هرگاه کاراکتر خوانده شده، حرف a بود، یکی به مقدار متغیر int اضافه کنید و این کار را آن قدر ادامه دهید تا به انتهای فایل برسید.)

فصل ۲۰: آشنایی با برنامه نویسی شی گرا

در برنامه هایی که تاکنون نوشته‌ایم توابع و متغیرها از هم مجزا بودند. مثلا اگر در برنامه تعدادی مستطیل داشته باشیم و بخواهیم اندازه محیط و مساحت آنها را به دست آوریم، مقادیر طول و عرض مستطیل ها به صورت تعدادی متغیر تعریف می‌شوند و توابع مجزایی نیز جهت محاسبه مساحت و محیط تعریف می‌شوند.

اما ما در دنیای اطراف خود با اشیاء سر و کار داریم که هر دو مفهوم متغیر و تابع در آنها وجود دارد. به طور مثال، اتومبیل از یک سو دارای توابعی مثل جلو رفتن و عقب رفتن و چپ و راست رفتن است و از سوی دیگر دارای خصوصیاتی مثل ابعاد اتومبیل و باز و بسته بودن درب و موقعیت جغرافیایی و غیره است که خصوصیات (متغیرها) و کارایی (توابع) را با هم و در کنار هم دارد و مثلا هنگامی که عمل جلو رفتن اتومبیل اتفاق می‌افتد، متغیر موقعیت جغرافیایی اتومبیل تغییر می‌کند. بنابراین در برنامه نویسی نیز پس از مدتی ایده برنامه نویسی شی گرا (Object Oriented Programming) مطرح شد تا بتوانیم در برنامه نویسی اشیایی را ایجاد کنیم که دارای توابع و متغیرهای مرتبط به هم باشند.

به طور مثال، در برنامه نویسی شی گرا، برای مستطیل (Rectangle) کلاسی به شکل زیر به وجود می‌آوریم که از یک سو دارای توابع محیط و مساحت (area) باشد و از طرفی خصوصیات طول و عرض (x و y) را داشته باشد. در این صورت تابع مساحت که قبلا لازم بود طول و عرض مستطیل را به عنوان آرگومان بگیرد و مساحت را حساب کند، در داخل کلاس نیازی به دریافت آرگومان ندارد. بلکه ابعاد مستطیل را دارد که در هم ضرب می‌کند و مساحت را به ما می‌دهد.

```
class Rect
{
  int x, y;

  public:
  int area()
  {
    return x * y;
  }

  Rect(int x1, int y1)
  {
    x = x1;
    y = y1;
  }
  void showSpecifications()
  {
```

```
    Serial.print("It is a ");
    Serial.print(x);
    Serial.print("x");
    Serial.print(y);
    Serial.print(" rectangle.");
  }
};
```

در تعریف کردن متغیرها، ما جنس متغیر (مثلا int) داشتیم که از آن متغیری را ایجاد می‌کردیم. int خود به تنهایی از خود وجودی ندارد. بلکه از آن متغیرهایی به نام مثلا x را به صورت زیر تعریف می‌کنیم و بعدا از x برای نگهداری مقدارها استفاده می‌کنیم:

```
int x;
```

در برنامه نویسی شی گرا هم کلاس را داریم که همانند جنس متغیر از خود وجودی ندارند. بلکه از آنها اشیایی را ایجاد می‌کنیم و سپس از آن اشیاء استفاده می‌کنیم. به طور مثال از جنس Rect شیی به نام R1 را به صورت زیر تعریف می کنیم که ابعاد آن 2 و 5 است:

```
Rect R1(2,5);
```

درون کلاس، یک تابع سازنده (Constructor) وجود دارد که در زمان به وجود آمدن شی صدا زده می‌شود. مثلا در کلاس Rect، تابعی به نام Rect وجود دارد که ابعاد مستطیل را در لحظه به وجود آمدن R1 دریافت می‌کند و درون x و y که داخل کلاس Rect وجود دارند، می‌ریزد.

طریقه تعریف کردن تابع سازنده به این صورت است که تابعی هم اسم با اسم کلاس تعریف می‌کنیم و جنسی برای مقدار برگشتی آن تابع ذکر نمی‌کنیم. درست همانند تابع Rect در کلاس فوق، که قبل از آن جنس مقدار برگشتی را ذکر نکرده‌ایم.

تعداد آرگومانهایی که تابع سازنده دریافت می‌کند بسته به نیاز می‌تواند متفاوت باشد و در برخی از کلاسها تابع سازنده آرگومانی را دریافت نمی‌کند. در این موارد هنگامی که می‌خواهیم شیی را از روی کلاس به وجود آوریم جلوی اسم شی پرانتز نمی‌گذاریم. مثلا اگر سازنده کلاس Rect آرگومان ورودی نداشت، شی R1 را به صورت زیر از روی آن ایجاد می‌کردیم:

```
Rect R1;
```

شما قبلا نیز از کلاسها استفاده کرده‌اید. به طور مثال در فصل هشت، با کلاس LiquidCrystal آشنا شدید و برای استفاده کردن از LCD خطوط زیر را ابتدای برنامه خود نوشتید:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
```

هنگامی که خطوط فوق را می‌نویسیم در حقیقت از کلاس LiquidCrystal شیی به نام lcd به وجود می‌آید و شماره پینهای آردوینو که به آنها پایه های RS و Enable و غیره متصل شده اند را به عنوان آرگومان به تابع سازنده آن، ارسال می‌کنیم. سپس با صدا زدن توابعی مثل begin و setCursor که داخل کلاس LiquidCrystal وجود دارند با LCD کار می‌کنیم. خود کلاس LiquidCrystal در داخل فایل LiquidCrystal.h وجود دارد و به همین خاطر است که در بالای برنامه فایل LiquidCrystal.h را به برنامه اضافه کرده‌ایم.

همان طوری که در مثال LCD می‌بینید، استفاده کردن از کلاسها مزایای زیر را دارند:

۱. کپسوله کردن (encapsulation) جزئیات داخلِ کلاس: هنگام نوشتن کد برای LCD ما هیچگاه درگیر جزئیات اینکه توابع مربوط به LCD چگونه کار می‌کنند، نشدیم. بلکه از دید ما LCD دارای تعدادی توابع هست که با صدا زدن آنها به سهولت از LCD استفاده می‌کنیم. همچنین متغیرهایی که درون کلاس LiquidCrystal وجود دارند از دید ما پنهان هستند و بنابراین از هر گونه دسترسی و تغییر ناخواسته توسط سایر قسمتهای کد، در امان هستند و بنابراین عیب یابی برنامه نیز راحت تر است.

۲. استفاده مجدد از برنامه (Reusability): هنگامی که برنامه‌ای نوشته می‌شود، خوبست به گونه‌ای نوشته شود که بعدا بتوانیم به سهولت از قسمتهای برنامه در برنامه‌های بعدی خود استفاده کنیم. اگر برنامه‌ای دارای این خصوصیت باشد، اصطلاحا می‌گوییم که دارای خصوصیت reusability است. الان کلاس مربوط به LiquidCrystal به گونه‌ای پیاده سازی شده است که میلیونها نفر در سطح جهان که از آردوینو استفاده می‌کنند، به سهولت قادرند که در برنامه‌های خود از LCD استفاده کنند. این استفاده مجدد راحت از برکت برنامه نویسی شی گرا فراهم شده است.

کلمات کلیدی private و public:

برخی از اجزای موجود در داخل کلاس لازم است که از بیرون قابل دسترس باشند. مثلا تابع area را می‌بایست بتوانیم از بیرون از کلاس صدا بزنیم. اما برخی از اجزای داخل کلاس می‌بایست که از دید بیرون، پنهان باشند. مثلا متغیرهای x و y که داخل Rect است، خوبست که از دید بیرون از کلاس پنهان باشند تا کسی نتواند از بیرون کلاس به آن دست بزند و مقادیر آنها را عوض کند. برای اینکه اجزایی را پنهان کنیم از کلمه کلیدی private استفاده می‌کنیم. هنگامی که کلمه private را بنویسیم، هر تابع یا متغیری که بعد

از آن قرار بگیرد، به صورت پنهان (خصوصی) تعریف می‌شود. همچنین هر تابع یا متغیری که بعد از کلمه کلیدی public قرار گیرد، از خارج کلاس، قابل رئویت است. اگر هیچ یک از کلمات public یا private را در کلاس ذکر نکنیم، همه اجزا به طور پیش فرض private هستند.

یک نمونه کامل از تعریف کردن کلاس و استفاده کردن از کلاس را در برنامه زیر می‌بینید:

برنامه ۲۰-۱

```
class Rect
{
  int x, y;

  public:
  int area()
  {
    return x * y;
  }

  Rect(int x1, int y1)
  {
    x = x1;
    y = y1;
  }

  void showSpecifications()
  {
    Serial.print("It is a ");
    Serial.print(x);
    Serial.print("x");
    Serial.print(y);
    Serial.print(" rectangle.");
  }
};

void setup() {
  Serial.begin(9600);
  Rect R1(2,5);
  R1.showSpecifications();
  Serial.print(" Its area is: ");
  Serial.println(R1.area());
}

void loop() {

}
```

پیغامی که در برنامه فوق، به صورت سریال برای کامپیوتر ارسال می‌شود، در زیر نمایش داده شده است:

```
It is a 2x5 rectangle. Its area is: 10
```

در عمل خوبست که هر کلاس را در داخل یک فایل مجزا بنویسیم و نام فایل را همنام با نام کلاس و با پسوند h بگذاریم و درون هر برنامه‌ای که می‌خواهیم از آن استفاده کنیم آن را بالای برنامه include کنیم. درست مانند LiquidCrystal که داخل فایل LiquidCrystal.h قرار گرفته است و در بالای برنامه آن را include می‌کنیم.

تمرین

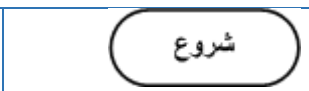
یک کلاس مشابه Rect برای مربع بنویسید. در نظر داشته باشید که در مربع طول و عرض با هم برابر است و بنابراین به عنوان خصوصیت یک مربع کافی است که طول یکی از اضلاع آن را داشته باشیم. همچنین در نظر داشته باشید که مساحت مربع از ضرب کردن طول یک ضلع مربع در خودش به دست می‌آید.

ضمیمه الف: فلوچارت (Flow chart)

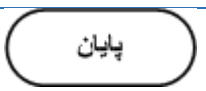
بخش الف-۱: آشنایی با فلوچارت

از جمله روشهای رایج جهت نشان دادن الگوریتم برنامه ها استفاده کردن از فلوچارت است. فلوچارت می‌تواند جزئیات یک الگوریتم را به زبان ساده بیان کند به گونه‌ای که کسانی که در برنامه نویسی مبتدی هستند نیز آن را درک کنند. ما نیز در این کتاب در مواقعی که موضوع جدیدی را مطرح می‌کنیم، برای اینکه درک موضوع راحت تر و دقیق تر باشد از فلوچارت استفاده می‌کنیم.

اجرای هر برنامه و الگوریتمی از نقطه‌ای آغاز می‌شود. در فلوچارت برای اینکه نقطه آغاز را مشخص کنند از علامت شکل آ-۱ استفاده می‌کنند. همچنین دستورات و کارهایی که انجام می‌دهیم را داخل مستطیل‌هایی می‌نویسیم و برای اینکه ارتباط و توالی کارها را نشان دهیم، فلش‌هایی را بین مستطیلها رسم می‌کنیم. برای نشان دادن نقطه‌ای که الگوریتم پایان می‌پذیرد، از علامت نشان داده شده در شکل الف-۲ استفاده می‌شود.

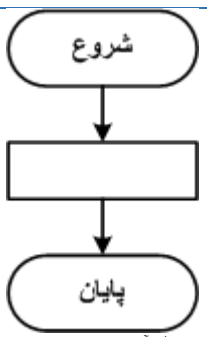


شکل آ-۱: نقطه شروع



شکل آ-۲: نقطه پایان

به طور نمونه در شکل آ-۳، ساختار کلی یک الگوریتم خیلی ساده که فقط یک کار را انجام می‌دهد نشان داده شده است. همچنین، الگوریتم آماده شدن برای مسواک زدن را در مثال زیر به صورت فلوچارت رسم کرده ایم.



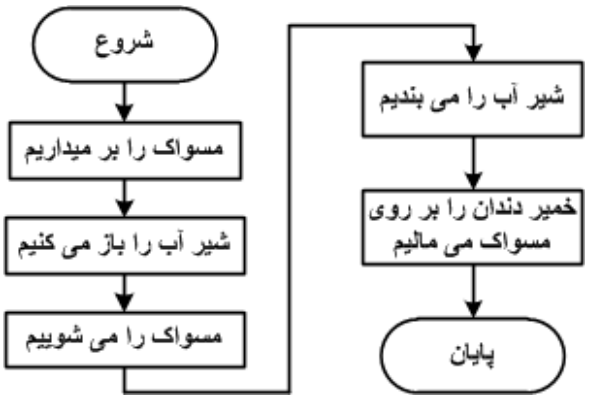
شکل آ-۳: اجزای یک فلوچارت ساده

مثال الف-۱

با فرض اینکه مراحل آماده شدن برای مسواک زدن به صورت زیر باشد، آن را به صورت فلوچارت بیان کنید:

مسواک را از جایش برمی‌داریم. سپس شیر آب را باز می‌کنیم و مسواک را می‌شوییم. سپس شیر آب را می‌بندیم و خمیر دندان را روی مسواک می‌مالیم.

پاسخ:

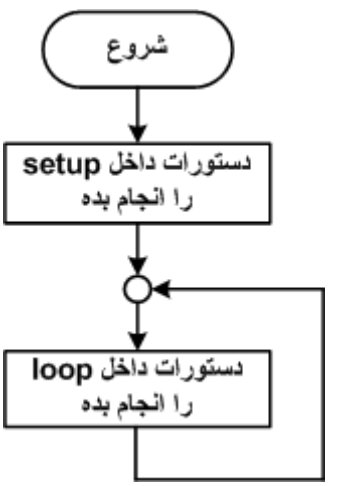


برای اینکه عملکرد ذکر شده در فلوچارتی را بررسی کنیم، از نقطه شروع آغاز می‌کنیم و فلش‌ها را دنبال می‌کنیم و کارهایی که در هر یک از مستطیلها ذکر شده را می‌بینیم تا اینکه به نقطه پایان برسیم. به طور نمونه، در مثال فوق از نقطه شروع آغاز کنید و مراحل را طی کنید تا به پایان برسید.

هنگامی که چندین فلش به یک نقطه بروند، از علامت نمایش داده شده در شکل زیر استفاده می‌کنند.

شکل آ-۴: محل به هم رسیدن چندین فلش

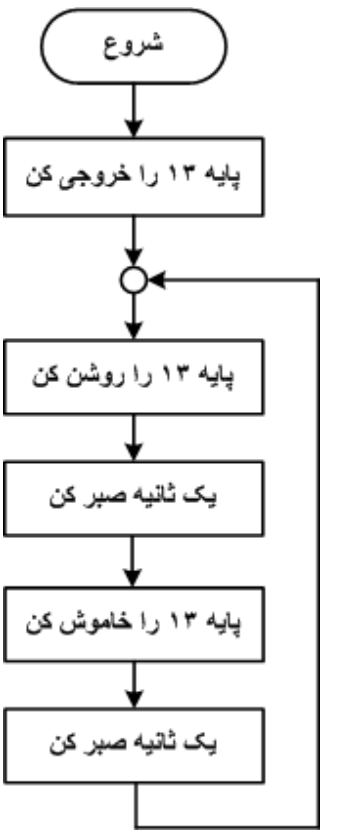
مثلا اگر بخواهیم مراحل اجرا شدن دستورات موجود درون توابع loop و setup را به صورت فلوچارت بیان کنیم، به صورت شکل زیر می‌شود.



شکل آ-۵: اجرا شدن دستورات داخل setup و loop

شکل فوق بیان می‌کند که پس از شروع ابتدا دستورات داخل setup اجرا می‌شوند. سپس دستورات داخل loop اجرا می‌شوند و بعد از اینکه دستورات داخل loop اجرا شدند، دوباره دستورات داخل loop اجرا می‌شوند و اجرای برنامه هیچ گاه به پایان نمی‌رسد.

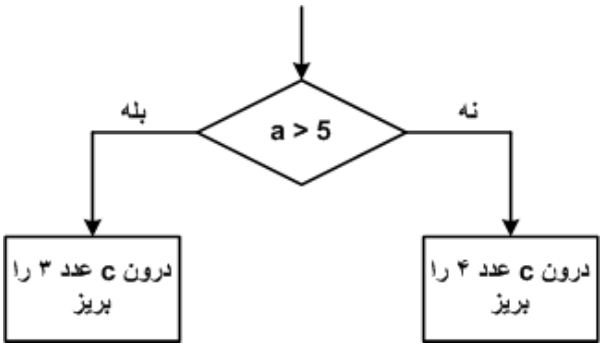
در شکل زیر فلوچارت مربوط به برنامه ای که هر یک ثانیه یکبار پایه ۱۳ را روشن و خاموش کند نمایش داده شده است.



شکل آ-۶: فلوچارت برای برنامه ای که هر یک ثانیه یکبار پایه ۱۳ چشمک بزند

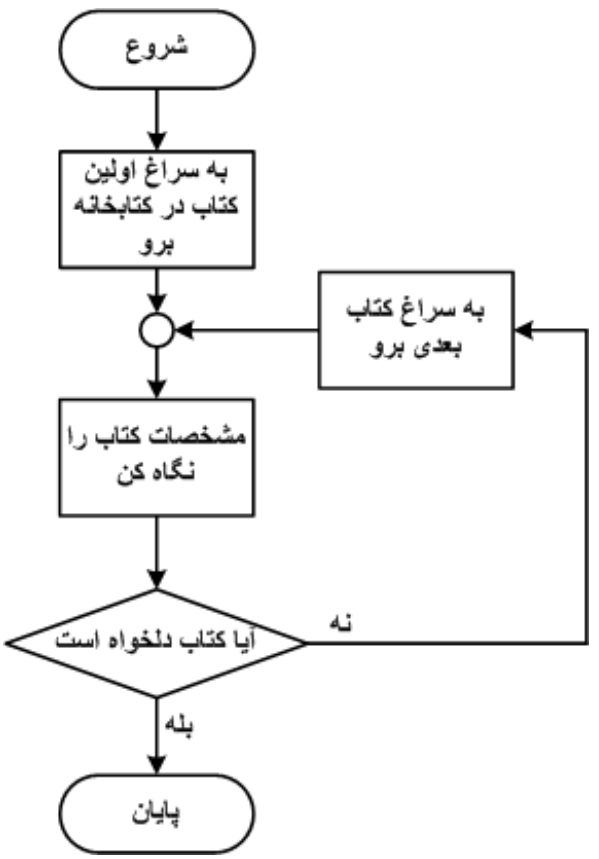
بخش الف-۲ : بیان کردن شرط در فلوچارت

در طی الگوریتمها گاهی شرطی را چک می‌کنیم و برحسب اینکه آیا شرط برقرار باشد یا اینکه برقرار نباشد، کارهای مختلفی را انجام می‌دهیم. برای نشان دادن شرط، لوزی می‌کشیم که از آن دو تا فلش خارج می‌شود: یکی برای حالتی که شرط درست است و دیگری برای حالتی که شرط غلط است. به طور مثال در شکل آ-۷، فلوچارتی کشیده شده است که اگر مقدار متغیر a بزرگتر از ۵ باشد، درون متغیر c، عدد ۳ را می‌ریزد و در غیر این صورت، درون متغیر c عدد ٤ را می‌ریزد.



شکل آ-۷: چک کردن اینکه آیا مقدار a بزرگتر از ۵ هست یا نه

اگر بخواهیم الگوریتم یافتن کتابی را در یک کتابخانه نامرتب به صورت فلوچارت بیان کنیم، به صورت شکل آ–۸ می‌شود.



شکل آ-۸: یافتن کتاب در کتابخانه مرتب نشده

ضمیمه ب: مختصر نویسی

برای اینکه برنامه ها ساده تر و قابل فهم تر باشد، در برنامه هایی که در این کتاب نوشته شده است از مختصرنویسی پرهیز شده است. پیشنهاد می‌کنم جهت جلوگیری از خطاهای سهوی این ضمیمه را در صورتی مطالعه کنید که بر عملیات ریاضی و دستورات شرط و حلقه کاملا مسلط شده باشید.

در زبان سی برخی از دستورات را می‌توان به شکل مختصرتری نوشت که در این فصل به این موضوع می‌پردازیم.

مختصر نویسی دستورات ریاضی

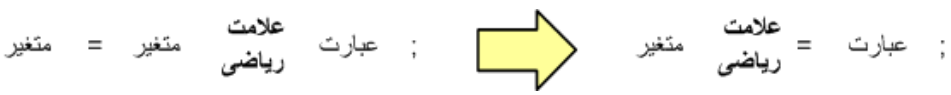
گاهی می‌خواهیم عملی ریاضی را بر روی متغیری انجام دهیم و حاصل را در همان متغیر بریزیم. مثلا می‌خواهیم مقدار متغیر k را ۲ تا زیاد کنیم و حاصل را در متغیر k بریزیم. برای این کار می‌توانیم دستور زیر را بنویسیم:

```
k = k + 2;
```

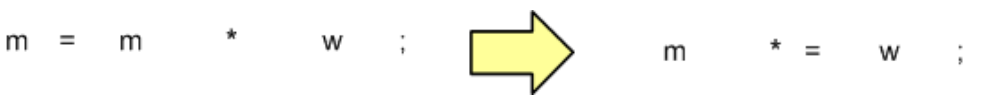
همین دستور را می‌توانیم به صورت فشرده شده زیر بنویسیم:

```
k += 2;
```

همچنین دستور i = i * 3; را می‌توانیم به صورت i *= 3; بنویسیم. در کل اگر قرار باشد، بر روی دو تا متغیر عملی ریاضی انجام دهیم و حاصل جواب را در اولین متغیر بریزیم، می‌توانیم دستور ریاضی را به صورت زیر مختصر کنیم:



به طور نمونه عبارت m = m * w; را می توانیم به صورت زیر مختصرنویسی کنیم:



در زیر نمونه‌های بیشتری از مختصر نویسی عملیات ریاضی را می‌بینید که در جلوی آنها شکل غیر مختصر آنها ذکر شده است:

```
i *= 5;      // i = i * 5;
q += b;      // q = q + b;
m /= 23;     // m = m / 23;
t %= 11;     // t = t % 11;
num -= 6;    // num = num - 6;
```

در برنامه نویسی خیلی اتفاق می‌افتد که بخواهیم مقدار متغیری را یکی زیاد کنیم یا یکی کم کنیم. عملگر ++ مقدار متغیر را یکی زیاد میکند و عملگر -- مقدار متغیر را یکی کم می‌کند. به طور نمونه دستور زیر مقدار متغیر a را یکی زیاد می‌کند:

```
a++;     //a = a + 1;
```

همچنین دستور زیر مقدار متغیر b را یکی کم می‌کند:

```
b--;     //b = b - 1;
```

در حلقه زیر، مقدار i را یکی یکی زیاد کرده ایم:

```
while(i < 5)
{
    i++;
}
```

مختصر نویسی در دستورات if و while و for

در فصل پنج دیدیم که ساختار کلی دستور if به صورت زیر است:

```
if(عبارت شرطی)
{
    مجموعه دستورات
}
```

در مواردی که بخواهیم اجرا شدن فقط یک دستور را منوط به شرط کنیم، می‌توانیم کروشه‌ها را حذف کنیم. مثلا، دستورات زیر چک می‌کند که اگر مقدار b کمتر از 5 بود، مقدار متغیر a را ۳ تا زیاد می‌کند:

```
if(b < 5)
    a += 3;
```

همچنین مجموعه دستورات زیر چک می‌کند که اگر a برابر با ۹ بود در متغیر c عدد ۵ را می‌ریزد و در غیر این صورت مقدار ۷ را درون c می‌ریزد:

```
if(a == 9)
    c = 5;
else
    c = 7;
```

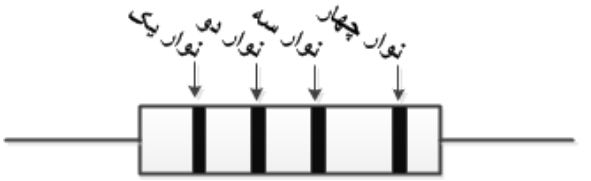
در دستورات for و while نیز اگر داخل حلقه فقط یک دستور باشد، می‌توانیم کروشه‌ها را حذف کنیم. مثلا مجموعه دستورات زیر حاصل جمع اعداد بین ٦ تا ١١ را حساب می‌کند:

```
int i;
int sum = 0;
for(i = 6; i <= 11; i++)
    sum += i;
```

ضمیمه پ: خواندن مقادیر مقاومتها

اندازه مقاومتهایی که در مدارها استفاده می‌کنیم معمولا کوچک هستند و اگر بخواهیم مقدار آنها را به صورت عددی به روی آنها بنویسیم نوشته‌ها بسیار ریز و ناخوانا می‌شود. بنابراین مقادیر مقاومتها را با استفاده از نوارهای رنگی که بر روی آن چاپ می‌کنند بیان می‌کنند. در زیر عکس یک مقاومت که بر روی آن چهار نوار رنگی رسم شده است را می بینید.

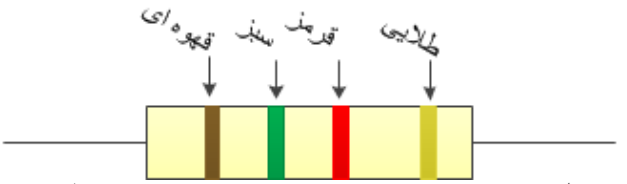
رنگ	عدد
سیاه	۰
قهوه‌ای	۱
قرمز	۲
نارنجی	۳
زرد	٤
سبز	٥
آبی	٦
بنفش	۷
خاکستری	۸
سفید	۹



شکل پ-۱: عکس یک مقاومت با چهار نوار رنگی

طریقه خواندن مقادیر مقاومتها به این صورت است که هر یک از رنگها دارای عدد مشخصی است که در جدول روبرو ذکر شده‌اند. هنگامی که می‌خواهید مقدار مقاومت را بخوانید به گونه‌ای آن را در دست می‌گیرید، که نواری طلایی رنگ در سمت راست قرار گیرد. سپس از چپ به راست به نوارها نگاه می‌کنیم و عدد مربوط به نوار اول و دوم را به ترتیب یادداشت می‌کنیم و سپس به تعداد عدد نوار سوم در جلوی عدد صفر می‌گذاریم.

مثلا در مقاومت زیر که اولین نوار قهوه‌ای (عدد ۱) و نوار دوم سبز (عدد ٥) و نوار سوم قرمز (عدد ۲) است، ابتدا عدد قهوه‌ای و سپس عدد سبز را می‌نویسیم که عدد ۱٥ تشکیل می‌شود و سپس با توجه به اینکه نوار سوم قرمز (۲) است، دو تا صفر به جلوی عدد ۱٥ می‌گذاریم که حاصل عدد ۱٥۰۰ می‌شود. پس مقدار این مقاومت ۱٥۰۰ اهم است.



شکل پ-۲: عکس یک مقاومت با نوارهای قهوه‌ای، سبز، قرمز و طلایی

برای اینکه مطمئن باشیم که اولین نوار را درست تشخیص می‌دهیم و آن را با نوار چهارم اشتباه نگیریم، معمولا نوار چهارم نسبت به بقیه نوارها فاصله بیشتری دارد. عکس مقاومت فوق را ببینید.