# برنامه نویسی کاربردی میکرو کنترلر


**مولف: دکتر مهدی سیاوش**

**استادیار دانشگاه فنی و حرفه ای**

# فهرست

# پیشگفتار

تدوین این کتاب با توجه به تجربه چندین سال تدریس و راهنمایی پروژه های دانشجویان در مقاطع کاردانی و کارشناسی بوده است و سعی بر آن است مخاطب گام به گام با میکروکنترلر آردوینو آشنا شود و تمام دستورات برنامه نویسی نیز آموزش داده شود. تلاش نویسنده در هنگام تالیف بر خودآموز بودن این کتاب بوده است که نه تنها یک منبع درسی برای میکروکنترلر آردوینو باشد، بلکه دانشجو و مخاطب بتواند به تنهایی با خواندن این کتاب تسلط کافی برای انجام پروژه‌های عملی همانند چندین نمونه که در انتهای کتاب آورده شده است، انجام دهد.

# راهنمای کتاب

این کتاب یک مرجع کامل تئوری و عملی در زمینه میکروکنترلر آردوینو می‌باشد که فصول آن طوری تنظیم شده است که با هم در ارتباط بوده و با مطالعه آنها به ترتیب، به فهم مطالب کمک به سزایی خواهد نمود. در فصل اول به معرفی آردوینو و توضیحات اجمالی در این زمینه پرداخته شده است. در فصل دوم نرم‌افزار آردوینو و نحوه نصب آن بیان شده و در فصل سوم محیط برنامه‌نویسی میکروکنترلر آردوینو معرفی شده است. در فصل چهارم به دستورات برنامه نویسی میکروکنترلر آردوینو پرداخته شده است و در فصل پنجم  نحوه اتصال ماژول‌های الکترونیکی به آردوینو بیان شده است. در فصل انتهایی نیز چند پروژه کاربردی با استفاده از آردوینو به صورت گام به گام معرفی شده است تا مخاطب کاربرد آموزش‌های فصول ابتدایی را به صورت عملی فرا گیرد.

# هدف کلی کتاب

از این کتاب به عنوان یک منبع درسی برای دانشجویان مقاطع کاردانی و کارشناسی مهندسی کنترل-مکاترونیک و الکترونیک می‌توان بهره برد. از سوی دیگر با توجه به کاربرد گسترده میکروکنترلر آردوینو در عمل برای تمام مهندسین و تکنسین که بدنبال یک مرجع برای برنامه نویسی و پیاده‌سازی یک میکروکنترلر به صرفه اقتصادی می‌باشند، مفید خواهد بود.

# ۱

# معرفی آردوینو

## ۱-۱- اهداف فصل

- آشنایی با آردوینو و تاریخچه آن
- شناخت انواع برد های آردوینو
- آشنایی با برد آردوینو UNO و پایه های آن
- شناخت انواع شیلد های آردوینو

## ۱-۲- مقدمه

بردهای آردوینو دارای تنوع فراوانی می‌باشند که با شناسایی انواع بردهای آردوینو و شیلدهای آن، می‌توان برای هر پروژه متناسب با نیاز، بهترین برد را انتخاب نمود.
آردوینو UNO، یکی از رایج‌ترین بردهای آردوینو بوده که در این فصل پایه‌های آن معرفی خواهند شد.

## ۱-۳- تاریخچه آردوینو

ایدهٔ ساخت آردوینو در سال ۲۰۰۳ میلادی در انستیتو طراحی تعاملی ایورئا در کشور ایتالیا شکل گرفت. ایده عبارت بود از ساخت وسیله‌ای ساده و کم هزینه برای انجام پروژه‌های دیجیتال دانشجویان، به‌خصوص آن‌هایی که آشنایی چندانی با اصول مهندسی و برنامه‌نویسی ندارند. سه فرد کلیدی در به ثمر نشاندن این ایده نقش داشتند:
هرناندو باراگان، ماسیمو بانزی، و کیسی ریس.
باراگان یکی از دانشجویان انستیتو ایورئا بود که تصمیم گرفت پایان‌نامهٔ کارشناسی‌ارشد خود را در این زمینه اجرا نماید. بانزی و ریس نیز استادان راهنمای پایان‌نامهٔ باراگان بودند. تا آن زمان هنوز اسمی از آردوینو در میان نبود. نتیجهٔ پایان‌نامهٔ باراگان بسیار موفقیت‌آمیز بود و منجر به ایجاد سخت‌افزار و نرم‌افزاری شد که وایرینگ نام گرفت. سخت‌افزار وایرینگ ویژگی‌های مورد نظر را نسبت به سایر نمونه‌های موجود در بازار آن زمان داشت یعنی ساده و کم‌هزینه بود. نرم‌افزار وایرینگ نیز بر مبنای یکی از زبان های برنامه نویسی موجود به نام پراسسینگ تهیه شده بود.
پس از اتمام پایان‌نامه، بانزی درصدد کاهش هزینه‌های سخت‌افزار وایرینگ برآمد و در سال ۲۰۰۵ میلادی با همکاری دیوید کوآرتلس و دیوید ملیس (که به ترتیب کارمند و دانشجوی انستیتو ایورئا بودند)، به توسعهٔ پروژهٔ وایرینگ پرداخت و نام آن را به آردوینو تغییر داد. این نام جدید برگرفته از نام کافه‌ای به نام آردوین در شهر ایورئا بود که اکثر جلسات گروه در آنجا تشکیل می‌شد. واژهٔ آردوین، نام یکی از شاهزادگان قدیم ایتالیا است که زمانی حکمران شهر ایورئا بود و در قرن یازدهم میلادی به پادشاهی ایتالیا رسید [۱].

## ۱-۴- معرفی آردوینو

آردوینو (Arduino) یک پلتفرم متن‌باز برای توسعه و ساخت دستگاه‌های الکترونیکی تعبیه شده است. این پلتفرم شامل یک میکروکنترلر و محیط توسعه نرم‌افزاری است که برای برنامه نویسی و کنترل اجزای الکترونیکی استفاده می‌شود. میکروکنترلر یک چیپ کامپیوتری کوچک است که شامل واحدهای پردازشی، ورودی و خروجی دیجیتال و آنالوگ و درگاه‌های ارتباطی مانندUART ، I2C و SPI است. آردوینو از میکروکنترلرهای مبتنی بر AVR از شرکت Atmel که در حال حاضر توسط میکروچیپ Microchip تصاحب شده است ،استفاده می‌کند. همچنین نسخه‌های آردوینو مبتنی بر میکروکنترلرهای دیگری مانند ARM Cortex-M نیز وجود دارند. متن باز یا OPEN-SOURCE را می‌توان به نوعی روش و راهی برای طراحی دانست که در آن سازنده‌ی یک سخت افزار و یا یک نرم افزار، این امکان را برای کاربران فراهم می کند که بتوانند آن نرم افزار و یا سخت افزار را به روش دلخواه خود تغییر بدهند. علاوه بر این، شما میتوانید از طریق IDE یا محیط توسعه اختصاصی که برای آردوینو طراحی شده است، به کتابخانه‌های متن باز و کدهای دیگران نیز دسترسی داشته باشید.

آردوینو در اصل یک سخت‌افزار، متشکل از یک میکروکنترلر مثل Atmega328 که یک برنامه خاص تحت عنوان bootloader در آن نصب شده است. این سخت‌افزار به گونه ای طراحی شده که شما میتوانید از طریق یک مبدل USB به TTL آن را پروگرام نمایید، بنابراین شما برای پروگرام کردن ماژول آردوینو، به پروگرامر جداگانه نیازی ندارید6t.

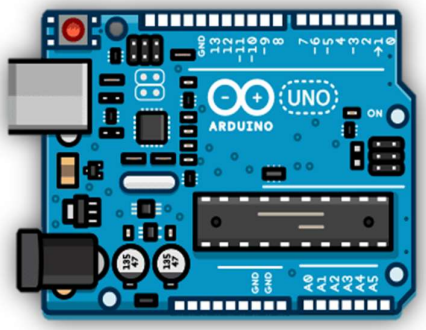
میکروکنترلر Atmega328 یک میکروکنترلر ۸ بیتی است که بسیار شبیه Atmega168 می‌باشد. با این تفاوت که حافظه فلش آن دوبرابر یعنی ۳۲ کیلوبایت فضا برای ذخیره کدها است، دارای ۳۲ خط ورودی و خروجی و سرعت پردازش آن با استفاده از یک کریستال خارجی حداکثر تا ۲۰ مگاهرتز می باشد، ولتاژ عملیاتی میکروکنترلر بین ۱،۸ تا ۵ ولت است.

محیط توسعه نرم‌افزاری آردوینو بر اساس زبان برنامه‌نویسی C/C++ است. برنامه‌های نوشته شده برای آردوینو به صورت کد منبع (سورس کد) به میکروکنترلر ارسال و در آن اجرا می‌شوند. این برنامه‌ها می‌توانند به سنسورها و اجزای الکترونیکی مختلف متصل شوند و وظایف مختلفی از جمله کنترل خروجی‌ها (مانند رله‌ها و موتورها)، اندازه‌گیری ورودی‌ها (مانند دما و رطوبت) و ارتباط با سایر دستگاه‌ها انجام دهند.آردوینو از طریق پین‌های ورودی و خروجی خود به عنوان رابطی بین دنیای فیزیکی و برنامه‌نویسی عمل می‌کند.

استانداردی که برای برنامه‌نویسی آردوینو استفاده می‌شود، کتابخانه‌های آردوینو است.

این کتابخانه‌ها حاوی توابع و کدهای آماده هستند که برای کنترل اجزای الکترونیکی مختلف مورد استفاده قرار می‌گیرند. برای مثال، با استفاده از کتابخانه‌های آردوینو، می‌توانید با سادگی

خروجی‌ها را روشن و خاموش کنید، ورودی‌ها را خوانده و عملیات‌های منطقی را بر روی آن‌ها انجام دهید و با اجزای الکترونیکی ارتباط برقرار کنید.




*شکل ۱-۱ نمای شماتیک آردوینو UNO [۱].*

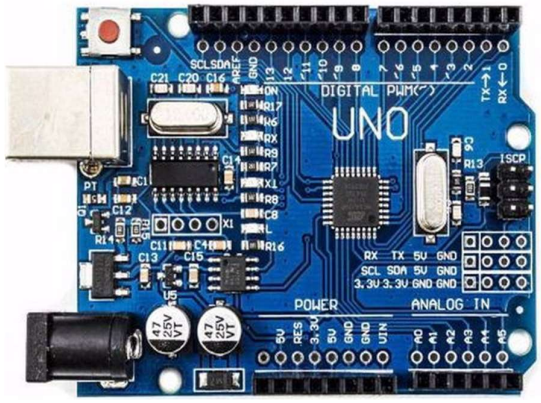
آردوینو به دلیل سادگی استفاده، وسعت امکانات و قابلیت همکاری با اجزای الکترونیکی مختلف، در بسیاری از پروژه‌های الکترونیکی و روباتیکی استفاده می‌شود. این شامل پروژه‌های آموزشی، پروژه‌های هوشمندسازی خانه، سیستم‌های اتوماسیون، روبات‌های هوشمند، دستگاه‌های پزشکی و بسیاری دیگر است.علاوه بر این، آردوینو به لطف جامعه فعال برنامه‌نویسان و سازندگان، منابع آموزشی فراوانی در دسترس دارد. وجود انجمن‌ها، وبسایت‌ها، فروم‌ها و ویدئوهای آموزشی این امکان را به مخاطب می‌دهد تا به راحتی با آردوینو آشنا شود و مهارت‌های برنامه‌نویسی و الکترونیکی خود را تقویت کند. در نهایت، آردوینو یک پلتفرم قدرتمند و انعطاف‌پذیر است که به هر سطح دانش و تجربه‌ای در زمینه الکترونیک و برنامه‌نویسی است.

## ۱-۵- انواع بردهای آردوینو

در این قسمت به معرفی انواع بردهای آردوینو پرداخته می‌شود[۵-۱].

### ۱) آردوینو Uno

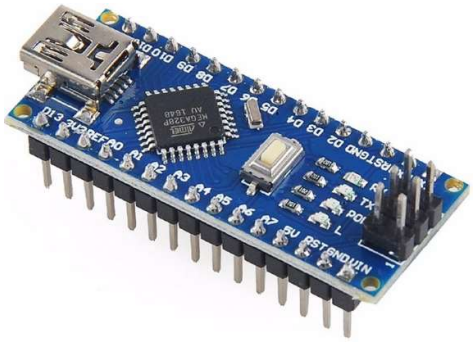
این برد آردوینو یکی از پرکاربردترین و پرطرفدارترین بردهای آردوینو است. دارای میکروکنترلر Atmega328P است و دارای ۱۴ پین دیجیتال ورودی/خروجی (که ۶ تا از آن‌ها را می‌توان به عنوان خروجی PWM استفاده کرد) و ۶ پین آنالوگ است. همچنین، دارای پورت USB برای برنامه‌ریزی و ارتباط با کامپیوتر است.




*شکل ۱-۲ تصویر مربوط به آردوینو UNO [۱].*

## ۲) آردوینو یونو

این برد آردوینو کوچکتر از Arduino Uno است که شامل ۱۴ پایه دیجیتالی، ۸ پایه آنالوگ، ۲ پایه تنظیم مجدد و ۶ پایه قدرت است و دارای میکروکنترلر Atmega328p یاAtmega168 است و برای پروژه‌هایی با محدودیت فضایی مناسب است. از لحاظ قابلیت‌ها و سخت‌افزار، شباهت زیادی به Arduino Uno دارد، با این تفاوت که دارای کانکتور میکرو USB و بعضی پین‌های اضافه است.



*شکل ۱-۳ آردوینو Nano [۱].*

### ۳) آردوینو مگا

این برد آردوینو برای پروژه هایی که به پورت های بیشتر و حافظه بزرگتری نیاز است، مناسب می باشد. دارای میکروکنترلر Atmega2560 است و دارای ۵۴ پین دیجیتال ورودی/خروجی (که ۱۵ تا از آن‌ها را می‌توان به عنوان خروجی PWM استفاده کرد) و ۱۶ پین آنالوگ است و می‌توان در ساخت انواع پروژه های پردازشی با فرکانس پایین، راه اندازی انواع سنسورها و ماژول ها استفاده کرد.


*(شکل ۱-۴) آردوینو مگا [۱].*

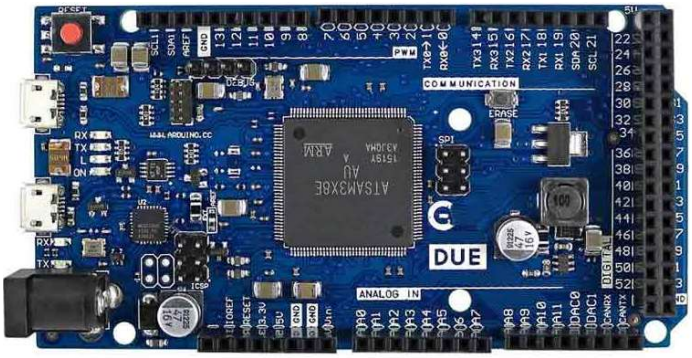
### ۴) آردوینو لئوناردو

آردوینو لئوناردو با سایر بردهای آردوینو متفاوت است زیرا دارای میکروکنترلر با ارتباط USB 2.0 داخلی است که قابلیت های USB را به خوبی پشتیبانی می‌کند. این نوع آردوینو دارای میکروکنترلر Atmega32U4 است و می‌تواند به عنوان یک دستگاه ورودی/خروجی HID(Human Interface Device) مانند یک صفحه کلید یا ماوس عمل کند.


*(شکل ۱-۵) آردوینو Leonardo [۱].*

### ۵) آردوینو Due

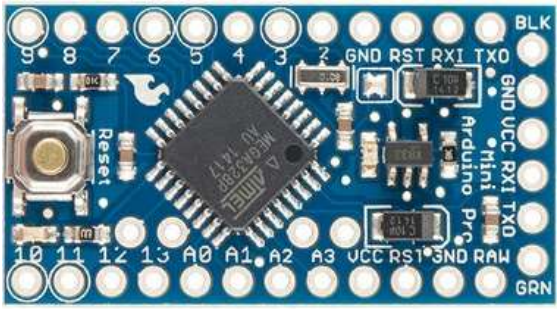
این برد آردوینو از معماری ARM Cortex-M3 استفاده می‌کند و دارای میکروکنترلر SAM3X8E است. اولین برد پایه ریزی شده برای کنترلرهای با معماری ۳۲ بیتی هست. برد آردوینو Arduino DUE با پردازنده ARM یک فیوز قابل ریست در خود گنجانده که کامپیوتر شما را از عبور جریانات ناخواسته بین برد و پورت سیستم شما محافظت کند.




*(شکل۱-۶) تصویر مربوط به آردوینو* Due *[۱].*

### ۶) آردوینو Pro Mini

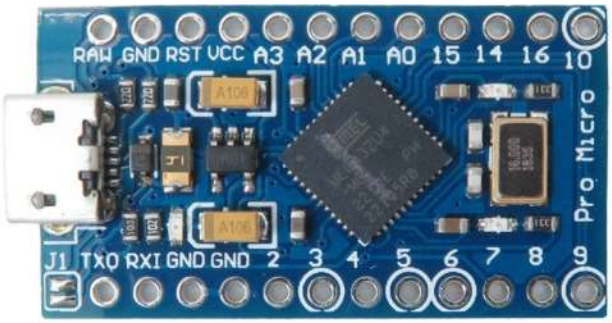
این برد آردوینو بسیار کوچک و قدرتمند است که قابلیت برنامه‌ریزی از طریق پورت سریال را دارد. اغلب برای پروژه‌های الکترونیکی حرفه‌ای که با فضای محدود یا پروژه‌هایی که نیاز به تمرکز بر روی کارایی و کمترین مصرف انرژی را دارند استفاده می‌شود.




*(شکل ۱-۷)برد pro mini [۱].*

### ۷) آردوینو Pro Macro

این برد، آردوینو ابعاد کوچکتری نسبت به Arduino Uno دارد و دارای میکروکنترلر Atmega32U4 است. با ابعاد ۴٫۸ × ۱٫۸ سانتی متر ، قابلیت اتصال به برد های بزرگتر آردوینو را دارا می‌باشد و ارتباط سریال USB، پین‌های دیجیتال و آنالوگ، PWM و I2C را پشتیبانی می کند. با این برد می‌توانید به عنوان یک دستگاه HID مانند صفحه کلید یا ماوس عمل کرد.

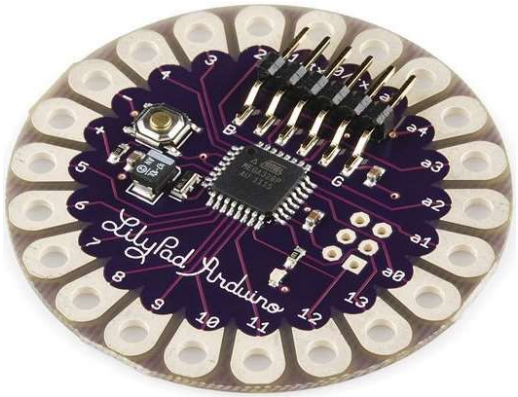



*(شکل ۱-۸) آردوینو Pro Micro [۱].*

### ۸) آردوینو LilyPad

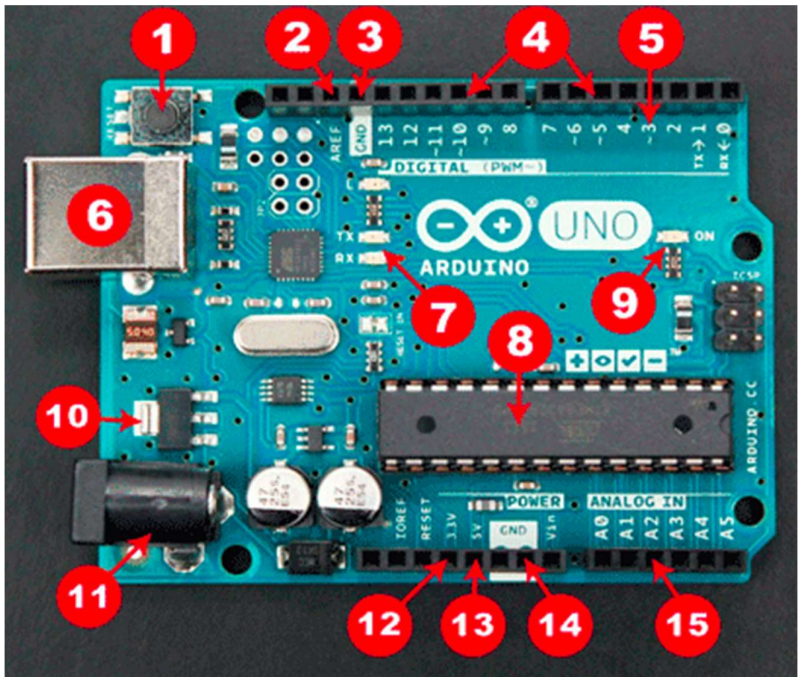
آردوینو LilyPad برای کاربردهای پوشیدنی و لباس‌های هوشمند طراحی شده است. این آردوینو به منظور دوخته شدن به محصول طراحی شده است و با استفاده از نخ رسانا می‌توان آن را به سایر اجزای قابل دوخت متصل کرد. بر روی Lilypad Main Board میکروکنترلر Atmega168 و یا Atmega328 نصب شده است که دارای 32KB حافظه فلش و 2KB حافظه SRAM بوده و قادر است برنامه‌ها را با سرعت 8MHz اجرا کند.

این نسخه از Lilypad دارای ۲۰ پین دیجیتال است که ۶ تای آن PWM را پشتیبانی می‌کند و همچنین ۶ پین آنالوگ دارد. جالب است که تمام این موارد بر روی یک دایره به قطر ۵ سانتی‌متر گنجانده شده است.




(شکل ۱-۹) LilyPad [۱].

## ۱-۶- معرفی برد آردوینو UNO




(شکل ۱-۱۰) آردینو Uno [۲].

۱- کلید ریست:

همانطور که در تصویر مشاهده می‌شود، آردوینو یک کلید فشاری دارد که از آن برای ریست کردن استفاده می شود. نحوه کار این کلید به این صورت است که با فشردن آن به طور موقت پین RESET میکروکنترلر به GND وصل شده و در نتیجه کدهای آردوینو مجددا اجرا می شوند.

۲- AREF:

هنگامی که از واحد ADC استفاده می‌شود، باید یک مرجع ولتاژ برای انجام محاسبات مربوط به مبدل آنالوگ به دیجیتال به میکرو معرفی شود، AREF نیز یکی از این مراجع است. از این پین برای اعمال یک ولتاژ مرجع بین صفر تا پنج ولت استفاده می شود.

۳- پین‌های GND:

همانطور که در تصویر مشاهده می‌شود ، به علت استفاده مکرر GND در مدارات الکترونیکی، معمولا چند پین GND بر روی برد آردوینو وجود دارد.

۴- ورودی و خروجی Digital Input/Output:

اگر با دقت به برد آردوینو توجه شود، مشاهده می‌شود که پین های ۱ تا ۱۳، مختص پایه‌های دیجیتالی هستند که درست در مقابل پین‌های آنالوگ قرار گرفته‌اند. برای خواندن حالت یک کلید فشاری یا روشن و خاموش کردن یک LED باید از این پین ها استفاده می‌شود.

۵- پین های PWM:

در کنار پین های دیجیتال (۳،۵،۶،۹،۱۰ و۱۱) علامتی به شکل ~ قرار گرفته است. معمولا هر پایه میکروکنترلر از چند وظیفه پشتیبانی می‌کند و این علامت نیز نشان دهنده وظیفه دوم این پایه‌های دیجیتال است. این پین ها علاوه بر استفاده به عنوان پین‌های دیجیتال معمولی، می تواند به عنوان PWM نیز مورد استفاده قرار گیرند.
از پین‌های PWM معمولا برای تغییر مدولاسیون پهنای پالس که منجر به تغییر شدت نور ، دور موتور و ... می‌شود، استفاده می‌کنند.

۶- پورت USB:

این پورت در تمام مدل‌های آردوینو، برای برنامه‌ریزی کردن و تغذیه برد استفاده می شود.

۷- LED های RX و TX:

برای برقراری ارتباط سریال از پین های RX و TX استفاده می‌شود. در همین راستا طراحان آردوینو، برای نمایش ارسال و دریافت دیتا بین کامپیوتر و... از این LED ها استفاده کرده اند.

TX: ارسال دیتا RX: دریافت دیتا

۸- آی سی اصلی برد آردوینو:

آردوینو در اصل یک سخت‌افزار متشکل از یک میکروکنترلر مثل Atmega328 که یک برنامه خاص تحت عنوان bootloader در آن نصب شده است. این میکروکنترلر در اصل مغز برد آردوینو است که تمام محاسبات، پردازش ها و ... توسط آن برنامه ریزی و مدیریت می‌شود. میکروکنترلر ماژول آردوینو معمولا عضوی از خانواده Atmega می باشد.

از آی سی Atmega328 بیشتر در Arduino Uno استفاده می‌شود. علت استفاده از آی سی Atmega328 در آردوینو ، پایین بودن مصرف جریان و سرعت بالای آن است.

۹- LED نشانگر تغذیه:

برد آردوینو علاوه بر منبع تغذیه خارجی ۵ ولت با USB کامپیوتر نیز فعال می‌شود. در طراحی برد آردوینو، یک LED ریز به منظور نمایش اعمال ولتاژ تغذیه به برد تعبیه شده است تا وقتی برد آردوینو به USB یا آداپتور وصل می شود، به کاربر اعلام کند که مشکلی در تغذیه وجود ندارد. اگر LED روشن نشود قطعا مشکلی در تغذیه برد وجود دارد که باید اصلاح شود.

۱۰- آی سی رگولاتور ولتاژ:

برای جلوگیری از ورود ولتاژهای بیشتر از ۵ ولت به مدار، از این رگولاتور استفاده می‌شود تا به کمک آن با محدود کردن ولتاژ اعمالی به مدار از آسیب دیدن آن جلوگیری کنند.

۱۱- DC Power Barrel Jack:

این سوکت آداپتور ، برای جلوگیری از آسیب دیدن پورت های لپ تاپ در هنگام تست مدار، اتصال کوتاه و ... بر روی برد تعبیه شده است. تا بدون استفاده از ورودی USB برد خود را روشن نمایید.

۱۲- پین ولتاژ ۳,۳ ولت:

قطعا در هنگام کار با ماژول های مختلف، به اهمیت دسترسی به یک ولتاژ ۳,۳ ولت پی برده اید. در نتیجه طراحان این برد خارق العاده یک پین ۳,۳ ولت نیز بر روی آن تعبیه کرده اند.

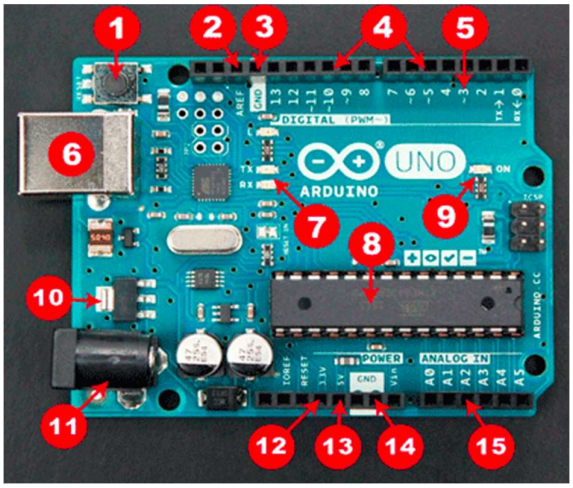
۱۳- پین ولتاژ ۵ ولت:

با اتصال این پین به برد بورد برای تامین تغذیه مدار استفاده می‌شود. بیشتر قطعات و ماژول‌های مورد استفاده با آردوینو با یکی از این دو ولتاژ تغذیه می شوند.

۱۴- پین های GND

۱۵- پین های ورودی آنالوگ:

با اتصال این پین به برد بورد برای تامین تغذیه مدار استفاده می‌شود. بیشتر قطعات و ماژول‌های مورد استفاده با آردوینو با یکی از این دو ولتاژ تغذیه می شوند.



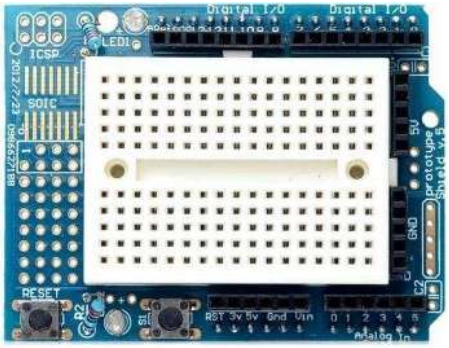

*(شکل ۱-۱۰) آردینو Uno [۲].*

## ۱-۷- شیلدهای آردوینو

شیلدهای آردوینو بردهایی هستند که عملکرد برد آردوینو را گسترش می‌دهند. برای انجام این کار باید آن‌ها را بالای برد آردوینو قرار داد. این شیلدها انواع مختلفی دارند که در این جا به معرفی تعدادی از آن ها پرداخته می‌شود:

۱. شیلد برد بوردProto Shield :

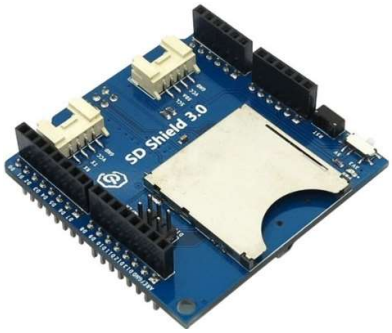
Proto Shield یک شیلد برد بورد است که به آسانی اتصال بین برد بورد و آردوینو را فراهم می‌کند و امکان اتصالات و رابط‌های متنوعی را ارائه می‌دهد تا بتوان قطعات الکترونیکی مختلف را به برد آردوینو UNO متصل کند. این شیلد شامل اتصالات دیجیتال، آنالوگ و منطقی(GPIO) ، رابط‌های سریال(Serial) ، رله‌ها، سوئیچ‌ها، کلیدها و سایر قطعات است.




*(شکل ۱-۹) شیلد برد بورد [۲].*

۲. شیلد micro SD :

این شیلد، آردوینو را به قابلیت ذخیره سازی انبوه تجهیز میکند که می‌توان از آن برای ثبت اطلاعات یا دیگر پروژه های مرتبط استفاده کرد.




*(شکل ۱-۱۰) تصویری از شیلد SD [۲].*

۳. شیلد دوربین :

با این شیلد می‌توان پیچیدگی رابط کنترل دوربین را کاهش داد. این شیلد از رابط I2C برای تنظیم حسگر و رابط SPI برای دستورات دوربین و جریان داده استفاده می‌کند.

*(شکل ۱-۱۱) تصویری از شیلد دوربین [۲].*

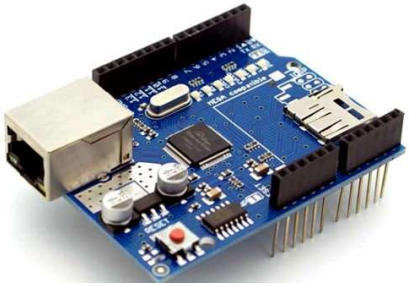
۴. شیلد موتور :

این شیلد امکان کنترل هر چه راحت تر موتور و سرعت آن را فراهم میکند. عکس (۱-۱۲) شیلد درایور موتور L293D است که دارای ۲ رابط موتور DC و ۴ رابط استپرموتور می‌باشد. هر کانال شیلد درایور موتور L293D تا ۶۰۰ میلی آمپر جریان دهی داشته و به مدار H-bridges نیز معروف می‌باشد. L293D یکی از بهترین راه ها برای کنترل موتورهای DC، Servo و Stepper است. درایور نسبتا رایج دیگر درایور موتور L298N است اما بر خلاف درایور L293D، موتورهای DC را کنترل می کند.

*(شکل ۱-۱۲) تصویری از شیلد موتورl293 [۲].*

۵. شیلد اترنت :

Ethernet Shield شیلدی است که به کمک آن میتوان آردوینو را به اینترنت متصل و سپس آن را به شبکه خود متصل کنید.



*(شکل ۱-۱۳) شیلد اترنت [۲].*

۶. شیلد Lcd :

این شیلد استفاده از ال سی دی ۲×۱۶ کاراکتری را آسان می‌کند. با این کار، تنها با استفاده از دو پایه ارتباط I2C موجود در آردوینو، می توان یک LCD کاراکتری ۲×۱۶ را کنترل کرد.

*(شکل ۱-۱۴) شیلد Lcd [۲].*

## ۱-۸- خود آزمایی

۱. متن باز بودن برد آردوینو به چه معناست؟
۲. چند مورد از انواع آردوینوها را نام برده و یکی را توضیح دهید.
۳. کاربرد پایه های RX و TX را شرح دهید.
۴. شیلد L293 را توضیح دهید.

# ۲

# نرم افزار آردوینو

## ۲-۱- اهداف این فصل

- نحوه دانلود و نصب برنامه
- آشنایی کامل با محیط برنامه
- شناخت منو و Toolbar برنامه

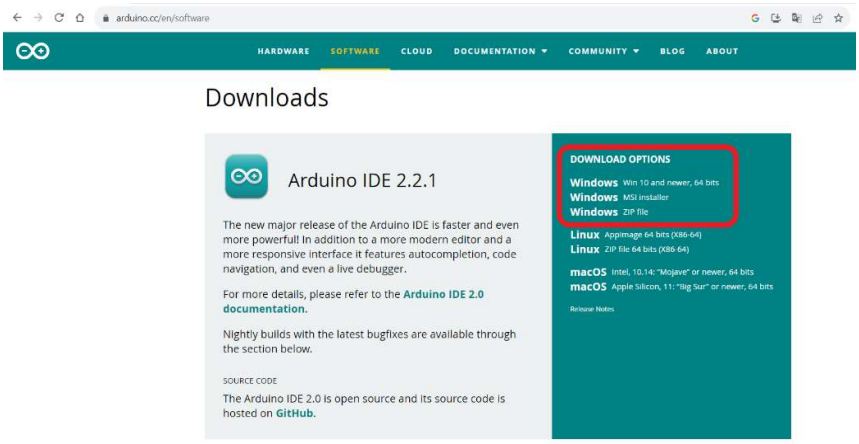
## ۲-۲- مقدمه

در این فصل، به دانلود و نصب برنامه گام به گام پرداخته شده است و سپس تمام گزینه‌های برنامه معرفی خواهند شد.

## ۲-۳- نصب نرم افزار Arduino IDE بر روی ویندوز

گام اول:

ابتدا باید آخرین نسخهٔ نرم افزار را از سایت رسمی آردوینو دانلود کرد. برای اینکار به سایت اصلی Arduino.cc رفته و از نوار ابزار بالا گزینه Software را انتخاب کرده و از صفحه باز شده، گزینه مربوط به ویندوز را انتخاب نمود.

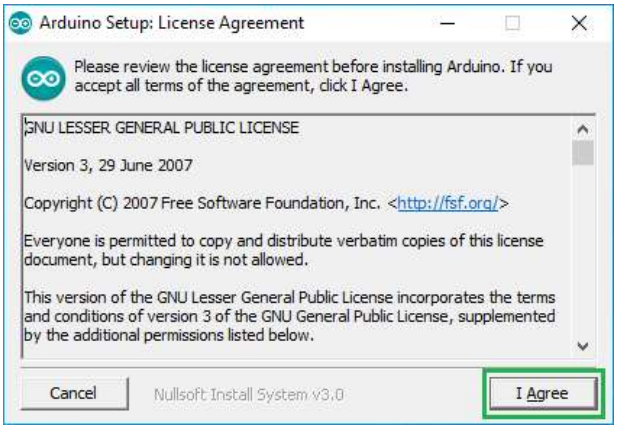



*(شکل ۲-۱) سایت آردینو [۱].*

گام دوم :

فایل دانلود شده را باز کرده تا عملیات نصب نرم افزار شروع شود. با انتخاب گزینه I Agree ، قوانین نرم افزار پذیرفته می‌شود.

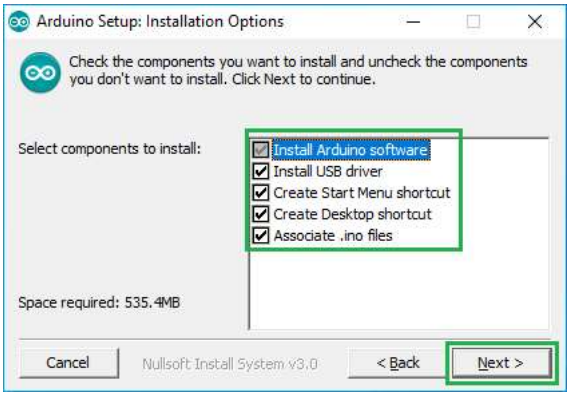

*(شکل ۲-۲) آیکون دانلود شده آردوینو*




*(شکل ۲-۳) پنجره پذیرش قوانین*

گام سوم :

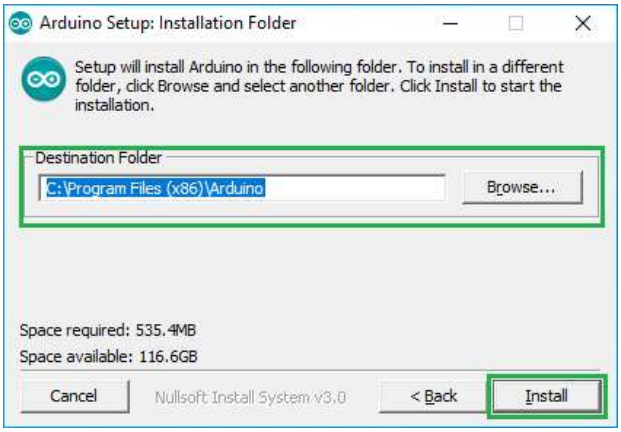
بخش های مختلفی از نرم افزار که کاربر به آن ها نیاز دارد تا بر روی سیستم نصب شوند را می‌توان انتخاب کرد. بهتر است تمامی گزینه ها را فعال کرد تا همه‌ی بخش‌های نرم افزار Arduino IDE نصب شود.




*(شکل ۲-۴) پنجره انتخاب نیازمندی ها برای نصب*

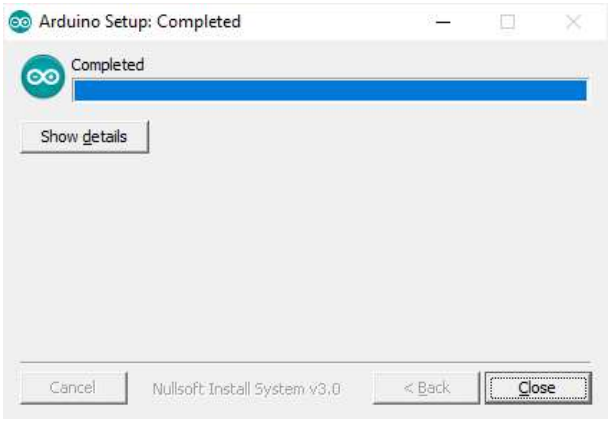
گام چهارم :

در این گام محل نصب نرم افزار را مشخص می‌شود. اگر درایو C (درایو پیش فرضی که خود نرم افزار پیشنهاد داده است) به اندازه کافی فضای خالی دارد، پیشنهاد می‌شود که این آدرس را تغییر داده نشود.




*(شکل ۲-۵) تصویر انتخاب پوشه برای نصب*

سپس باید منتظر ماند تا نصب نرم افزار کامل شود.




*(شکل ۲-۶) آماده سازی پایانی برای نصب*

حال نصب Arduino IDE به اتمام رسید.

## ۲-۴- معرفی محیط نرم افزار آردوینو

نرم‌افزار آردوینو، به نام Arduino IDE مخفف Integrated Development Environment شناخته می‌شود. این محیط توسعه نرم‌افزاری برای برنامه‌نویسی بردهای

آردوینو مورد استفاده قرار می‌گیرد. Arduino IDE دارای یک ویرایشگر کد ساده است که به امکان می‌دهد کدهای برنامه‌نویسی را برای بردهای آردوینو نوشته و ویرایش شود.


*(شکل ۲-۷) محیط کلی برنامه آردوینو*

Verify / Upload: به وسیله این دستور می توان کد را کامپایل کرده و در برد آردوینو خود آپلود نمود.

Select Board & Port: تمامی برد های آردوینو که توسط سیستم شناسایی شده در این قسمت نمایش داده می شود و می توان برد آردوینو مد نظر را از این بخش انتخاب کرد.

Sketchbook: جهت ذخیره سازی برنامه های نوشته شده (اسکچ ها) می باشد و برنامه‌های نوشته شده را می توان از این قسمت باز کرد.

Boards Manager: در این قسمت می توانید پکیج های Third party مورد نیاز پروژه را نصب کرد.

Library Manager: با استفاده از این گزینه می‌توان هزاران کتابخانه آردوینو را که توسط آردوینو و افراد مختلف در سراسر جهان ساخته شده است را نصب نمود.

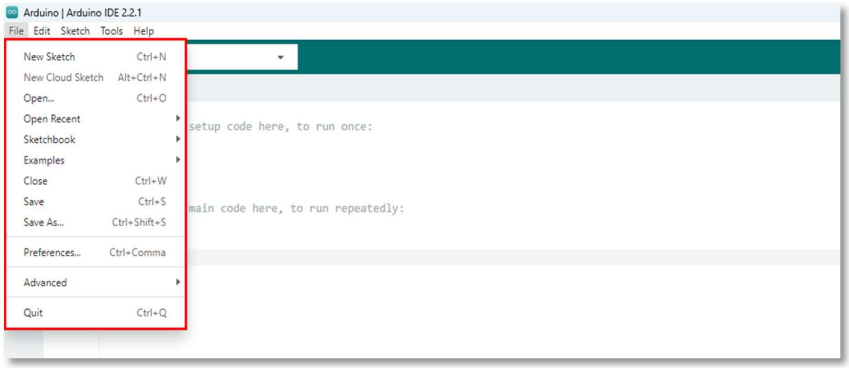
Debugger: دیباگر یک ابزار نرم افزار است که برای آزمایش و اشکال زدایی برنامه‌ها استفاده می شود.

Search: کلمات کلیدی مورد نظر را در کد نوشته شده جست و جو می کند.

Open Serial Monitor: سریال مانیتور را به عنوان یک تب جدید در کنسول باز می کند و داده های سریالی که از برد آردوینو دریافت می‌شود را نمایش می دهد. جهت ارسال داده به برد،

می‌توان را متن را وارد کرد و baud rate را که برابر با پارامتر ورودی Serial.begin در اسکچ است را از لیست کشویی انتخاب کرد.

منوی File




*(شکل ۲-۸) منوی file*

New Sketch :یک اسکچ (پروژه) جدید ایجاد می‌کند.

Open... : هرستی از همه اسکچ (پروژه) های موجود در حافظه را نمایش می‌دهد . با کلیک کردن روی یکی از آنها، درون پنجره جاری باز می شود.

Examples: مثال هایی از پروژه های مختلفی که با آردوینو کار شده است و کد آن خط به خط توضیح داده شده است که بیشتر جنبه آموزشی دارد ؛ همچنین می‌توان از این کدهای آماده در بخش های مختلف پروژه استفاده کرد.

Close : برای بستن پروژه یا برنامه مورد استفاده قرار می‌گیرد.

Save : اسکچ (پروژه) را ذخیره می کند.

Save As. : اسکچ (پروژه) را در جایی که در همان لحظه مشخص می‌شود را ذخیره می‌کند.

Preferences. :تغییر تنظیمات برنامه مانند : زبان برنامه ، رنگ زمینه و ...

Sketchbook : محیط برنامه نویسی آردوینو از مفهوم Sketchbook استفاده می کند که محلی استاندارد جهت ذخیره سازی برنامه ها می‌باشد. اسکچ های درون sketchbook را می توان از منوی File > Sketchbook یا از دکمه Open در نوار ابزار باز شوند. اولین بار که نرم افزار آردوینو را اجرا می کنید، به طور خود کار پوشه ای برای sketchbook ایجاد خواهد کرد. می توان مکان sketchbook را از طریق کادر مکالمه Preferences مشاهده و یا تغییر داد.

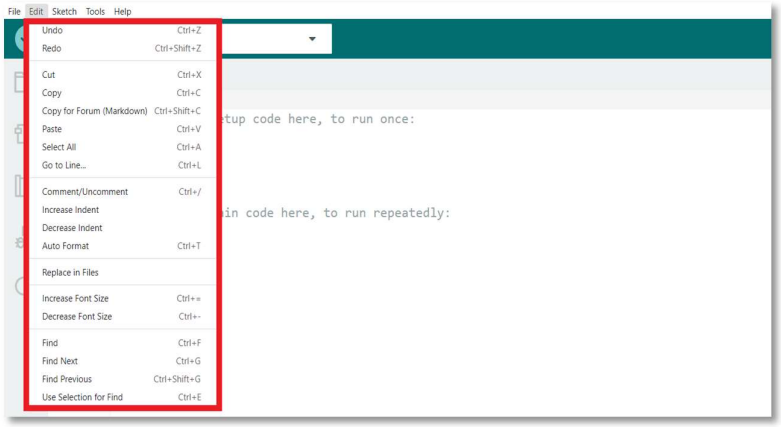
کاربر زمانی که با نسخه ۱,۰ کار می کند، فایل ها با پسوند ino. ذخیره می شوند. نسخه های قبلی از پسوند pde. استفاده می کنند.

با این حال هنوز هم می توان فایلهایی که در نسخه ۱،۰ و قبلتر از آن با پسوند pde. ذخیره شده اند را باز کرد و نرم افزار به طور خودکار پسوند آنهارا به ino. تغییر می دهد.

Advanced => Keyboard Shortcuts در این بخش می‌توان کلید های میانبر را مشاهده کرد که جهت راحت تر و سریع تر شدن عملیات برنامه نویسی مورد استفاده قرار می‌گیرد.

Quit : جهت خروج (بسته شدن) از برنامه میتوان از این گزینه استفاده نمود .

منوی Edit

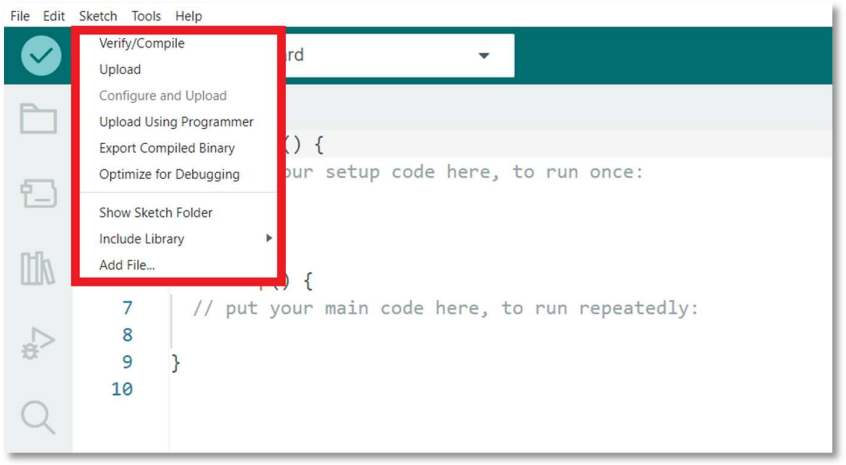



*(شکل ۲-۹) منوی Edit*

گزینه‌هایی که در این منو وجود دارد، بیشتر برای تنظیمات ظاهری و ابزارهای کمکی، برای هر چه راحت تر کد نویسی می‌باشند.

گزینه هایی همچون Undo , Redo , Cut ,Copy , Paste, Go to line... و ... که با کلید میانبر هم در دسترس میباشند .

## ۲-۵- منوی Sketch




*(شکل ۲-۱۰) منوی Sketch*

Verify / Compile: پس از اجرای یکی از این گزینه‌ها و در صورتی که کد بدون مشکلات کامپایل شود، می‌توانید از گزینه (Upload) برای آپلود کد در برد (Arduino) استفاده کرده و برنامه را روی برد خود اجرا کنید.

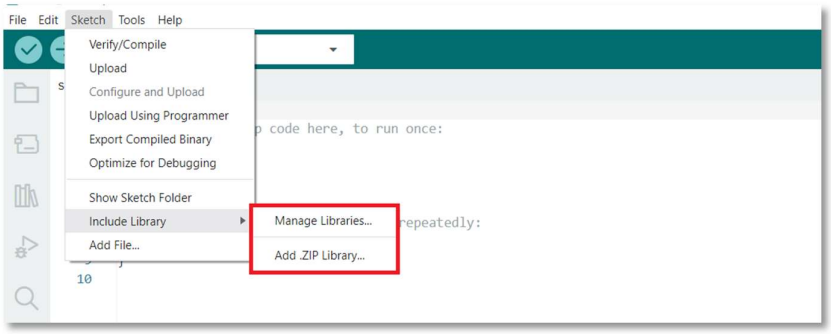
Upload: با انتخاب این گزینه، می‌توانید اقدام به کامپایل کردن کد کرده و بردهایArduino خود را برای اجرا آماده کنید.

Export Compiled Binary: این گزینه این امکان را به شما می‌دهد که فایل کامپایل شده (باینری) را در مسیری دلخواه ذخیره کنید.

Show Sketch Folder: این گزینه این قابلیت را می‌دهد که پوشه حاوی فایل‌های پروژه را در سیستم خود باز کنید.

Add File: با استفاده از این گزینه، می‌توانید فایل‌های جدید را به پروژه خود اضافه کرده، مثل تصاویر و فایل‌های دیگری که نیاز دارید.

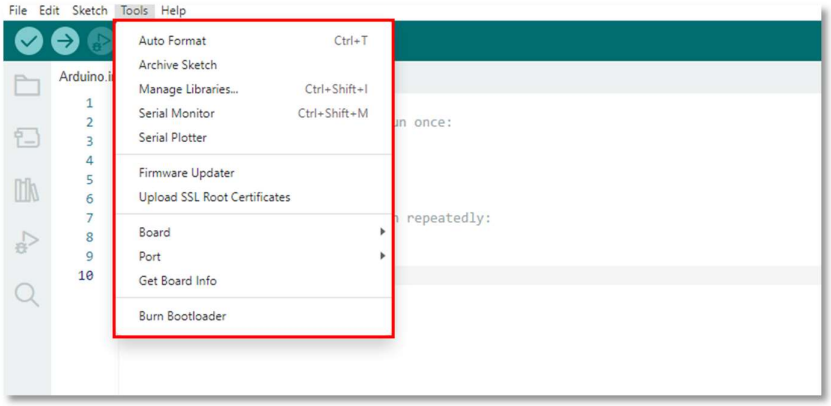
Include Library: این زیرمنو به شما این اجازه را می‌دهد که کتابخانه‌های مورد نیاز پروژه را به کد اضافه کنید. می‌توانید از کتابخانه‌های استاندارد(Arduino) و یا کتابخانه‌های شخصی نیز استفاده کنید.




*(شکل ۲-۱۱) زیر مجموعه گزینه Include Library*

Manage Libraries: این گزینه به شما این قابلیت را می‌دهد که کتابخانه های موجود را مشاهده، به روزرسانی و یا نصب کنید. با انتخاب این گزینه، پنجره‌ای باز می‌شود که در آن می‌توان کتابخانه ها را جستجو کرد.

Add .ZIP Library: این گزینه این قابلیت را می‌دهد که کتابخانه های شخصی و یا سفارشی که به صورت فایل فشرده ZIP در اختیار دارید را به پروژه‌های خود اضافه کنید.




*(شکل ۲-۱۲) منوی Tools*

Auto Format: این گزینه کد شما را قالب بندی می کند برای مثال آکولادهای باز شده و بسته شده را به خط می کند.

Archive Sketch: یک کپی از اسکچ کنونی را در قالبZip بایگانی می کند. فایل بایگانی در همان مسیری قرار می گیرد که اسکچ در آن وجود دارد.

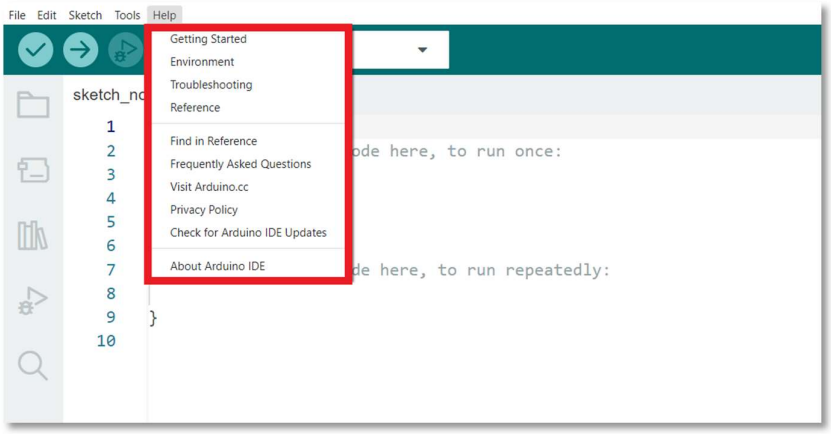
Board: با استفاده از این گزینه می توان بردی که قصد استفاده از آن را دارید انتخاب کرد.

Serial Port: این منو شامل همه وسایل Serial روی سیستم می‌باشد. این منو هر بار که منوی Toolsرا باز می‌شود را به صورت خودکار به روزرسانی می‌کند.

Programmer: این منو جهت انتخاب یک پروگرمر سخت افزاری در زمان پروگرم کردن یک تراشه جدید بدون استفاده از اتصال USB-serial موجود بر روی برد استفاده می شود.

Burn Bootloader: این منو به شما این امکان را می دهد تا یک Bootloader را روی یک میکروکنترلر بر روی برد آردوینو بارگذاری کنید. این مورد جهت استفاده عادی از یک برد آردوینو لازم نیست اما اگر یک میکروکنترلر ATmega جدید خریداری کنید که به صورت پیش فرض بدون یک Bootloader باشد این گزینه می تواند مفید باشد. توجه کنید که پیش از بارگذاری Bootloader برد صحیح را از منوی Boards انتخاب کرده باشید.

منوی Help




*(شکل ۲-۱۳) منوی Help*

هر آنچه که درباره آردوینو، تنظیمات آن ، منابع، به روز رسانی نرم افزار و هر کمکی که درباره نرم افزار و کدنویسی پروژه کاربر نیاز داشته باشد را می توان از گزینه های این منو استفاده نمود.

## ۲-۶- خود آزمایی

۱. IDE مخفف چیست؟
۲. Verify / Compile را توضیح دهید.
۳. چگونگی اضافه کردن کتابخانه ها را شرح دهید.

# ۳

# برنامه نویسی آردوینو

## ۳-۱- اهداف فصل

- شناخت زبان آردوینو
- آشنایی با دستورات ابتدایی و اصلی

## ۳-۲- مقدمه

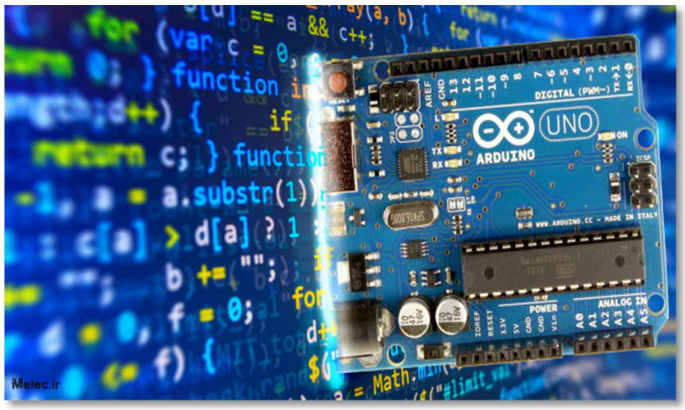
فهم درست از زبان برنامه نویسی آن تراشه الکترونیکی، به بهبود عملکرد و بهینه سازی کد کمک خواهد کرد. به همین علت در این فصل به آشنایی با زبان آردوینو و محیط اصلی برنامه نویسی آن پرداخته شده است.

## ۳-۳- زبان برنامه نویسی آردوینو

زبان برنامه نویسی آردوینو C++ است. زبان برنامه نویسی C++ از زبان برنامه نویسی C توسعه پیدا کرده و از سبک برنامه نویسی شی گرا پیروی می‌کند. زبان برنامه نویسی شی گرا یک شیوه ی نوین برای کدنویسی است که از ترکیب قطعات نرم افزاری ساخته شده است. هدف زبان برنامه نویسی شی گرا مثل C++ این است که اشیای دنیای واقعی را مدلسازی و نمونه سازی می‌کند. زبان برنامه نویسی C++ انعطاف پذیر است و محدودیتی برای آردوینو ایجاد نمی‌کند. مهم ترین نکته این است که این زبان برنامه نویسی نسبت به حروف حساس است و بین حروف کوچک و بزرگ تفاوت است و تمامی کلمات کلیدی با حروف کوچک نوشته می‌شود.

دستوالعمل های برنامه C++ دارای چند پارامتر مهم است :

- هر دستور زبان برنامه نویسی به سمیکالن ( ; ) منتهی می‌شود.
- حداکثر طول هر خط برنامه ۲۵۵ کاراکتر است.
- برای مشخص شدن هر خط از برنامه از کامنت ( // ) استفاده می‌شود تا پس از زیاد شدن تعداد خطهای برنامه نویسی امکان رفع عیب سریع تر باشد.




*(شکل ۳-۱) تصویری از برد UNO [۶].*

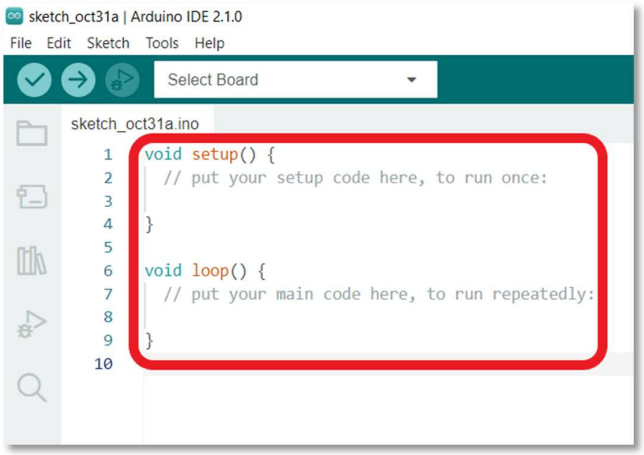
## ۳-۴- محیط برنامه نویسی نرم افزار

برنامه آردوینو به سه بخش اصلی تقسیم می شود:

۱- Structure

دو بخش اصلی در این قسمت نهفته است، که کاربر با هر بار ورود به نرم افزار آردوینو و ایجاد یک تب جدید برای شروع برنامه نویسی با آن رو به رو می‌شود.

Setup () Function
Loop () Function




*(شکل ۳-۲) محیط اصلی برای برنامه نویسی*

Void setup: در این قسمت، توابع اصلی برای شروع برنامه نویسی تعریف می‌شود. ازاین تابع برای معرفی متغیرها، حالت های پایه، فراخوانی دستورات کتابخانه ای استفاده می‌شود. توابعی تعریف شده در setup ، فقط یک بار اجرا می‌شود.

از بخش های مهم دیگری که در Void setup تعریف می‌شود، ورودی input و خروجی output است. در نهایت در این بخش از Return هم استفاده می‌شود. برنامه نوشته شده در این بخش با باز و بسته شدن سریال مانیتور و یا ریست توسط کلید قرار گرفته شده با آردوینو، دوباره از اول اجرا می‌شود.

Void loop: هنگامی که دستورات نوشته شده در void setup اجرا شد، نوبت به این تابع می‌رسد؛ توابع حلقه در برنامه تا زمانی که حلقه در برنامه تعریف شده باشد، تمامی دستورات نوشته شده اجرا می‌شود. تنها با قطع ارتباط و یا ریست برنامه متوقف خواهد شد. دستورات در این تابع از بالا به پایین به صورت پیوسته اجرا می‌شود. تابع loop یکی از اساسی ترین بخش

های برنامه نویسی در آردوینو است و بدون آن در عملکرد میکروکنترلر تداخل ایجاد می‌شود. حتی اگر در این قسمت برنامه ای هم نوشته نشده باشد، به آن نیاز خواهد بود.

## ۳-۵- خود آزمایی

۱. دستور عمل نوشتن کد در آردوینو را توضیح دهید.

۲. محیط برنامه آردوینو به طور کلی چند نوع است؟ به اختصار توضیح دهید.

# ۴

# دستورات برنامه نویسی

## ۴-۱- اهداف این فصل

- یادگیری مفهوم ساختار های کنترلی
- یادگیری دستورات شرطی if, if-else و از این قبیل
- فهم درست از حلقه های for, while و ...
- آشنایی با متغیر ها و داده ها
- یادگیری عملگر ها، توابع، ورودی و خروجی ها و ....

## ۴-۲- مقدمه

آموزش کامل از فهم ابتدایی برنامه نویسی، شروعی برای نوشتن برنامه هاست تا با کمک آن‌ها بتوان کدهایی در ربات ها، صنایع و در حرفه های ذی ربط نوشت و استفاده کرد. در نگاه کلی، می‌توان آنها را مهم ترین قسمت برنامه نویسی خواند.

## ۴-۳- ساختارهای کنترلی

ساختارهای تصمیم‌گیری نیازمند این است که برنامه نویس یک یا چند دستور کنترلی را برای ارزیابی و یا آزمایش توسط برنامه خود در پروژه تعیین کند که برای این منظور باید یک یا چند مرحله اجرا شود.

در صورتی که در مرحله اول جواب دستور کنترلی درست True باشد برنامه اجرا خواهد شد و مرحله بعد زمانی اتفاق می افتد که جواب دستور کنترلی False باشد؛ البته برای نوشتن یک جمله شرطی این قسمت اختیاری است.

دستورات کنترلی، قسمت هایی از برنامه هستند که روند اجرای برنامه را کنترل می‌کنند و در ادامه انواع آن را بررسی می‌شود.

### دستور شرطی if

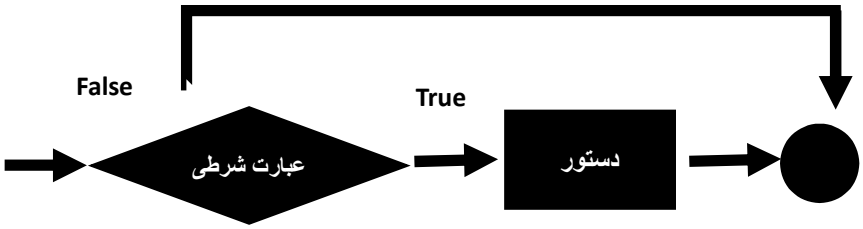
از مهم ترین دستورات کنترلی می باشد و شامل یک یا چند دستور شرطی داخل پرانتز است. اگر دستورات شرطی داخل پرانتز برابر با True شد (درست بود) عبارات یا دستورات درون بلوک شرط اجرا خواهند شد و در صورتی که شرط برقرار نباشد از اجرای دستورات داخل بلوک شرط صرف نظر خواهد شد. دستور شرطی if را به دو صورت می‌توان نوشت: در صورتی که با درست بودن شرط بخواهیم فقط یک عبارت یا دستور اجرا شود به صورت زیر استفاده می‌شود.

```
if (condition)
statement;
```

اگر بخواهیم در صورت درست بودن شرط چندین عبارت یا دستور اجرا شود، از آکولاد برای تعیین آنها استفاده می‌شود:

```
if (condition)
```

```
{
  Block of statements;
}
```

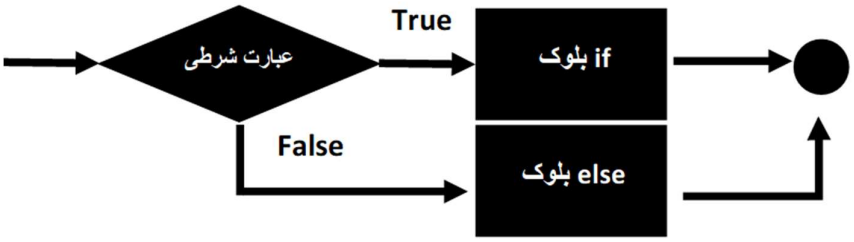



*(شکل ۴-۱) بلوک دیاگرام دستور If*

**دستور شرطی if _ else :**

مانند if می باشد و زمانی که شرط صحیح باشد همانند دستور if عمل خواهد کرد و زمانی که شرط صحیح نباشد عبارت یا دستوراتی که در قسمت else نوشته می‌شوند، اجرا خواهد شد.

```
if (expression)
{
    Block of statements;
}
else
{
    Block of statements;
}
```




*(شکل ۴-۲) دیاگرام If-else*

**دستورات شرطی تو در تو ( if _ else if _ else ):**

در اغلب اوقات پیش می آید که بعد از اینکه یک دستور شرطی بررسی شد و شرط آن درست یا غلط بود، شرط دیگری نیز بررسی شود که در این صورت می‌توان از دستورات شرطی پشت سر هم و یا دستورات تو در تو استفاده کرد.

```
if (expression)
```

```
{
    Block of statements;
}
else if(expression)
{
    Block of statements;
}
else
{
    Block of statements;
}
```

نکاتی که در نوشتن دستورات شرطی باید رعایت کرد :

دستور شرطی if می‌تواند دارای یک قسمت else باشد و یا اصلا نداشته باشد و در صورتی که وجود دارد باید بعد از همه دستورهای else if قرار گیرد.

دستور شرطی if می‌تواند دارای یک و یا چند قسمت else if باشد و یا اصلا نداشته باشد، در صورت وجود باید قبل از قسمت else قرار بگیرند.

در صورتی که شرط یکی از else if ها صحیح باشد فقط عبارات یا دستورات مربوط به همان شرط اجرا خواهد شد و مابقی else if ها و else نه بررسی می‌شوند و نه اجرا می‌شوند.

**دستور کنترلی Switch _ case :**

این دستور نیز همانند if ,else if, else روند اجرای برنامه را کنترل می‌کند. بدین صورت که به برنامه نویس اجازه می‌دهد کدهای مختلف که در شرایط مختلف باید اجرا شوند را تعیین کند و اجرای کد را کنترل کند.

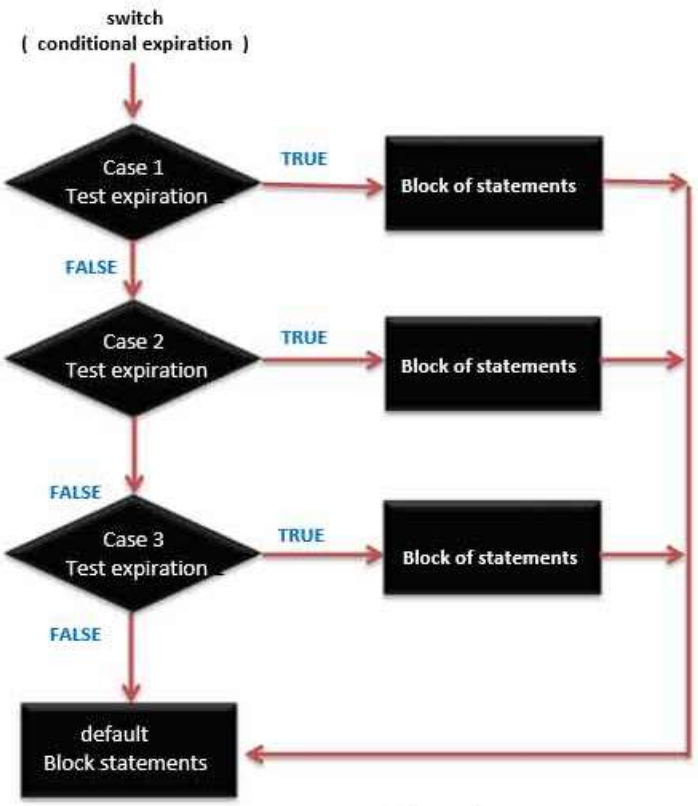
دستور Switch به طور خاص مقدار یک متغیر را بررسی می‌کند و با مقدار همه Case های موجود مقایسه می‌کند و در صورتی که با یکی از آنها برابر باشد، کد داخل بلوک آن case اجرا خواهد شد و case های دیگر اجرا نخواهند شد.

دستور Break به ما این امکان را می‌دهد که از بلوک case خارج شویم. در صورتی که از دستور Break در پایان بلوک Case استفاده نشود باعث می‌شود همه کیس‌های موجود را به ترتیب طی کند تا به دستور Break و یا به پایان بلوک Switch برسد.

```
switch (variable)
{
    case label:
          // statements
    break;
```

```
    case label:
    // statements
    break;

    default:
    // statements
    break;
}
```

در صفحه بعد بلوک دیاگرام و یک مثال از switch case گذاشته شده است.

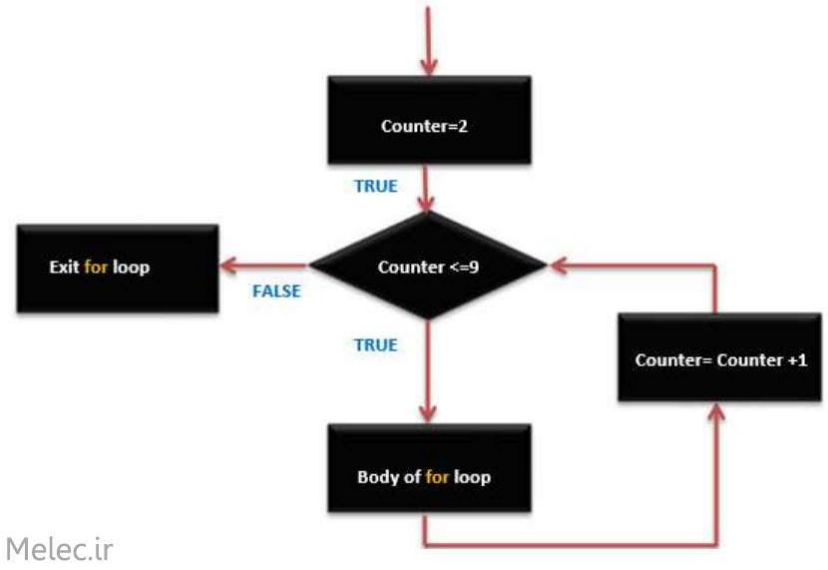



(شکل ۴-۳)بلوک دیاگرام switch

### حلقه for :

در حلقه for دستورات داخل آن به تعداد مشخصی اجرا می‌شود. شرط اجرای حلقه در همان ابتدای داخل پرانتز نوشته می شود و مقداردهی اولیه می شود. عیب یابی در حلقه for ساده است چون شرط اجرای حلقه به دستورات داخل حلقه ارتباطی ندارد.
هر حلقه for دارای ۳ بخش است که عملکرد این حلقه را مشخص می کنند. مثالی که در ادامه آورده می‌شود ساختار عمومی نگارشی حلقه for را نشان می دهد.
توجه شود که سه بخش موجود در پرانتز با ; از یکدیگر جدا می شوند.

```
for (initialize; control; increment or decrement)
{
    // statement block
}
```



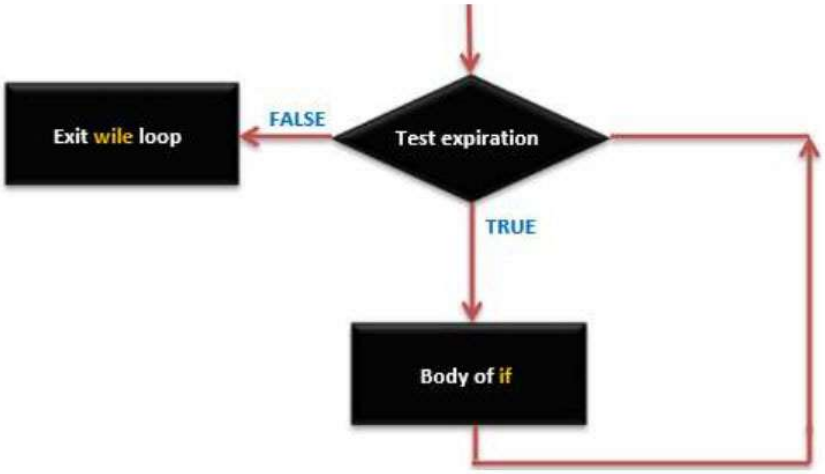

*(شکل ۴-۴) بلوک دیاگرام for*

### حلقه while :

دستورات داخل حلقه while به صورت دائم تا زمانی که عبارت داخل () نقض شود تکرار می گردد. در صورتی که متغیر داخل () تغییر نکند حلقه دائما تکرار می شود و یک حلقه بی‌نهایت خواهیم داشت.

```
while(expression)
```

```
{
Block of statements;
}
```




*(شکل ۴-۵) بلوک دیاگرام while*

**حلقه do...while** :

مشابه حلقه while است. در حلقه while شرط ادامه حلقه در ابتدای حلقه پیش از شروع اجرای دستورات داخل حلقه بررسی می شود. در حلقه do...while شرط اجرای حلقه در انتهای حلقه بررسی می شود و در نتیجه این حلقه حداقل یکبار اجرا خواهد شد. پس از اتمام حلقه do...whileدستوراتی که پس از این حلقه آمده است اجرا خواهند شد. در صورتی که دستورات داخل حلقه تنها یک خط است نیازی به استفاده از {} وجود ندارد اما برای این که شبیه حلقه whileباشد گاهی {} قرار داده می شود.

```
do
 {
    Block of statements;
 }
  while (expression);
```

**حلقه بی نهایت** : حلقه هایی که هیچ شرطی برای پایانش وجود ندارد و برای همیشه اجرا می شود.

استفاده از حلقه for :

```
for (;;)
 {
  // statement block
```

```
}
```

استفاده از حلقه while :

```
while(1)
 {
 // statement block
 }
```

استفاده از حلقه do...while :

```
do
{
    //Block of statements;
}
while(1);
```

**Break**: دستور break در آردوینو با نادیده گرفتن شرط، برای خروج از حلقه‌های for، do و while استفاده می‌شود. از این دستور همچنین برای خروج از کیس‌های ساختار switch نیز استفاده می‌شود.

**Continue**: این دستور در آردوینو باعث چشم پوشی از ادامه تکرار حلقه ( do , for while ) می شود و به قسمت بررسی عبارت شرطی حلقه می شود و با نتیجه ی بررسی شرط ، در صورت برقرار بودن شرط حلقه ، دستورات بدنه حلقه اجرا می شود. در واقع هنگامی که دستور Continue اجرا می شود ، دستورات بین دستور Continue تا انتهای حلقه اجرا نمی شود.

**Return**: این دستور در آردوینو تابع را خاتمه می دهد و اگر نیاز باشد، مقداری را به تابع فراخواننده برمی گرداند.

**Goto**: این دستور در آردوینو جریان برنامه را به نقطه ی برچسب گذاری شده انتقال می دهد.

```
label:

goto label;
```

**define**: یکی از اجزای مفید زبان c است و به برنامه‌نویس این امکان را می‌دهد که قبل از کامپایل کردن برنامه، به یک مقدار ثابت، نامی را اختصاص دهد. ثابت‌هایی که با استفاده از #define تعریف می‌شوند، هیچ فضایی از حافظه‌ی برنامه‌ی روی چیپ را اشغال نمی‌کنند. کامپایلر در زمان کامپایل، مقدار تعریف‌شده را به این ثابت‌ها اختصاص خواهد داد. اما این تعریف عوارض جانبی ناخواسته‌ای هم می‌تواند داشته باشد. به‌عنوان‌مثال اگر اسمی که قبلاً به یک ثابت از نوع #define داده‌ایم را در نام ثابت یا متغیری دیگر بیاوریم (در نام متغیر یا ثابت، این متنِ

اسم نیز آمده باشد)، به‌جای این اسم، مقدارِ (عددی یا متنی) csc در ثابتِ #define قرار خواهد گرفت.

به‌طورکلی تعریف ثابت‌ها با استفاده از const ارجحیت دارد و بهتر است به‌جای#define از const استفاده شود.

```
#define  مقدار نام‌ثابت
```

```
#define constantName value
```

**include** : در آردوینو برای اضافه کردن کتابخانه‌های خارجی به برنامه استفاده می‌شود. این کار باعث دسترسی برنامه‌نویس به گروه بزرگی از کتابخانه‌های استاندارد زبان سی (گروهی از توابع از پیش تعریف شده) و هم‌چنین کتابخانه‌هایی که به‌طور اختصاصی برای آردوینو نوشته شده‌اند، می‌شود.

```
#include <avr/pgmspace.h>
```

## ۴-۴- متغیر ها و داده ها

متغیر (Value) یک روش نام دهی و ذخیره‌ی مقدار برای استفاده در آینده، توسط برنامه است؛ به‌عنوان‌مثال، داده‌ای از یک سنسور یا مقداری واسطه‌ای که در یک محاسبه استفاده می‌شود.

تعریف متغیرها:

متغیرها باید پیش از استفاده، تعریف یا اعلان شوند. تعریف یک متغیر یعنی تعریف نوع آن و تنظیم مقدار اولیه برای آن می‌باشد.

```
Int value1;
int value2 = 0;
```

هر دو متغیر تعریف شده، عدد صحیح هستند .

از قوانینی که باید در نام گذاری متغیر رعایت شود :

۱- نام متغیر نباید با عدد شروع شود.

۲- نام متغیر نباید از keyword ها (کلمات ذخیره شده در آردوینو) باشد.

انواع داده ها در آردوینو:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| void | Boolean | char | Unsigned char | byte | int | Unsigned int | word |
| long | Unsigned long | short | float | double | array | String-char array | String-object |

*(شکل ۴-۶) جدول داده ها*

### Void

کلیدواژه void فقط در تعریف توابع مورد استفاده قرار میگیرد. این کلمه نشان دهنده این است که تابع فراخوانی شده به تابعی که فراخوانی را انجام می دهد داده ای ارائه نمی دهد یا مقدار بازگشتی ندارد.

```
Void loop()
{}
```

### Boolean

این نوع داده فقط میتواند مقدار ۰ یا ۱ ، صحیح یا غلط را نگه داری کند. هر متغیر از نوع Boolean یک بایت از حافظه را اشغال میکند.

```
Boolean value3 = false ;
Boolean value4 = true ;
```

### Char

این نوع داده یک بایت از حافظه را اشغال می کند و شامل یک کاراکتر است. کاراکترهای تکی برای ذخیره سازی باید به عنوان مثال به صورت 'A' نوشته شوند و برای ذخیره سازی رشته ها به عنوان مثال "ABC" نوشته می شود. اگر چه باید توجه شود که کاراکترها نیز مطابق کد ASCII به صورت عدد ذخیره سازی می شوند.

```
Char word1 = 'a';
char word2 = 65 ;
```

همچنین می‌توانیم روی کاراکترها عملیات حسابی مانند جمع ، تفریق و ... انجام داد زیرا این متغیرها به صورت عدد ذخیره میشوند.

### Unsigned char

این نوع از داده یک بایت از حافظه را اشغال می‌کند و میتواند از عدد ۰ تا ۲۵۵ را ذخیره سازی کند.

```
Unsigned char number = 123 ;
```

### Byte

این نوع داده نیز ۸ بیت را اشغال کرده و از عدد صفر تا ۲۵۵ را ذخیره سازی می کند.

```
Byte number2 = 255 ;
```

به منظور محکم سازی و استحکامِ سَبْک برنامه‌نویسی آردوینو، نوع‌داده‌ی byte در اولیت است. (بهتر است به‌جای unsigned char از byte استفاده شود.)

**Int**

از این نوع داده برای ذخیره‌سازی اعداد صحیح استفاده می‌شود. نوع int یک عدد دو بایتی (۱۶ بیت) را ذخیره سازی می‌کند که معادل است با -۳۲۷۶۸ تا ۳۲۷۶۷ .

(حداقل مقدار -۲^۱۵ و حداکثر ۲^۱۵-۱)

ظرفیت نوع int ممکن است در بردهای مختلف متفاوت باشد. مثلادر Arduino Due، این نوع از داده معادل ۳۲ بیت یا ۴ بایت است که معادل ذخیره سازی از عدد -۲۱۴۷۴۸۳۶۴۸ تا ۲۱۴۷۴۸۳۶۴۷ می باشد.

(حداقل مقدار -۲^۳۱ و حداکثر (۲^۳۱) – ۱)

```
int value2 = 0 ;
```

**Unsigned int**

این نوع از داده مشابه نوع int است با این تفاوت که فقط اعداد صحیح مثبت را ذخیره سازی می کند که شامل صفر تا ۶۵،۵۳۵ (۲^۱۶) – ۱ می شود. در برد Due این محدوده ۴ بایت است و از صفر تا ۴۲۹۴۹۶۷۲۹۵ (۲^۳۲ – ۱) را شامل می شود.

```
Unsigned int Max = 100 ;
```

**Word**

در برد Uno و بردهای دیگر مبتنی بر میکروهای ATMEGA ، این نوع داد ۱۶ بیت بدون علامت را ذخیره سازی می کند. در بردهای Due و Zero این نوع داده ۳۲ بیت بدون علامت را ذخیره سازی می کند .

```
Word A = 1000 ;
```

**Float**:

این نوع برای ذخیره سازی اعداد اعشار استفاده می شود. از نوع اعشار معمولا برای ذخیره سازی مقادیر آنالوگ و پیوسته استفاده می شود زیرا دارای دقت بالاتری نسبت به نوع صحیح می باشد. محدوده اعداد این نوع عبارت است از ۳۴۰۲۸۲۳۵ تا حداقل مقدار -۳۴۰۲۸۲۳۵ و در حافظه ۴ بایت ذخیره سازی می شود.

```
Float input = 25.102 ;
```

**Double**

در برد Uno و دیگر بردهای مبتنی برد میکروهای ATMEGA ، این نوع دارای ظرفیت ۲ برابر نسبت به float است و ۴ بایت را اشغال می کند ولی دقت آن مشابه float است. در برد Arduino Due این نوع دارای ۸ بایت (۶۴ بیت) دقت می باشد.

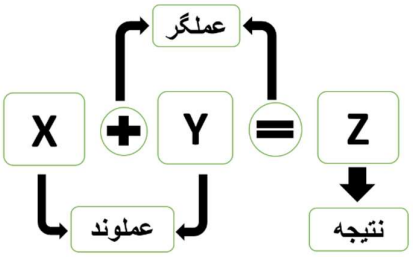
```
Double num = 12.345;
```

## ۴-۵- عملگرها (Operators)

پس از اینکه با انواع داده ها و متغیرها آشنا شدیم، باید عملیات هایی که می‌توان بر روی آن ها انجام داد را شناخت.

عملگرها نمادهایی هستند که اعمال خاصی را انجام می‌دهند، مانند جمع ، تفریق ، ضرب و... ، همچنین آن ها انواع مختلفی دارند که به شرح زیر می‌باشند:

- عملگرهای محاسباتی
- عملگرهای رابطه ای
- عملگرهای منطقی
- عملگرهای ترکیبی
- عملگرهای بیتی




*(شکل ۴-۷) مفهوم کلی عملگر با عملوند*

## عملگر های محاسباتی

این عملگرها اعمال محاسباتی را روی عملوندها یا مقادیر انجام می‌دهند؛ که در همه زبان های برنامه نویسی این عملگرها ( + . - . * . / ) مشترک هستند.

| عملگر | نوع عملگر | مثال |
|---|---|---|
| + | جمع | A + B |
| - | تفریق | A – B |
| * | ضرب | A * B |
| / | تقسیم | A / B |
| % | باقی مانده تقسیم | A % B |
| -- | کاهشی | --A یا A-- |
| ++ | افزایشی | ++A یا A++ |

*(شکل ۴-۸) جدول عملگر های محاسباتی*

## اولویت عملگرهای محاسباتی

زمانی که در یک برنامه از چندین عملگر استفاده می‌شود، ترتیب اجرا با دو ویژگی زبان برنامه نویسی C++ تعیین می‌شود. ابتدا تقدم عملگرها و بعد شرکت پذیری عملگرها که بدین شرح است:

بالاترین تقدم متعلق به عملگرهای کاهشی و افزایشی میباشد ( ++ ، -- )

تقدم بعدی متعلق به ضرب ، تقسیم و باقی مانده تقسیم میباشد ( * ، / ، ٪ )

پایین ترین تقدم هم متعلق به جمع و تفریق میباشد ( + - )

### عملگرهای رابطه ای

این نوع عملگرها ارتباط بین دو متغیر را مشخص می‌کنند ؛ عملگرهایی مانند مساوی ، کوچکتر و بزرگتر و... از این نوع می‌باشند.

| عملگر | نوع عملگر | مثال |
|---|---|---|
| > | بزرگتر | A > B |
| >= | بزرگتر یا مساوی | A >= B |
| < | کوچکتر | A < B |
| <= | کوچکتر یا مساوی | A <= B |
| == | متساوی | A == B |
| != | نامتساوی | A != B |

*(شکل ۴-۹) جدول عملگر های رابطه ای*

### عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می‌کند. عبارات منطقی دارای دو ارزش درستی و نادرستی است. به عبارتی ارزش نادرستی با صفر یعنی False مشخص می‌شود. ارزش درستی با مقادیر غیر صفر True مشخص می‌شود. ثابت True به عدد یک و ثابت False به عدد صفر تبدیل می‌شود.

| عملگر | نوع عملگر | مثال |
|---|---|---|
| ! | نقیض : not | !A |
| && | و : AND | A>B && C<D |
| \|\| | یا : OR | A>B \|\| C<D |

*(شکل ۴-۱۰) جدول عملگر های منطقی)*

### تقدم عملگرهای منطقی و رابطه ای

ابتدا دو عملگر ! و عملگر && بررسی می‌شود.

نتیجه ی عملگر ! وقتی درست است که دارای ارزش نادرستی باشد.

نتیجه ی عملگر && وقتی درست است که ارزش درستی داشته باشد.

نتیجه ی عملگر || وقتی نادرست است که ارزش نادرستی داشته باشد.

عملگرهای ترکیبی

ترکیب دو عملگر محاسباتی و = ، عملگرهای ترکیبی را ایجاد میکند.

| عملگر | نوع عملگر | مثال | معادل |
|---|---|---|---|
| += | انتساب جمع | A += B | A = A + B |
| -= | انتساب تفریق | A -= B | A = A – B |
| *= | انتساب ضرب | A *= B | A = A * B |
| /= | انتساب تقسیم | A /= B | A = A / B |
| %= | انتساب باقی مانده تقسیم | A %= B | A = A % B |

*(شکل ۴-۱۱) جدول عملگر های ترکیبی*

عملگرهای بیتی

تست، مقدار دهی و شیفت کاربرد عملگرهای بیتی است.

| عملگر | نام |
|---|---|
| & | و AND |
| \| | یا OR |
| ^ | یا انحصاری XOR |
| ~ | نقیض NOT |
| >> | شیفت به راست RIGHT SHIFT |
| << | شیفت به چپ LEFT SHIFT |

*(شکل ۴-۱۲) عملگر های بیتی*

نتیجه‌ی عملگر & وقتی یک است که هر دو بیت یک باشد.

نتیجه‌ی عملگر | وقتی صفر است که هر دو بیت صفر باشد.

نتیجه‌ی عملگر ^ وقتی یک است که یکی از بیت ها صفر و دیگری یک باشد.

عملگر شیفت

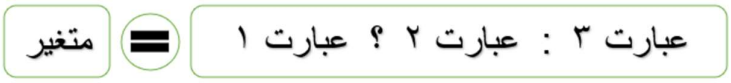
عملگر شیفت به راست و شیفت به چپ به این صورت تعریف میشود:


در عملگر شیفت ، متغیر یک بایت از حافظه است که باید به تعداد مشخص شده به سمت راست یا چپ شیفت داده میشود. هنگام شیفت دادن به راست ، بیت ها از سمت راست خارج میشود و از سمت چپ به تعداد مورد نظر صفر وارد میشود. در شیفت به چپ بیت ها از سمت چپ خارج شده ، به تعداد لازم صفر از سمت راست وارد میشود.

بعضی از عملگرها شامل هیچکدام از دسته بندی های ذکر شده نیست اما بسیار کاربرد دارد.

عملگر ؟

این عملگر یک عبارت را بررسی کرده و سپس بر اساس ارزش آن عبارت که ، به صورت TRUE و FALSE بوده، نتیجه‌ی عبارت دیگر را در متغیری قرار میدهد.




اگر عبارت ۱ دارای ارزش درستی باشد ، مقدار ارزیابی شده عبارت ۲ در متغیر قرار میگیرد وگرنه مقدار ارزیابی شده عبارت ۳ در متغیر قرار خواهد گرفت.

عملگر کاما ( , )

برای انجام چند عمل در یک دستور یا تعریف چند متغیر از کاما استفاده میکنیم.

```
int a , b , c ;
```

## ۴-۶- توابع

توابع فراخوانی‌های کدهای از پیش نوشته شده توسط شخص سومی هستند که کارهای مختلفی را انجام می‌دهند. این توابع حاوی کدهایی برای انجام کارهایی مانند دریافت داده از یک سنسور، تنظیم وضعیت ولتاژ یک پین یا نمایش متن بر روی صفحه نمایش LCD هستند.

توابع یک عملکرد را در یک نقطه کدنویسی می کنند لذا تنها کافی است که یک بار عملکرد آنها تایید شود. این نکته همچنین باعث کاهش خطاها در فرایند تغییر کدنویسی می شود. توابع برنامه را کوتاه‌تر و سبک‌تر می‌کنند زیرا توابع نوشته شده چندین بار در برنامه مورد استفاده قرار می‌گیرند.

اعلان توابع:

یک تابع در خارج از توابع دیگر در بالا یا پایین تابع loop تعریف می شود.

این روش نوشتن تنها بخشی از تابع است که به آن نمونه اولیه تابع گفته می شود و در قسمت بالای تابع loop نوشته میشود که شامل :

- مقدار بازگشتی تابع
- نام تابع
- نوع ورودی ها (آرگومان ها) تابع، نوشتن نام آرگومان‌ها اهمیتی ندارد.

پس از نوشتن نمونه اولیه تابع باید ازسمیکالن ( ; ) استفاده شود.

در ادامه با انواع توابع آشنا میشویم که ما را در نوشتن برنامه نویسی کمک شایانی میکند.

توابع زمان

millis(): این دستور در آردوینو برای شمارش تعداد میلی ثانیه ها استفاده میشود. زمانی که برنامه و برد آردوینو شروع به کار می‌کند، تعداد میلی ثانیه ها را می‌شمارد و در یک داده از نوع unsigned long ذخیره می‌کند. شمارش تقریبا بعد از ۵۰ روز سرریز میشود و به صفر بر میگردد.

micros(): برای شمارش تعداد میکروثانیه ها استفاده میشود. این دستور از زمانی که برنامه آردوینو شروع به کار می‌کند، تعداد میکروثانیه ها را می‌شمارد و در یک داده از نوع unsigned long ذخیره می کند. شمارش تقریبا بعد از ۷۰ دقیقه سرریز میشود و به صفر بر می‌گردد.

delay (): دستور delay در آردوینو برای ایجاد تاخیر در برنامه استفاده می‌شود. برنامه را برای مدت زمان تعیین شده (بر حسب میلی ثانیه) که به عنوان پارامتر مشخص شده متوقف می‌کند. (در هر ثانیه ۱۰۰۰ میلی ثانیه وجود دارد)

```
delay(ms);
```

به جای ms مدت زمان تاخیر مورد نیاز وارد می‌شود. مثلا برای توقف ۲ ثانیه ای باید دستور را به این شکل نوشت :

```
delay(2000);
```

delayMicroseconds(): این دستور در آردوینو برای ایجاد تاخیر بر حسب میکروثانیه استفاده می شود. delayMicroseconds(us) برنامه را برای مدت زمان مشخص برحسب میکروثانیه متوقف می‌کند. هزار میکرو ثانیه در یک میلی ثانیه و یک میلیون میکرو ثانیه در ثانیه وجود دارد.

توابع ریاضی

min(): از این دستور در آردوینو برای تشخیص مقدار کوچکتر استفاده می‌شود. در تابع min() دو مقدار وارد می‌شود و آردوینو محاسبه می‌کند که کدام مقدار کوچک‌تر از مقدار دیگری است.

نحوه نوشتن:

```
min(number1 , number2 );
```

پارامترها:

number1: عدد اول ورودی number2: عدد دوم ورودی

خروجی : مقدار کوچک تر را تشخیص می دهد و آن را برمی گرداند.

```
value = min(number1 , number2 );
```

max(): برخلاف دستور min() ، این دستور در آردوینو برای تشخیص مقدار بزرگتر استفاده می شود. در تابع max() شما دو مقدار وارد می‌شود و آردوینو محاسبه می‌کند که کدام مقدار بزرگتر است.

نحوه نوشتن و پارامترها دقیقا مانند دستور min می باشد؛ با فرق اینکه در خروجی مقدار بزرگتر را تشخیص داده و آن را بر می‌گرداند.

```
max( number1 , number2 );
```

abs(): برای محاسبه قدر مطلق مقادیر در آردوینو استفاده می‌شود.

نحوه نوشتن :

```
abs(x);
```

پارامتر:

x: عدد مورد نظر که قدر مطلق آن را می خواهیم محاسبه کنیم.

خروجی : x اگر عدد مثبت باشد، خود عدد را برمی‌گرداند و اگر عدد منفی باشد، منفی شده عدد را بر می گرداند.

Constrain(): برای ایجاد یک محدوده و رِنج به کار می‌رود. زمانی استفاده می شود که کاربر نمی‌خواهد عدد خروجی از یک محدوده کمتر یا بیشتر باشد.

نحوه نوشتن :

```
constrain(x , a , b);
```

پارامترها :

x: عدد ورودی شما a: تعیین حداقل خروجی b: تعیین حداکثر خروجی

خروجی :

اگر عدد بین a و b باشد ، خود مقدار x برمی گردد.

اگر عدد کمتر از a باشد ، مقدار بازگشتی برابر با a است.
و اگر عدد بیشتر از b باشد ، مقدار بازگشتی برابر با b است.
pow(): مقدار عددی که به توان رسیده است را محاسبه می کند. در این تابع ما عدد پایه و توان را تعیین می کنیم و آردوینو عدد نهایی را محاسبه می کند . از این دستور برای تولید منحنی ها و ... استفاده می شود.
نحوه نوشتن :

```
pow(base , exponent);
```

پارامترها :
base: عدد پایه که می تواند از نوع داده float باشد.
exponent: عدد توان که می تواند از نوع داده float باشد.
خروجی : عدد پایه را به توان می رساند و نتیجه را بازگشت می دهد.
sqrt(): ریشه عدد را محاسبه می کند. در این تابع ما مقدار را تعیین می کنیم و آردوینو از آن جذر می گیرد.
نحوه نوشتن :

```
sqrt(x);
```

پارامتر :
x : عدد ورودی که آردوینو آن را زیر رادیکال می برد و جذر آن را محاسبه می‌کند.
خروجی : ریشه عدد ورودی را بر می گرداند.
توابع مثلثات
sin(): برای محاسبه سینوس استفاده می شود.
نحوه نوشتن :

```
sin(rad);
```

پارامتر:
rad: مقدار زاویه که بر حسب رادیان است.
خروجی : مقدار سینوس زاویه را برمی گرداند که می تواند بین ۱ و ۱- باشد .
cos(): برای محاسبه کسینوس استفاده می شود.
نحوه نوشتن :

```
cos(rad);
```

پارامتر:
rad: مقدار زاویه که بر حسب رادیان است.
خروجی : مقدار کسینوس زاویه را برمی گرداند که می تواند بین ۱ و ۱- باشد .

tan(): برای محاسبه تانژانت استفاده می شود.
نحوه نوشتن :

```
tan(rad);
```

پارامتر:
rad: مقدار زاویه که بر حسب رادیان است.
خروجی : مقدارتانژانت زاویه را برمی گرداند که می‌تواند بین مثبت بی نهایت تا منفی بی نهایت باشد.

## ۴-۷- ورودی و خروجی ها

ورودی / خروجی دیجیتال
pinMode( ) : این تابع در آردوینو پین مشخص شده را تنظیم می کند که به عنوان ورودی (input) یا خروجی (output) عمل کند.
نحوه نوشتن :

```
pinMode(pin , mode);
```

پارامتر ها :
pin: عدد پین یا نامی که می‌خواهید مدش (mode) را تنظیم کنید.
mode: انتخاب از بین ثابت های INPUT_PULLUP یا INPUT یا OUTPUT
( در ادامه به طور کامل با ثابت ها آشنا میشویم)
Digital Write()

LOW یا HIGH این تابع در آردوینو روی یک پین دیجیتال یک مقدار می نویسد.
اگر پین با pinMode() به عنوان OUTPUT تنظیم شده باشد، ولتاژ آن به مقدار متناظر ۵ ولت ( یا ۳,۳ ولت روی بردهای ۳,۳ ولت ) برای HIGH و ۰ ولت (زمین) برای LOW تنظیم خواهد شد. اگر پین به‌عنوان OUTPUT پیکربندی شده باشد، digitalWrite() پول آپ درونی روی پین ورودی را فعال (HIGH) یا غیرفعال (LOW) خواهد کرد. پیشنهاد می‌شود که برای فعال کردن مقاومت پول آپ درونی، pinMode() را به INPUT_PULLUP تنظیم (ست) شود. می‌توان برای برای اطلاعات بیشتر به digital pins tutorial مراجعه کرد.

نکته: اگر pinMode() را به‌عنوان INPUT تنظیم نکنید و یک LED را به یک پین وصل کنید، هنگام فراخوانی digitalWrite (HIGH) ، ممکن است LED کم نور به نظر برسد. بدون تنظیم pinMode() به‌طور مشخص digitalWrite() ، مقاومت (resistor) پول آپ درونی را فعال خواهد کرد که مثل یک مقاومت محدودکننده‌ی جریانِ بزرگ عمل می‌کند.

نحوه نوشتن:

```
digitalWrite(pin, value);
```

پارامتر ها:

Pin: عدد پین

mode: ثابت HIGH یا LOW

مثال:

```
int ledPin = 13;
                          // ال ای دی به پین شماره‌ی سیزده متصل شده
void setup()
{
  pinMode(ledPin, OUTPUT);
                        // پین دیجیتال را به‌عنوان خروجی تنظیم می‌کند
}
void loop()
{
  digitalWrite(ledPin, HIGH);
                                    // ال ای دی را روشن می‌کند
  delay(1000);
                                    // یک ثانیه صبر می‌کند
  digitalWrite(ledPin, LOW);
                                   // ال ای دی را خاموش می‌کند
  delay(1000);
                                    // یک ثانیه صبر می‌کند
}
```

## digitalRead()

این تابع در آردوینو مقدار یک پین دیجیتال مشخص شده که HIGH یا LOW است را می خواند.

نحوه نوشتن:

```
digitalRead(pin);
```

پارامتر ها:

pin : عدد پین دیجیتالی که می‌خواهید بخوانید(int) .

خروجی: HIGH یا LOW

ورودی / خروجی آنالوگ

### analogWrite()

این تابع در آردوینو یک مقدار آنالوگ (PWM wave) را روی پین می‌نویسد. از analogWrite() می‌توان برای تنظیم روشنایی یک ال ای دی با شدت نور متنوع یا راندن یک موتور با سرعت‌های مختلف استفاده کرد. پس از فراخوانی analogWrite()، تا فراخوانی بعد همین دستور (یا فراخوانی digitalRead() یا digitalWrite() روی همان پین )پین یک موج مربعی ثابت را طبق چرخه‌ی وظیفه‌ی (دوره‌ی کاری duty cycle) مشخص‌شده، تولید خواهد کرد. در بیشتر پین‌ها، فرکانس سیگنال PWM، تقریباً ۴۹۰ هرتز است. در آردوینو Uno و بردهای مشابه، پین‌های ۵ و ۶ حدوداً فرکانس ۹۸۰ هرتز دارند. پین‌های ۳ و ۱۱ Leonardo هم فرکانس ۹۸۰ هرتز دارند.

نحوه نوشتن:

```
analogWrite(pin , value );
```

پارامتر ها :

pin: پین برای نوشتن.

مقدار چرخه‌ی وظیفه: بین 255 (همیشه on ) و 0 (همیشه off )

### analogRead()

این تابع در آردوینو داده را از پین آنالوگ مشخص شده می‌خواند. برد آردوینو حاوی یک مبدل آنالوگ به دیجیتال ۶ کاناله 8) کاناله در Mini و Nano و ۱۶ کاناله در (Mega و ۱۰ بیتی است؛ به این معنا که این مبدل، ولتاژهای ورودی بین ۰ تا ۵ ولت را به مقداری صحیح(integer) بین ۰ تا ۱۰۲۳، تبدیل خواهد کرد. این کار رزولوشنی بین خوانش‌های:

۵ ولت / ۱۰۲۴ واحد یا ۰۰۴۹. ولت (4.9 mV) بر واحد را نتیجه خواهد داد. این محدوده‌ی ورودی و رزولوشن را می‌توانید با استفاده از analogReference() تغییر دهید. خواندن یک ورودی آنالوگ، تقریباً حدود ۱۰۰ میکروثانیه (۰،۰۰۰۱ ثانیه) طول می‌کشد؛ پس حداکثر (ماکزیمم) نرخ خواندن، حدود ۱۰،۰۰۰ بار در ثانیه است.

نحوه نوشتن :

```
analogRead( pin ) ;
```

پارامتر:

pin: عدد پین آنالوگی که می‌خواهیم از آن بخوانیم (روی بیشتر بردها از ۰ تا ۵، از ۰ تا ۷ روی Mini و Nano و ۰ تا ۱۵ روی Mega)

خروجی : از نوع int ( صفر تا ۱۰۲۳ )

**مثال:**

```
int led = 8;      پایه متصل به ال ای دی //
int analogpin = 3;      پایه متصل به پتانسیومتر //
int dcyc = 0;     متغیر برای ذخیره چرخه وظیفه //

Void setup(){
      pinMode(led, OUTPUT);
}
Void loop(){
      dcyc =analogRead(analogpin) مقدار ورودی خوانده میشود//
      analogWrite (led, dcyc / 4)  ال ای دی روشن میشود //
}
```

## ۴-۸- ثابت‌ها

ثابت ها عبارات از پیش تعریف شده در زبان برنامه نویسی آردوینو هستند. از آنها برای سهولت در خواندن برنامه‌ها استفاده می‌شود. ثابت‌ها را در چند گروه طبقه بندی می‌شود.

### true و false

دو ثابت وجود دارد که برای نشان دادن درست (true) و غلط (false) در زبان آردوینو استفاده می شود.

false به عنوان ۰ (صفر یا غلط) تعریف می شود.

غالباً گفته می شود true به عنوان ۱ تعریف می شود، که صحیح است ، اما true تعریف گسترده تری دارد. هر عدد صحیحی که غیر صفر باشد، به معنای بولی true است. بنابراین ۱-، ۲ و ۲۰۰- همه به معنای بولی نیز true تعریف می شوند.

### HIGH و LOW

هنگام خواندن یا نوشتن روی یک پین دیجیتال ، فقط دو حالت وجود دارد که یک پین می تواند بگیرد : HIGH وLOW

معنای HIGH (با اشاره به یک پین) بسته به اینکه پین روی INPUT یا OUTPUT تنظیم شده باشد ، تا حدودی متفاوت است. هنگامی که یک پین به عنوان INPUT با دستور PinMode پیکربندی شود و با دستور digitalRead خوانده شود در دو حالت زیر مقدار HIGHرا بازگشت می‌دهد:

- ولتاژ بیشتر از ۳,۰ ولت در پین وجود دارد (برد های ۵ ولت)
- ولتاژ بیشتر از ۲,۰ ولت در پین وجود دارد (برد های ۳,۳ ولت)

همچنین ممکن است یک پین به عنوان INPUT با pinMode پیکربندی شود و با دستور digitalWrite در حالت HIGH تنظیم شود. این کار مقاومت پول آپ داخلی که ۲۰ کیلو اهم می‌باشد را فعال می‌کند تا در حالت HIGH قرار بگیرد مگر اینکه توسط یک مدار خارجی LOW شود.

این کار را می توان با قرار دادن INPUT_PULLUP به عنوان حالت در دستور pinMode انجام داد..

هنگامی که یک پین با pinMode در حالت خروجی (OUTPUT) پیکربندی شود و با digitalWrite روی HIGH تنظیم شود، پین در ۲ حالت زیر قرار می‌گیرد.

- ۵ ولت (برد ۵ ولت)
- ۳,۳ ولت (برد های ۳,۳ ولت)

در این حالت می‌توان یک LED را روشن کرد.( پایه منفی را به GND متصل کنید)

معنای LOW نیز بسته به تنظیم بودن پین روی INPUT یا OUTPUT معنای متفاوتی دارد. هنگامی که یک پین به عنوان INPUT با pinMode پیکربندی شود و با digitalRead به عنوان LOW خوانده شود، حالت های زیر وجود خواهد داشت:

- ولتاژ کمتر از ۱,۵ ولت در پین وجود دارد (برد های ۵ ولت)
- ولتاژ کمتر از ۱,۰ ولت (تقریبا) در پین وجود دارد (برد های ۳,۳ ولت)

وقتی یک پین با pinMode در حالت OUTPUT پیکربندی می شود و باdigitalWrite روی LOW تنظیم می شود ، پین در حالت ۰ ولت است (در برد های ۵ ولت و ۳,۳ ولت). در این حالت میتوان ال ای دی را روشن کرد (پایه دیگر را به پین ۵ ولت یا یک مقاومت سری یا پین ۳,۳ ولت متصل کنید).

### OUTPUT و INPUT و INPUT_PULLUP

پین ها درحالت INPUT

پین هایی که با دستور pinMode در حالت INPUT قرار بگیرند در یک حالت با امپدانس بالا قرار دارند. برای خواندن مقادیر سنسور ها بسیار مناسب هتسند. اگر پین خود را به عنوان INPUT پیکربندی کنید و بخواهید وضعیت یک کلید را بخوانید، بهتر است از یک مقاومت پول آپ استفاده کنید. هدف این مقاومت بالا کشیدن پین سوئیچ در حالت باز بودن است. به طور معمول از مقاومت ۱۰ کیلو اهم استفاده میشود تا حالت غیر مطمئن و نویز ها را از بین ببرد.

پین ها در حالت INPUT_PULLUP

میکروکنترلر ATmega در Arduino دارای مقاومت پول آپ داخلی (مقاومت هایی است که به طور داخلی به برق متصل می شوند) که می توانید به آنها دسترسی داشته باشید. اگر ترجیح می دهید از اینها به جای مقاومت کششی خارجی استفاده کنید ، می توانید از حالت INPUT_PULLUP در دستور pinMode استفاده کنید.

پین هایی که به صورت ورودی با INPUT یا INPUT_PULLUP پیکربندی شده اند ، اگر به ولتاژ های منفی یا بیشتر از محدوده مثبت (۵ ولت یا ۳ ولت) متصل شوند ، آسیب می بینند یا از بین می روند.

پین ها در حالت OUTPUT

پین هایی که به صورت OUTPUT با دستور pinMode پیکربندی شده اند در حالت امپدانس پایین قرار دارند. این بدان معنی است که آنها می توانند مقدار قابل توجهی جریان را برای سایر مدار ها فراهم کنند. پین های ATmega می توانند تا ۴۰ میلی آمپر جریان را برای دستگاه ها / مدارهای دیگر تأمین کنند. این امر آنها را برای تأمین انرژی LED مفید می کند زیرا LED ها معمولاً کمتر از ۴۰ میلی آمپر استفاده می کنند. برای تامین انرژی مولفه های دیگر مثل موتور ها که به جریان الکتریکی بیشتر از ۴۰ میلی آمپر نیاز دارند، به ترانزیستور یا رابط‌های دیگر احتیاج دارید. پین هایی که به عنوان خروجی پیکربندی شده اند ، اگر به ولتاژ – و + بالا اتصال داشته باشند می توانند آسیب ببینند یا از بین بروند.

## ۴-۹- وقفه‌ها

وقفه یا intterupt یک سیگنال ریزپردازنده است که به توجه و پاسخ سریع CPU نیاز دارد. هنگامی که یک وقفه رخ می‌دهد، پردازنده عملیات جاری خود را متوقف می‌کند تا به درخواست وقفه(کدهای خاصی که دربرنامه نوشته شده است) رسیدگی کند. پس از رسیدگی به درخواست مورد نظر و انجام آن، برنامه از همان‌جایی که متوقف شده است شروع به اجرا می‌کند. استفاده از وقفه باعث مصرف کمتر انرژی، بهینه تر شدن کد، درک بهتر از کد و همچنین استفاده کمتر از منابع میکروکنترلر می‌شود.

توابع واحد وقفه در آردوینو :

Software Interrupts: آنها در پاسخ به دستورالعمل ارسال شده در نرم افزار رخ می دهند. تنها نوع وقفه ای که "زبان آردوینو" از آن پشتیبانی می کند ،attachInterrupt() است.

Hardware Interrupts: آنها در پاسخ به یک رویداد خارجی رخ می دهند ، مانند یک پین قطع خارجی که HIGH یا LOW میشود.

دستور interrupts() : برای فعال سازی واحد وقفه از این دستور استفاده می‌شود.

دستور ()nointerrupts : برای غیر فعال سازی واحد وقفه از این دستور استفاده می‌شود.
دستور ()attachInterrupt : از این دستور برای پیکربندی واحد وقفه استفاده می‌شود و این دستور باعث میشود پایه های مربوط به وقفه از حالت ورودی خروجی خارج می‌شوند.
دستور ()detachInterrupt : این دستور پایه هایی که به عنوان وقفه تعریف شده اند رو از حالت وقفه خارج میکند.

## ۴-۱۰- ارتباط سریال

:Serial.Begin(9600)

مهم ترین تابع در زمینه ارتباط سریال است. عدد ۹۶۰۰ در این تابع به معنی سرعت یا نرخ باود (Baud Rate) است.

:Serial.available()

این تابع بررسی می کند که آیا داده های سریال در دسترس هستند یا خیر. یعنی اگر داده های سریال موجود باشند این تابع مقدار ۱ را برمیگرداند. از این تابع اغلب در شروع کد و در یک شرط استفاده می شود.

:Serial.find()

داده ها را از بافر سریال میخواند. اگر داده مربوطه یافت شود این تابع مقدار ۱ را بر می‌گرداند و اگر هدف یافت نشود مقدار ۰ را بر می‌گرداند.

:Serial.print()

برای چاپ مقدار در سریال مانیتور یا ارسال مقادیر از طریق ارتباط سریال استفاده می شود. کافیست در میان پرانتز مقدار مورد نظر برای چاپ را بنویسید.

:Serial.println()

همانند تابع Serial.print() عمل میکند با این تفاوت که داده ها را در یک خط جداگانه مینویسد در صورتی که تابع Serial.print() داده ها را پشت سر هم مینویسد و در سریال مانیتور خوانایی آن پایین می آید.

:Serial.read()

داده های سریال ورودی را می خواند. این تابع به شکل Serial.read() در کد قرار می‌گیرد و کافی است مقدار یک متغیر را برابر با این تابع قرار دهیم تا داده های ورودی در آن ذخیره شود.

:Serial.write()

برای نوشتن و ارسال مقادیر از طریق ارتباط سریال استفاده میشود. این تابع داده های باینری را

از طریق پورت سریال ارسال می‌کند .کافیست متغیر مورد نظر یا رشته دلخواه خود را در پرانتز این تابع قرار دهید.

## ۴-۱۱- خودآزمایی

۱. دستور شرطی تو در تو را توضیح دهید.
۲. نمونه کد از دستور switch case بیاورید.
۳. انواع داده ها را نام برده و دو مورد را توضیح دهید.
۴. تابع زمان millis را شرح دهید.
۵. کد روشن و خاموش کردن یک ال ای دی در پایه ۱۱ با تاخیر ۱ ثانیه.
۶. دستورات ارتباط سریال را نام ببرید.

# ۵

# ماژول ها و برنامه نویسی آنها

## ۵-۱- اهداف این فصل

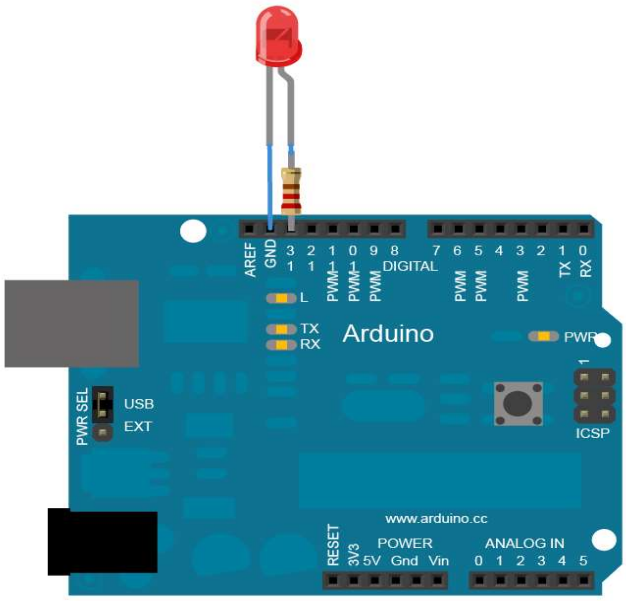
- آشنایی با قطعه ها و نحوه کد نویسی با آنها
- راه اندازی ماژول ها و نوشتن کد برای آنها
- شناسایی و بررسی پایه ها و پین ها

## ۵-۲- مقدمه

در این فصل با نحوه نوشتن کد و راه اندازی components و ماژول های مرسوم که با آردوینو بیشتر استفاده می‌شود، پرداخته شده و نحوه عملکرد آنها مورد بررسی قرار گرفته است.

## ۵-۳- راه اندازی LED

برای راه اندازی لامپ LED و اتصال آن به برد آردوینو باید همانند نقشه شماتیک زیر عمل کرد، به این صورت پایه بلند لامپ LED که نشان دهنده پلاریته مثبت آن است را به واسطه مقاومت ۲۲۰ اهمی به یکی از پایه های دیجیتال آردوینو متصل می‌کنیم؛ و پایه کوتاه تر LED که نشان دهنده پلاریته منفی آن است را به منفی منبع تغذیه یا GND بر روی آردوینو متصل کرد.




*(شکل ۵-۱) نحوه پیکربندی ال ای دی با آردوینو [۶].*

برنامه نویسی:

در اولین خط برنامه ما پایه D13 در آردوینو که با نام LED نام گذاری شده است را به عنوان خروجی پیکربندی می‌کنیم.

```
pinMode(LED, OUTPUT);
```

میتوان به جای کلمه LED در pinMode شماره پایه مورد نظر را نوشت.

```
pinMode(13, OUTPUT);
```

در حلقه اصلی برنامه لامپ LED را روشن میکنیم با این دستور:

```
digitalWrite(LED, 1);
```

در خط بعدی مدت زمانی را تعین میکنیم تا LED روشن بماند و سپس خاموش شود، این مدت زمان برای این است که کاربر بتواند روشن و خاموش شدن LED را به صورت واضح مشاهده کند. به همین خاطر دستور زیر مدت زمان تاخیر را به واحد میلی ثانیه برای پردازشگر تعریف میکنیم.

```
Delay (1000);
```

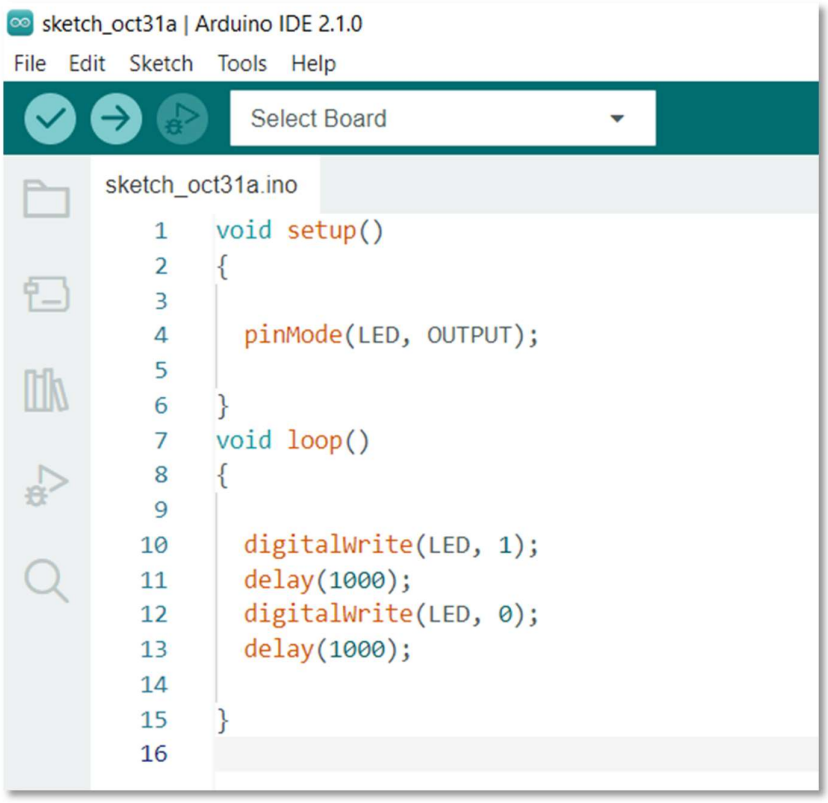
در خط بعدی برنامه در همان حلقه اصلی برنامه LED را با دستور زیر خاموش میکنیم.

```
digitalWrite (LED, 0);
```

در انتهای حلقه دوباره همانند زمان روشن بودن LED یک دستور تاخیر برای زمان خاموش بودن LED برای پردازشگر تعریف می‌شود.

```
Delay (1000);
```

در نهایت می‌شود:




*(شکل ۵-۲) کد مربوط به روشن و خاموش کردن ال ای دی*

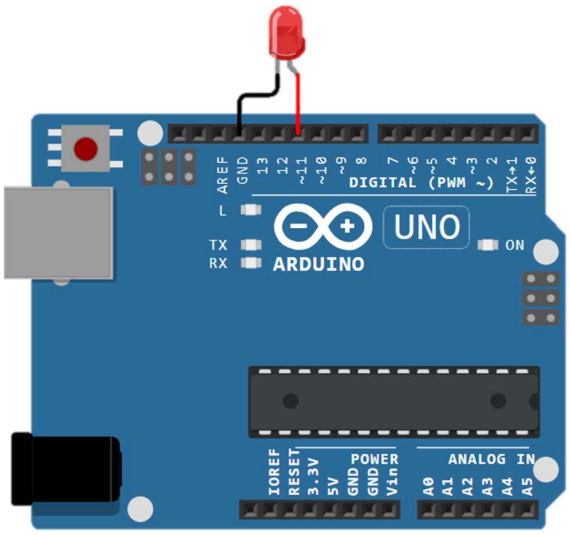
## تنظیم نور ال ای دی

در این برنامه با استفاده از دستور for، ال ای دی از حالت خاموش، در زمان مشخص به بالاترین حد روشنایی خود رسیده و سپس تکرار می‌شود.

```
int led_Pin = 11;
int i = 0;
void setup ()
{
  pinMode (led_Pin, OUTPUT);
}
void loop ()
{
   for (i=0; i <= 255; i++) {
```

```
        analogWrite (led_Pin, i);
        delay (10);
    }
}
```

نحوه اتصال ال ای دی به آردوینو



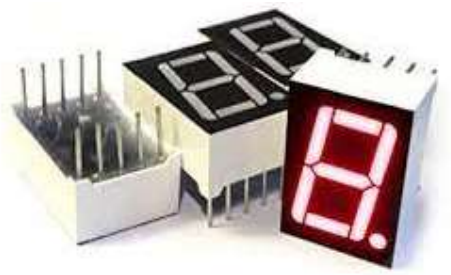

*(شکل ۵-۳) نحوه اتصال با ال ای دی [۶].*

## ۵-۴- راه اندازی سون سگمنت

سون سگمنت‌ها (seven-segment display) دسته‌ای از نمایشگرهای پرکاربرد الکترونیکی هستند که می‌توانند اعداد ۰ تا ۹ را نمایش دهند. از آن‌ها به طور عمده در ساعت‌های دیجیتال، تایمرها و ماشین‌حساب‌ها برای نمایش اطلاعات عددی استفاده می‌شود.

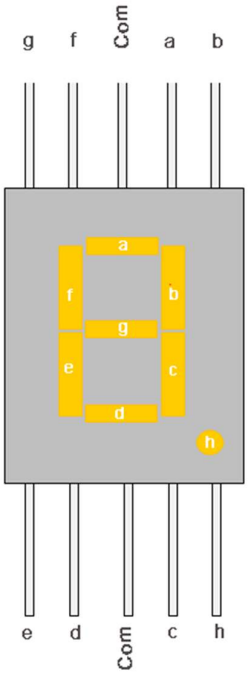
این قطعه از هشت عدد LED ساخته شده است که هفت عدد از آن‌ها مسئول روشن کردن ۷ بخش سون سگمنت، و هشتمین LED نیز مسئول روشن نمودن علامت نقطه در گوشه‌ی سمت راست پایین سون سگمنت است. هر بخش شبیه یک خط تیره است. این خطوط در کنار هم روشن و خاموش می‌شوند و خطوط روشن هربار یک حرف یا عدد را نمایش خواهند داد.

در برخی از منابع هفت خط سون سگمنت را با نام‌های a ، b، c، d، e، f، g یاد می‌کنند و نقطه را نیز با نماد h نشان می‌دهند.



*(شکل ۵-۴) سون سگمنت [۱۲].*

این قطعه دارای ۱۰ پایه است که ۸ پایه به led های a تا h متصل‌اند و ۲ پایه نیز به آند/کاتد مشترک تمام led ها متصل است. از آنجا که آند/کاتد مشترک تمام led ها معمولا در داخل خود سون سگمنت به هم اتصال کوتاه می‌شود، تنها کافی است سیگنال مورد نظر را به یکی از این پایه‌های مشترک اعمال کرد.




*(شکل ۵-۵) جزئیات سون سگمنت [۱۲].*

راه اندازی سون سگمنت (7-segment) با استفاده از Arduino

پین ها به عنوان یک خروجی تنظیم می‌شود.

```
void setup () {
pinMode (2, OUTPUT);
pinMode (3, OUTPUT);
pinMode (4, OUTPUT);
pinMode (5, OUTPUT);
pinMode (6, OUTPUT);
pinMode (7, OUTPUT);
pinMode (8, OUTPUT);

}
void
```

این توابع به ترتیب هر عدد را نمایش می‌دهند.

```
zero();
one();
two();
three();
four();
five();
six();
seven();
eight();
nine();
}
```

با استفاده از دستورات digitalWrite پین‌های مربوط به نمایش اعداد روشن و خاموش می شود. سپس با دستور delay(1000) به مدت ۱ ثانیه نمایش داده می‌شوند و سپس خاموش می‌شود.

```
Void zero() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
  delay(1000);
}
```

```
void one(){
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);
}
void two() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  delay(1000);
}
void three() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  delay(1000);

 }
void four() {
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
```

```
  digitalWrite(8,HIGH);
  delay(1000);
}
void five() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}


void six() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}
void seven() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);
  }
void eight() {
  digitalWrite(2,HIGH);
```

```
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}

void nine() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

همه توابع نمایشگر برای نمایش هر عدد از ۰ تا ۹، پین‌های مربوط به هر عدد را به وضعیت مناسب تنظیم می‌کنند.

این کد برای آموزش و تست سون سگمنت با Arduino مناسب است، برای پروژه‌های واقعی می‌توان از کد های بهینه تر و ساده‌تری برای کنترل سون سگمنت استفاده کرد برای مثال کد زیر کارایی کد بالا را دارد اما بسیار کوتاه تر می باشد.

```
#include “SevSeg.h”                    // فراخوانی کتابخانه سون سگمنت
SevSeg sevseg;

void setup()
{
    byte numDigits = 1;                // تعیین تعداد رقم های سون سگمنت
    byte digitPins[] = {};             // پین رقم ها
    byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
                                       // پین a-g های سگمنت ها
    bool resistorsOnSegments = true;
                                       // تعیین مقاومت متصل به سگمنتها

    byte hardwareConfig = COMMON_CATHOD   // تعیین کاتد مشترک
```

```
sevseg.begin(hardwareConfig,numDigits,digitPins,segmentP
ins, resistorsOnSegments);
}

void loop()
{
    for(int I = 0; I < 10; i++)
      {                    // یک حلقه ایجاد می کند که از ۰ تا ۹ را می شمارد

        sevseg.setNumber(I, i%2);   //عدد روی نمایشگر تنظیم می‌شود
        delay(1000);
        sevseg.refreshDisplay();
                    // نمایشگر را پاک می کند تا عدد مورد نظر نمایش داده شود
     }
}
```

برای نوشتن کد بالا ابتدا باید کتابخانه سون سگمنت را نصب کنید. کتابخانه ای که در این کد از آن استفاده شده کتابخانه SevSeg می باشد و از آن برای کنترل سون سگمنت استفاده می‌شود.

دستور hardwareConfig = COMMON_CATHODE نوع سون سگمنت را تعیین می‌کند. اگر از یک سون سگمنت کاتد مشترک استفاده شود باید از همین دستور استفاده نمود اما اگر از یک سون سگمنت آند مشترک استفاده شود به جای آن باید از دستور hardwareConfig = COMMON_ANODE استفاده می‌شود.

دستور byte numDigits = 1 تعداد ارقام روی سون سگمنت را مشخص می‌کند. برای مثال اگر از سون سگمنت تک رقمی استفاده شود باید به آن مقدار ۱ را داد یا اگر از نمایشگر ۲ رقمی استفاده شود باید مقدار ۲ را وارد شود.

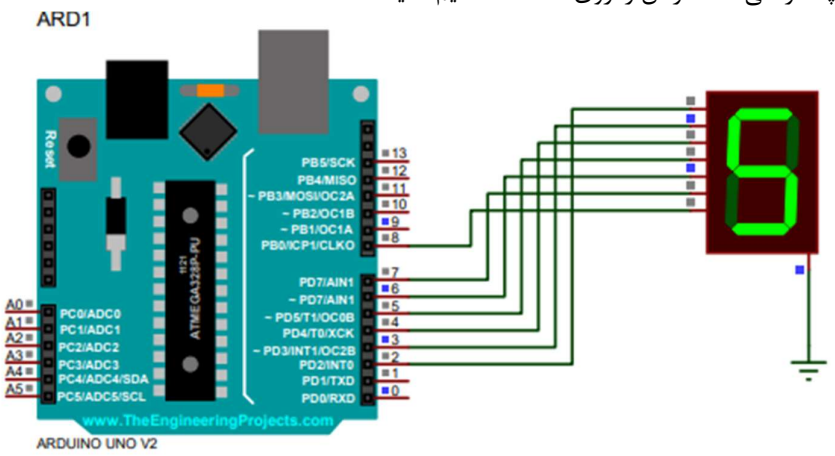
دستور Byte digitPins[] = {} آرایه ای ایجاد می‌کند که پایه های زمین (ground) را هنگام استفاده از نمایشگر چند رقمی مشخص می‌کند. به عنوان مثال اگر سون سگمنت تک رقمی استفاده شود باید آن را خالی گذاشت و یا اگر سون سگمنت ۴ رقمی استفاده شود و می خواهید از پین های ۱، ۲، ۳، و ۴ آردوینو به عنوان پایه‌های زمین استفاده شود، باید از این حالت استفاده نمود:

byte digitPins[] = {1, 2, 3, 4}

دستور byte segmentPins[] آرایه ای را ایجاد می کند که مشخص می کند کدام پایه های آردوینو به هر سگمنت از نمایشگر متصل است. این آرایه به ترتیب حروف الفبا است.

.( A , B , C , D , E , F , G )

هنگام استفاده از دستور resistorsOnSegments اگر مقاومت های محدود کننده فعلی شما با پین های سگمنت سری هستند، باید آن را روی true تنظیم کنید و اگر مقاومت ها با پین های رقمی سری هستند باید آن را روی false تنظیم کنید. هنگام استفاده از نمایشگرهای چند رقمی مقدار آن را روی true تنظیم کنید.




*(شکل ۵-۶) پیکربندی سون سگمنت با آردوینو*

## ۵-۵- راه اندازی ماژول آلتراسونیک

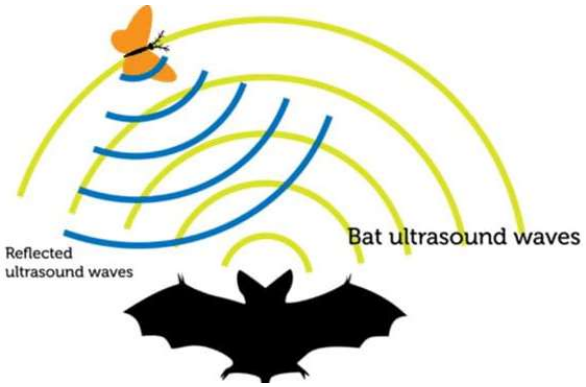
سنسور التراسونیک از امواج فراصوتی برای سنجش استفاده می‌کند. این باعث میشود تا تفاوتی در استفاده از آن در محیط روشن یا تاریک وجود نداشته باشد. موج صوتی یک نوع موج مکانیکی است که معمولاً به صورت یک جبهه فشاری حرکت می‌کند. البته انواع دیگری از امواج مکانیکی نیز وجود دارد و موج صوتی تنها یکی از آن ها می‌باشد.

ویژگی امواج مکانیکی نیاز به وجود محیط مادی برای انتقال است. این امواج می‌توانند در گازها، مایعات یا جامدات منتقل شوند. تفاوت مهم محیط‌های مختلف در انتشار امواج صوتی، سرعت صوت در آنهاست. هرچه چگالی محیط بیشتر باشد، سرعت انتشار موج صوتی در آن نیز بیشتر خواهد شد. برای مثال، در شرایط یکسان سرعت انتشار موج صوتی در محیط جامد بیشتر از مایع و در محیط مایع بیشتر از گاز است.

موج صوتی می‌تواند فرکانس‌های مختلفی داشته باشد و می‌توان بر این اساس آنها را دسته‌بندی کرد. امواج التراسونیک یا فراصوت دسته‌ای از امواج صوتی است که فرکانس آن بالاتر از محدوده شنوایی انسان (۲۰ کیلوهرتز) است.

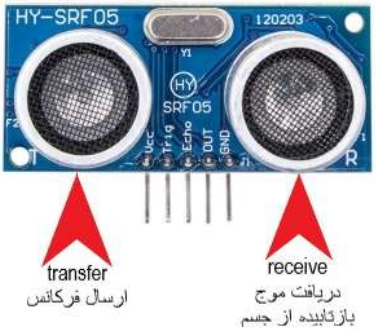
نحوه عملکرد سنسور التراسونیک :

ایده سنسورهای التراسونیک از حیوانات گرفته شده است. دلفین و خفاش موجوداتی هستند که توانایی انتشار و شنیدن امواج التراسونیک را دارند و از این قابلیت برای تخمین فاصله خود تا موانع و اشیا استفاده می‌کنند. خفاش چشمهای ضعیفی دارد اما از امواج التراسونیک مثل یک رادار پیشرفته استفاده می‌کند و با آن می‌تواند محیط اطرافش را حتی در تاریکی مطلق شناسایی کند.



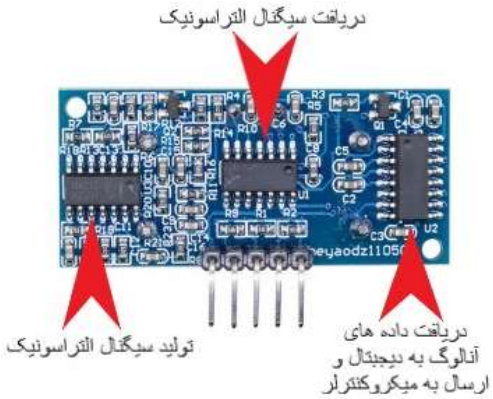

*(شکل ۵-۷): نمای آلتراسونیک در طبیعت [۱۲].*

بر روی این ماژول دو قسمت استوانه‌ای شکل دیده می‌شود. توسط یکی از این دو قسمت یک موج التراسونیک با فرکانس ۴۰ کیلوهرتز ارسال می‌شود. این موج در هوا حرکت کرده و در صورت برخورد به یک جسم خارجی بازتابیده شده و به سمت سنسور بر می‌گردد. قسمت دوم ماژول وظیفه دریافت موج بازتابیده را بر عهده دارد. بر روی این بخش یک قطعه پیزوالکتریک حساس به ارتعاش وجود دارد. برخورد یک موج التراسونیک به این قطعه باعث به وجود آمدن یک ولتاژ الکتریکی در آن می‌شود.



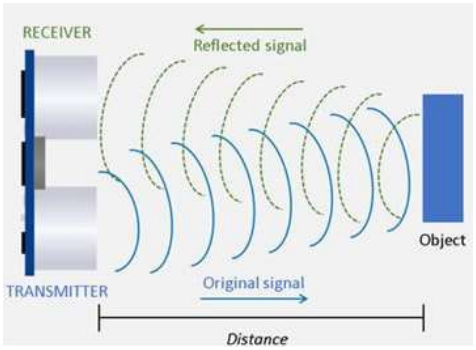

*(شکل ۵-۸) تصویر التراسونیک [۱۲].*

در پشت سنسور التراسونیک، سه تراشه وجود دارد. یکی سیگنال التراسونیک را تولید کرده، دیگری سیگنال التراسونیک را دریافت کرده و تراشه سوم داده‌های آنالوگ را به دیجیتال تبدیل کرده و به میکروکنترلر می‌فرستد.




*(شکل ۵-۹) نمای پشت التراسونیک [۱۲].*

زمانی که طول می‌کشد تا موج از فرستنده ارسال شده به مانع برخورد کرده و توسط سنسور دریافت شود مبنای اندازه‌گیری فاصله بین سنسور تا مانع است. طبق رابطه زیر سرعت برابر است با فاصله طی شده تقسیم بر زمان. از آنجایی که زمان اندازه گیری شده از سنسور التراسونیک، زمان رفت و برگشت موج است، برای محاسبه فاصله باید نصف این زمان در نظر گرفته شود.

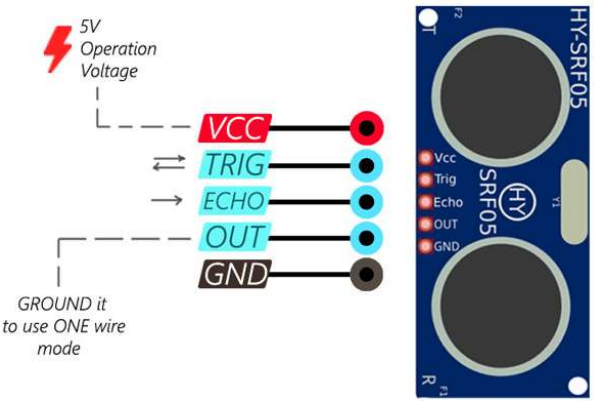



*(شکل ۵-۱۰) نحوه کار التراسونیک [۱۲].*

سنسور التراسونیک انواع مختلفی دارد نمونه هایی مانند :

HC-SR04 , SRF05 , SRF06 , JSN-SRF04T

نحوه کارکرد این سنسورها تقریبا یکسان است و تفاوت آنها عمدتا به میزان دقت، محدوده کاری، ضد آب بودن و غیره مربوط می‌شود.




*(شکل ۵-۱۱) پایه های التراسونیک [۱۲].*

ماژول التراسونیک دارای ۵ پایه است، نام هر پایه در کنار آن نوشته شده است. دو پایه GND و VCC ماژول را به 5V و GND آردوینو و پایه های Trig و Echo را به دو پایه دیجیتال (۱۲ و ۱۳ دیجیتال) آردوینو وصل می‌شود.

پایه Trig ورودی ماژول است و فرمان ارسال موج التراسونیک به آن وارد می‌شود . پایه Echo خروجی ماژول است و به موج دریافتی از محیط حساس است.

برنامه نویسی التراسونیک

تعریف ثابت در ابتدا برنامه جهت دسترسی به پین ها با نام مستعار:

```
const int TRIG_PIN = 12;
const int ECHO_PIN = 13;
const int LED = 8;
```

تنظیم پایه های ورودی و خروجی به عنوان دریافت و ارسال دیتا:

```
void setup()
{
  Serial.begin(9600);
  pinMode (LED,OUTPUT)
  pinMode (TRIG_PIN,OUTPUT)
  pinMode (ECHO_PIN,INPUT)
}
```

جهت اینکه مقدار پالس بالا در پینها پاک شود یک پالس پایین در حد ۲ میکروثانیه به پین ارسال شود:

```
int duration, distanceCm;
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
duration = pulseIn(ECHO_PIN,HIGH)
```

تبدیل مقادیر دریافتی به سانتی متر:

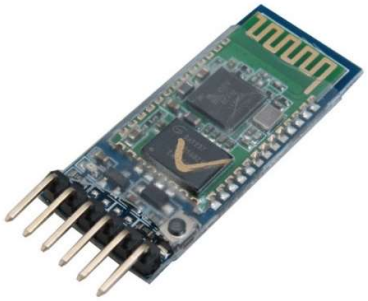
```
distanceCm = duration / 29.1 / 2;
```

اگر فاصله محاسبه شده کمتر از ۱۰ سانتی متر باشد led روشن میشود و در غیر اینصورت مقدار فاصله را محاسبه میکند و از طریق سریال نمایش می‌دهد:

```
if (distanceCm <= 10){
  Serial.println("OUT OF RANGE");
  digitalWrite(LED , HIGH);
}

else{
  digitalWrite(LED , LOW);
  Serial.print(distanceCm);
}
```

## ۵-۶- راه اندازی ماژول بلوتوث HC-05

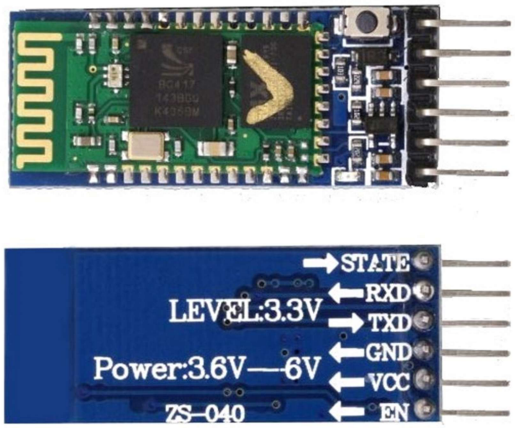
بلوتوث یکی از روش های عالی برای ارتباط بی سیم است که در بسیاری از زمینه ها استفاده می‌شود و برق بسیار کمی مصرف می کند.



*(شکل ۵-۱۲) ماژول HC-05 [۱۳].*

ماژول بلوتوث HC-05 :

ماژولی است که می تواند دوطرفه ارتباط برقرار کند. ما می توانیم آن را با اکثر میکروکنترلرها راه اندازی شود. زیرا این ماژول از طریق ارتباط سریال می‌تواند با هر میکروکنترلر و برد آردوینو که دارای واحد ارتباط سریال است، ارتباط برقرار کند. باود ریت پیشفرض این ماژول ۹۶۰۰ است اما میتوان باود ریت آن را تغییر داد. HC-05 می تواند در دو حالت کار کند؛ یکی حالت داده و دیگری حالت فرمانAT . هنگامی که پین Enalbe روی ماژول "LOW"است، HC-05 در حالت داده است. اگر آن پین به عنوان "HIGH" تنظیم شود، ماژول در حالت فرمان AT است.




*(شکل ۵-۱۳) نمای متفاوت از HC-05 [۱۳].*

مشخصات فنی :

ولتاژ کاری: ۴ تا ۶ ولت (معمولاً ۵ ولت)

جریان عملیاتی: ۳۰ میلی آمپر

برد : کمتر از ۱۰۰ متر

با ارتباط سریال و سازگار با TTL کار می کند

می توان به راحتی با لپ تاپ یا تلفن همراه با بلوتوث ارتباط برقرار کرد

اتصالات ماژول با آردوینو طبق جدول زیر می باشد:

| Arduino Uno | HC-05 |
|---|---|
| Rx | Tx |
| Tx | Rx |
| 5v | 5v |
| GND | GND |

*(شکل ۵-۱۴) جدول اتصال با آردوینو [۱۳].*

برنامه نویسی

ابتدا متغیری به نام "data" را با نوع داده char ایجاد میکنیم و مقدار اولیه آن را "۲" تنظیم می‌شود. با توجه به شرایطی که در ادامه تعریف میکنیم، زمانی که این داده برابر ۲ باشد ال ای دی خاموش است. برنامه را طوری تنظیم می‌توان کرد که در زمان شروع ال ای دی خاموش باشد.

```
char data = '2';
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT)
}
```

سپس باید قسمت setup را کدنویسی کرد. HC-05 از ارتباط سریال استفاده می‌کند. بنابراین ارتباط سریال را با استفاده از تابع Serial.begin() آغاز می‌کنیم. نرخ باود را روی ۹۶۰۰ تنظیم می‌کنیم. سپس پایه دیجیتال ۱۳ را به عنوان پایه "OUTPUT" تنظیم کنید.

سپس قسمت loop کدنویسی می‌شود. از حلقه while و تابع"Serial.available()" استفاده می‌شود. این تابع تعداد بایت های موجود برای خواندن را بر می‌گرداند. سپس داده‌های موجود در پورت سریال خوانده می‌شود. برای اینکار از تابع"Serial.read()" استفاده می‌شود. سپس آن را در داده "data" که ایجاد شده ذخیره می‌شود. سپس با یک شرط " if " اگر "data"برابر با "۱" باشد، پین ۱۳ "HIGH" می‌شود و ال ای دی روشن می‌شود. در مرحله بعد از شرط " else if " برای خاموش کردن LED استفاده می کنیم. شرط خاموش کردن LEDبه شکل "data==2" نوشته می‌شود.

```
void loop() {
  while(Serial.begin()>0)
  {
```

```
    data = Serial.read();
    Serial.println(data);
    if (data == '1')
    {
      digitalWrite(13,HIGH);
    }
    else if (data == '2')
    {
      digitalWrite(13,LOW);
    }
  }
}
```

نرم افزار اندروید ارتباط بلوتوث با آردوینو :

حالا باید یک برنامه را در موبایل خود نصب کنید. در حقیقت می‌توانید از هر برنامه بلوتوث سریالی که میخواهید استفاده کنید. کافی است به پلی استور رفته و Bluetooth Serial را سرچ کرده ویکی از نرم افزار ها را نصب کنید.

سپس به قسمت بلوتوث گوشی خود بروید و با ماژول بلوتوث Hc-05 جفت سازی(pair) را انجام دهید. اگر از شما رمز خواسته شد، ۱۲۳۴ را وارد کنید. در نهایت وارد اپلیکیشن بلوتوث سریال شوید و مشاهده می‌کنید که با ارسال عدد ۱ ال ای دی روشن میشود و با ارسال عدد ۲ ال ای دی خاموش میشود.

## ۵-۷- راه اندازی سنسور PIR

سنسورهای PIR یا Passive Infra-Red بر پایه تشخیص نور مادون قرمز ناشی از گرما در محیط کار می‌کنند. به همین دلیل، در سیستم‌های امنیتی برای شناسایی حرکت انسان یا ورودهای غیرمجاز به طور گسترده استفاده می‌شوند. راه‌اندازی اولیه و تنظیم این سنسورها حدود ۱۰ تا ۶۰ ثانیه طول می‌کشد و در این دوره، به منظور تنظیم دقیق آنها، توصیه می‌شود هیچ حرکتی در محدوده دید سنسور انجام نشود.

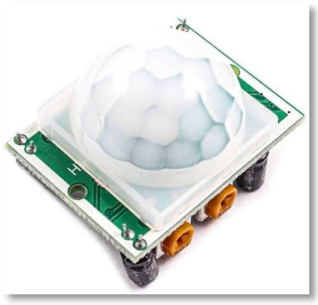
این سنسور دارای دو دریچه به شکل مستطیلی است، که از موادی ساخته شده‌اند که امکان عبور امواج مادون قرمز را فراهم می‌کنند. در پشت این دو دریچه، دو الکترود سنسور مادون قرمز قرار دارند که یکی خروجی مثبت و دیگری خروجی منفی تولید می‌کنند. اتصال دو الکترود به نحوی است که مقادیر خروجی هرکدام از آن‌ها یکدیگر را خنثی می‌کنند. در صورتی که یک الکترود بیشتر یا کمتر امواج مادون قرمز را دریافت کند، خروجی سنسور به طور طبیعی به وضعیت HIGH یا LOW تغییر خواهد کرد. زمانی که هیچ حرکتی در محیط اتفاق نیافتاده باشد، هر دو الکترود مادون قرمز امواج یکسانی را دریافت کرده و نتیجه تفاضل خروجی دو الکترود صفر خواهد

بود. اما هنگامی که یک جسم گرم مانند بدن انسان یا حیوان از جلوی این سنسور عبور می‌کند، سنسور PIR تحت تأثیر قرار گرفته و ایجاد تغییر تفاضلی مثبت در خروجی دو الکترود خواهد کرد. در هنگامی که جسم گرم محیط را ترک می‌کند، این فرآیند به صورت معکوس رخ خواهد داد و تغییر تفاضلی منفی ایجاد می‌شود. این سیگنال پالسی باعث فعال شدن پین خروجی سنسور می‌شود.

از سنسور های PIR در سیستم های امنیتی و ساخت دزدگیر ، کنترل و اتوماسیون صنعتی ، سیستم های هشدار ، درب های اتوماتیک ، نور پردازی هوشمند و ... استفاده می شود.

ماژول HC-SR501 :

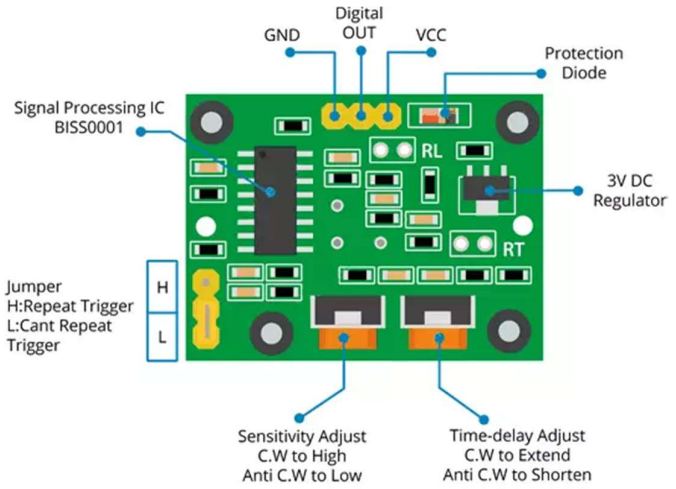
در بسیاری از پروژه‌های آردوینو که نیاز به تشخیص حرکت ورود و خروج یا تشخیص نزدیک شدن اشخاص داریم، سنسور PIR مدل HC-SR501 به عنوان یک انتخاب مناسب شناخته می‌شود. این سنسورها به دلیل هزینه کم، مصرف انرژی کم و تنوع لنزهای مختلف در بازار، جزو انتخاب‌های محبوب می‌باشند. همچنین، این سنسورها به عنوان یکی از ساده‌ترین و کارآمدترین گزینه‌ها در پروژه‌های مختلف شناخته شده‌اند.



*(شکل ۵-۱۵) ماژول HC-SR501*

ماژول HC-SR501 دارای سه پین است؛ پین VCC برای تغذیه، پین GND برای اتصال به زمین و پین Digital OUT که به عنوان خروجی داده‌ها عمل می‌کند.

همچنین، این سنسور دارای یک رگولاتور ولتاژ داخلی است که می‌تواند با ولتاژ متنوع از ۴,۵ تا ۱۲ ولت تغذیه شود. اما به طور معمول، تغذیه سنسور با ولتاژ ۵ ولت مدنظر قرار می‌گیرد.




(شکل ۵-۱۶) نمای پشت و شرح اجزا

ماژول HC-SR501 دارای دو پتانسیومتر برای تنظیم پارامترهای حساسیت وتایم سنسور دارد.
پتانسیومتر حساسیت: (Sensitivity) برای تنظیم حداکثر فاصله‌ای که در آن فاصله حرکت قابل تشخیص باشد استفاده می‌شود. رنج فاصله قابل تنظیم بین ۳ متر تا حدود ۷ متر می‌باشد.
پتانسیومتر زمان: (Time) مشخص کننده مدت زمان فعال بودن خروجی سنسور پس از آشکارسازی حرکت می‌باشد. این زمان بین حداقل ۳ و حداکثر ۳۰۰ ثانیه قابل تنظیم است.
ماژول HC-SR501 دارای یک جامپر است که دارای دو حالت کاری می‌باشد:
حالت H (Hold/Repeat/Retriggering) : در این حالت خروجی سنسور تا زمانی که سنسور حرکت را آشکار کند، کماکان فعال نگه داشته خواهد شد.
حالت L (No-Repeat/Non-Retriggering) : در این حالت به محض تشخیص حرکت توسط سنسور، خروجی سنسور برای مدت زمان مشخصی که توسط پتانسیومتر تایم تنظیم شده است، فعال نگه داشته خواهد شد.
ماژول HC-SR501 دارای دو پد اضافی به نام‌های RT و RL می باشد. پد RT برای یک ترمیستور یا مقاومت حرارتی در نظر گرفته شده است که امکان کار سنسور در دماهای خیلی بالا را فراهم می‌کند. همچنین صحت اندازه گیری سنسور در برخی دماهای خاص را افزایش می دهد و پد RL برای اتصال یک مقاومت حساس به نور (LDR) و یا فتوسل در نظر گرفته شده است. با اضافه کردن مقاومت LDR، سنسور PIR فقط در محیط های تاریک کار خواهد کرد که برای کاربردهای سیستم روشنایی حساس به حرکت قابل استفاده است.

راه اندازی ماژول HC-SR501 با استفاده از آردوینو

```
int ledPin = 13;
int inputPin = 8;
int pirState = LOW;
int val = 0;

void setup(){
  pinMode(ledPin,OUTPUT);
  pinMode(inputPin, INPUT);
  Serial.begin(9600);
}
```

int ledPin = 13: تعریف یک متغیر به نام ledPin که به پین ۱۳ متصل شده است.

int inputPin = 8: تعریف یک متغیر به نام inputPin که به پین ۸ متصل شده است. این پین مربوط به سنسور PIR است.

int pirState = LOW : تعریف یک متغیر به نام pirState که در ابتدا به مقدار LOW (خاموش) تغییر داده می شود. این متغیر برای بررسی وضعیت حالت حرکت است.

int val = 0 : تعریف یک متغیر به نام val که برای خواندن وضعیت پین ورودی به کار می‌رود.

pinMode(ledPin, OUTPUT) : تعیین نوع پین ledPin به OUTPUT (خروجی) برای کنترل LED.

pinMode(inputPin, INPUT) : تعیین نوع پین inputPin به INPUT (ورودی) برای خواندن وضعیت سنسور PIR.

Serial.begin(۹۶۰۰): شروع ارتباط سریال با سرعت ۹۶۰۰ بیت بر ثانیه.

```
void loop(){
  val = digitalRead(inputPin, HIGH);
  if (val == HIGH)
  {
    digitalWrite(ledPin,HIGH);

    if (pirState == LOW)
    {
      Serial.println("Motion Detected");
    }
  }

  else
```

```
  {
    digitalWrite(ledPin,LOW);

    if (pirState == HIGH)
    {
      Serial.println("Motion Ended");
      priState = LOW;
    }
  }
}
```

val = digitalRead(inputPin): خواندن وضعیت پین ورودی و ذخیره آن در متغیر `val`.

if (val == HIGH): بررسی اینکه آیا مقدار خوانده شده از ورودی HIGH است یا خیر.

digitalWrite(ledPin, HIGH) : روشن کردن LED در صورت تشخیص حرکت.

if (pirState == LOW): بررسی وضعیت حالت حرکت و چاپ پیام در صورت تغییر وضعیت.

else: اگر حرکت تشخیص داده نشده باشد، به این قسمت وارد می‌شود.

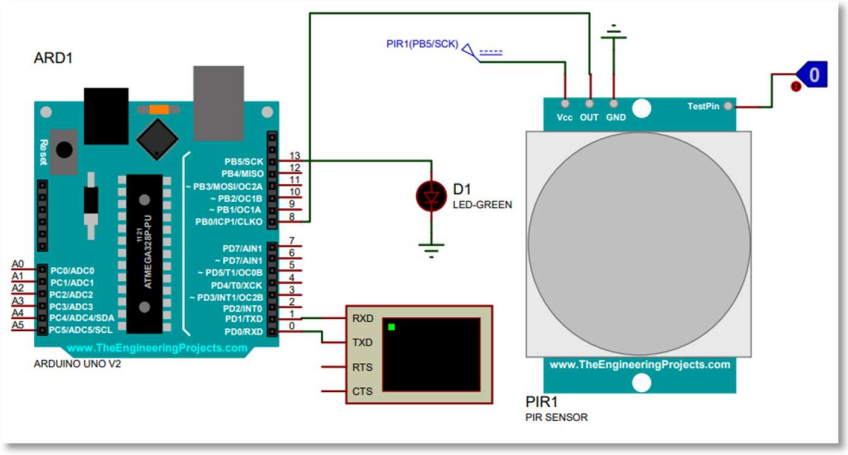
digitalWrite(ledPin, LOW) : خاموش کردن LED.

if (pirState == HIGH): بررسی وضعیت حالت حرکت و چاپ پیام در صورت تغییر وضعیت.

Serial.println("Motion ended!"): چاپ پیام در صورت پایان حرکت در پورت سریال.

pirState = LOW: تغییر وضعیت متغیر `pirState` به LOW (خاموش).

شبیه سازی در نرم افزار پروتئوس:




*(شکل ۵-۱۷) پیکربندی مدار در پروتئوس*

## ۵-۸- راه اندازی سنسور دما LM35

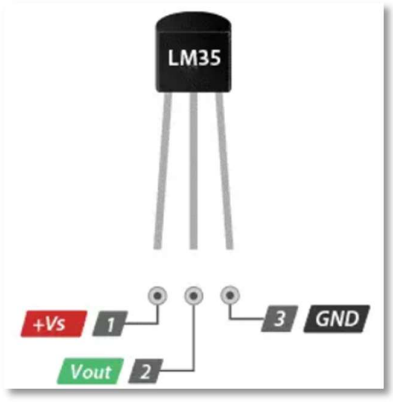
LM35 یک سنسور دمای ولتاژ پایین ساخته شده توسط Texas Instruments است که می‌تواند دما را بر حسب درجه سانتی گراد اندازه گیری کند. خروجی این تراشه، سیگنال ولتاژی است که به طور خطی با دما (برحسب درجه سانتی‌گراد) تغییر می‌کند ، بنابراین استفاده از آن با آردوینو بسیار ساده است.

سنسور دمای LM35 نسبتاً دقیق است ، هرگز فرسوده نمی‌شود، در شرایط محیطی مختلف کار می‌کند و برای اندازه‌گیری دما به قطعات خارجی نیاز ندارد. علاوه براین ، سنسور LM35 نیازی به کالیبراسیون ندارد و معمولاً با دقت ±۰٫۵ درجه سانتی‌گراد در دمای اتاق و ±۱ درجه سانتی‌گراد در محدوده دمایی ۵۵- تا ۱۵۵+ درجه سانتی‌گراد، دما را اندازه می‌گیرد .

ولتاژ تغذیه این سنسور ۴ تا ۳۰ ولت و جریان مصرفی آن هنگام تبدیل دما کمتر از ۶۰ میکروآمپر است. همچنین خود گرمایی بسیار پایینی دارد (کمتر از ۰٫۰۸ درجه سانتی گراد در هوای ساکن).

### پایه‌های سنسور LM35

LM35 به سه شکل مختلف عرضه می‌شود، اما رایج ترین نوع آن پکیج سه پایه TO-92 است که دقیقاً شبیه یک ترانزیستور است.



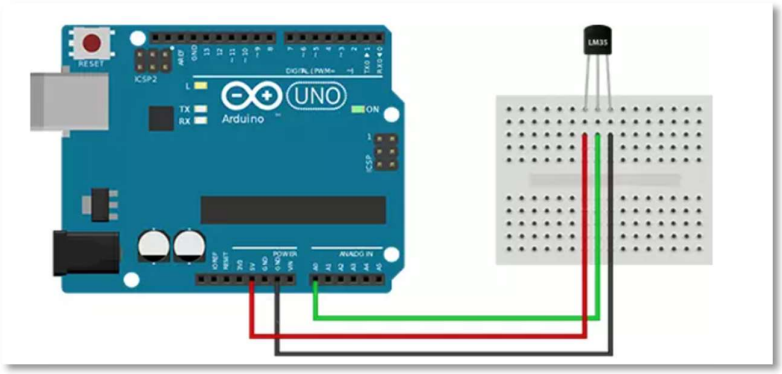

*(شکل ۵-۱۸) پایه های LM35*

پین VCC پایه‌ی تغذیه سنسور که میتواند به ۴ تا ۳۰ ولت متصل شود.

پین Vout این پایه یک ولتاژ آنالوگ تولید می‌کند که با دما رابطه‌ی مستقیم دارد و باید به پایه ورودی آنالوگ (ADC) متصل شود. پین GND پایه زمین است.

برای اتصال LM35 به آردوینو فقط کافی است سه پایه را به آن متصل کنید: دو پایه برای تغذیه و یک پایه برای خواندن مقدار سنسور.

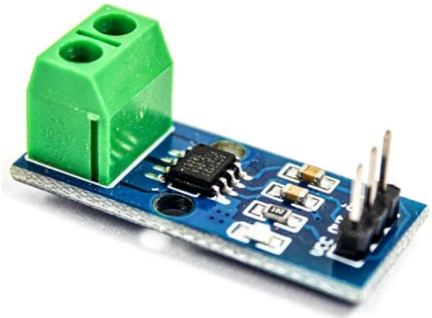
این سنسور را می‌توان با ۵ ولت تغذیه کرد. ولتاژ مثبت به "Vs+" و زمین به "GND" متصل می‌شود. پایه میانی "Vout" خروجی سیگنال آنالوگ سنسور است و به پایه ورودی آنالوگ (A0)یک آردوینو متصل می‌گردد.




*(شکل ۵-۱۹) اتصال با آردوینو*

## ۵-۹- راه اندازی ماژول سنسور جریان ACS712

ماژول سنسور جریان ACS712، یک قطعه الکترونیکی مبتنی بر سنسور جریان ACS712 است که با استفاده از اصل پدیده ای به نام اثر هال، جریان الکتریکی AC یا DC را با دقت بالا تا حداکثر ۳۰ آمپر اندازه گیری می کند. این ماژول دارای ولتاژ تغذیه ۴,۵ الی ۵,۵ ولت و دارای خروجی آنالوگ می باشد و سیگنال آنالوگ خروجی از طریق پایه OUT روی ماژول قابل دسترس است. در این ماژول با تغییر جریان ورودی، ولتاژ خروجی به صورت خطی تغییر می‌کند و در حالتی که جریانی از ماژول عبور نکند، ولتاژ خروجی برابر ۲,۵ ولت خواهد بود. همچنین حساسیت این ماژول جریان ۶۶ میلی ولت بر آمپر می باشد، به عبارتی به ازای هر یک آمپر افزایش جریان ورودی، ولتاژ خروجی ۶۶ میلی ولت افزایش خواهد یافت. این ماژول برای صنایع مختلفی مانند سیستم های کنترلی، صنایع خودرو، ماشین آلات و سیستم های حفاظتی و امنیتی به منظور کنترل موتور ها، مدار های رگولاتور ولتاژ، مدار های کنترل شارژ باتری و دیگر مدار هایی که نیاز به اندازه گیری جریان دارند، مورد استفاده قرار می گیرد.



*(شکل ۵-۲۰) ماژول سنسور جریان ACS712*

ماژول ACS712در نسخه‌های مختلفی مانند ACS712-05B ، ACS712-20A و-ACS712 30A وجود دارد. هر نسخه به توانایی اندازه‌گیری جریان‌های مختلف بستگی دارد.

ACS712-05B برای اندازه‌گیری جریان‌های تا ۵ آمپر، ACS712-20A برای جریان‌های تا ۲۰ آمپر و ACS712-30A برای جریان‌های تا ۳۰ آمپر مناسب است. این ماژول‌ها دارای واسطهای ورودی و خروجی آنالوگ (مانند ولتاژ تغذیه، ولتاژ خروجی جریان و گراند) هستند که به سادگی می‌توان آنها را به میکروکنترلرها و سایر سیستم‌های الکترونیکی متصل کرد.

به طور کلی، ماژول ACS712 یک سنسور جریان الکتریکی است که بر اساس اصل اثر هال کار می‌کند. اصل اثر هال به این شکل است که وقتی یک جریان الکتریکی از یک تراشه عبور می‌کند، یک ولتاژ خروجی متناسب با شدت جریان ایجاد می‌شود. در مورد ماژول ACS712، جریان الکتریکی ورودی از طریق یک سیم گذرانده می‌شود. ماژول دارای یک چیپ اصلی است

که از اصل اثر هال استفاده می‌کند تا جریان را اندازه‌گیری کند. ولتاژ خروجی ماژول متناسب با شدت جریان ورودی است و می‌توان آن را با استفاده از یک پین آنالوگ میکروکنترلر یا سایر سیستم‌های الکترونیکی خواند.

ویژگی ها و قابلیت های سنسور ACS712

قابلیت اندازه‌گیری جریان AC و DC: ماژول ACS712 قادر است به طور همزمان جریان‌های مستقیم (DC) و جریان‌های متناوب (AC) را اندازه‌گیری کند.

واسط آنالوگ: ماژول دارای واسطهای ورودی و خروجی آنالوگ است که به راحتی می‌توان آنها را به میکروکنترلرها و سایر سیستم‌های الکترونیکی متصل کرد. ولتاژ خروجی ماژول مستقیماً متناسب با شدت جریان ورودی است و می‌توان آن را با استفاده از یک پین آنالوگ میکروکنترلر یا سایر سیستم‌های الکترونیکی خواند.

دقت بالا: ماژول ACS712 دارای دقت بالای اندازه‌گیری جریان است. بسته به نوع نسخه، میزان دقت اندازه‌گیری ممکن است متفاوت باشد.

عملکرد غیرتماسی: یکی از مزیت‌های ماژول ACS712، عدم نیاز به مستقیم برخورد با جریان است. این به معنای این است که ماژول بدون تماس مستقیم با سیم جریان، قادر به اندازه‌گیری جریان الکتریکی است.

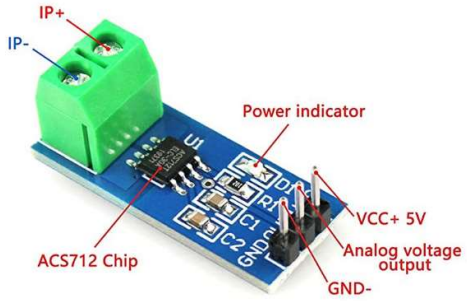
حفاظت در برابر نویز: ماژول ACS712 دارای قابلیت حفاظت در برابر نویز الکترومغناطیسی و تداخلات محیطی است. این ویژگی مهم برای دقت و استحکام اندازه‌گیری جریان است.

کاربرد های ماژول ACS712

از ماژول ACS712 برای کنترل و اندازه گیری جریان در دستگاه‌های الکترونیکی، رباتیک، سیستم‌های خودرویی و سایر برنامه‌های الکترونیکی استفاده می‌شود. برخی از کاربردهای رایج آن عبارتند از:

- کنترل و مانیتورینگ جریان در سیستم‌های الکترونیکی و صنعتی.
- اندازه‌گیری جریان در دستگاه‌های رباتیک و خودروها.
- استفاده در سیستم‌های حفاظتی و ایمنی برای اندازه‌گیری جریان اضافی یا ناگهانی.
- استفاده در سیستم‌های مانیتورینگ مصرف انرژی و کنترل توان.
- اندازه‌گیری جریان در سیستم‌های تغذیه و توزیع برق.

پین‌های ماژول اندازه‌گیری جریان ACS712




*(شکل ۵-۲۱) پایه های ACS712*

پایه VCC : این پایه به منبع تغذیه ماژول متصل می‌شود و ولتاژ تغذیه مورد نیاز ماژول را فراهم می‌کند. معمولاً با ولتاژ ۵ ولت یا ۳،۳ ولت سیستم الکترونیکی سازگار است.

پایه GND : این پایه به میزان صفر ولتاژ تغذیه متصل می‌شود و به عنوان نقطه مشترک برای سیستم الکترونیکی عمل می‌کند.

پایه OUT : این پایه ولتاژ خروجی را تولید می‌کند که متناسب با شدت جریان ورودی است. می‌توان از این پایه برای اتصال به پین آنالوگ یک میکروکنترلر یا سایر سیستم‌های الکترونیکی استفاده کرد تا جریان را اندازه‌گیری کند.

پایه NC : این پایه به منظور افزودن پایه‌های اضافی در برخی نسخه‌های ماژول ACS712 استفاده می‌شود. این پایه‌ها معمولاً به عنوان پایه‌های خالی یا عدم اتصال استفاده می‌شوند و در برخی نسخه‌ها حضور دارند.

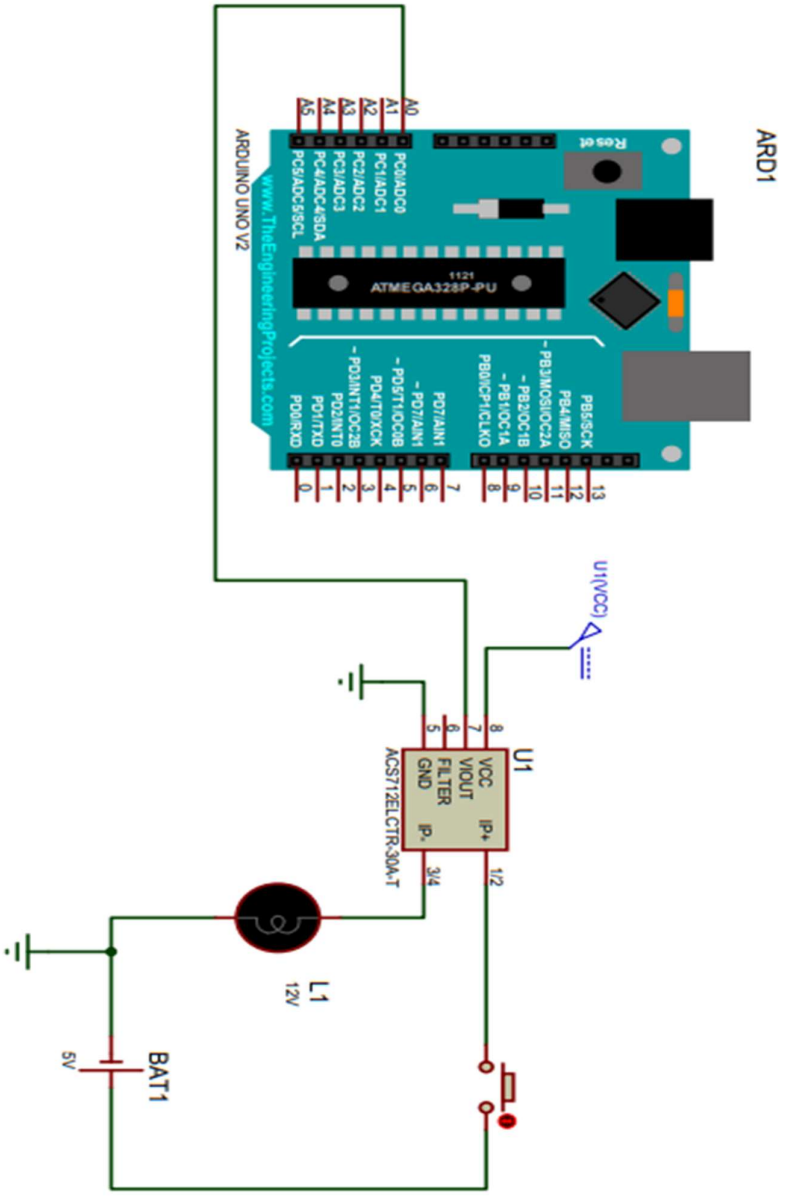
پایه VIOUT و VREF: این پایه‌ها در برخی نسخه‌های ACS712 برای تنظیم خروجی و ولتاژ مرجع استفاده می‌شوند. این پایه‌ها به طور معمول برای کاربردهای خاص و تنظیمات پیشرفته استفاده می‌شوند و در نسخه‌های ساده‌تر این ماژول‌ها حضور ندارند.

راه اندازی ماژول ACS712 با استفاده از آردوینو :

```
int analogPin = A0;                    // تعیین پین آنالوگ برای خواندن واتاژ
const int averageValue = 500;          // تعداد میانگین گیری برای بهبود دقت خواندن
long int sensorValue = 0;              // متغیر برای دخیره مقدار خوانده شده از حسگر
float voltage = 0;
float current = 0;
                                       // متغیر برای دخیره ولتاژ و جریان محاسبه شده
```

```
void setup() {
  Serial.begin(9600);        //شروع ارتباط سریال با سرعت ۹۶۰۰ بیت بر ثانیه
}

void loop() {
  for (int i = 0; i < averageValue; i++){
//حلقه برای میانگین گیری از مقادیر خوانده شده از حسگر
    sensorValue += analogRead(analogPin);
//افزون مقدار خوانده شده به مجموعه مقادیر
    delay(200); //تاخیر ۲۰۰ میلی ثانیه برای جلوگیری از خواندن سریع مقادیر
  }

  sensorValue = sensorValue / averageValue;
//  محاسبه میانگین مقدار خوانده شده
  voltage = sensorValue * 5.0 / 1024.0;
//  محاسبه ولتاژ معادل با مقدار خوانده شده از حسگر
  current = (voltage - 2.5) / 0.185;  //محاسبه جریان بر اساس ولتاژ

  Serial.print("مقدار ADC: ");  //نمایش مقدار خوانده شده از حسگر
  Serial.print(sensorValue);
  Serial.print("ولتاژ ADC: ");  //  نمایش ولتاژ معادل با مقدار خوانده شده
  Serial.print(voltag);
  Serial.print("V");
  Serial.print("جریان: ");  //  نمایش جریان محاسبه شده
  Serial.print(current);
  Serial.print("A");
}
```

شبیه سازی در نرم افزار پروتئوس

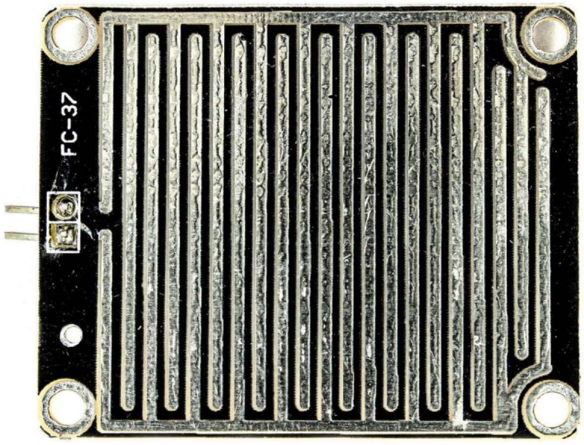



*(شکل ۵-۲۲) سیم کشی در نرم افزار*

## ۵-۱۰- راه اندازی ماژول رطوبت

ماژول تشخیص باران معمولاً از یک برد مدار چاپی (PCB) با سنسور باران، یک مدار پردازش سیگنال و مجموعه ای از پایه های خروجی تشکیل شده است. سنسور باران معمولاً از دو عنصر رسانا ساخته می‌شود که توسط یک ماده عایق از هم جدا شده اند. هنگامی که باران روی سطح سنسور می‌بارد، یک مسیر رسانا بین دو عنصر ایجاد می‌کند که توسط مدار پردازش سیگنال شناسایی می‌شود. پین های خروجی ماژول سیگنال دیجیتالی را ارائه می دهند که نشان می‌دهد باران می بارد یا خیر.

ماژول تشخیص باران معمولاً در طیف وسیعی از کاربردها مانند سیستم های جمع آوری خودکار آب باران، سیستم های آبیاری، ایستگاه های نظارت بر آب و هوا و سیستم های اتوماسیون خانه هوشمند استفاده می شود. این یک ماژول مقرون به صرفه و با استفاده آسان است که می تواند تشخیص دقیق باران را با حداقل تنظیم و کالیبراسیون ارائه دهد.




*(شکل ۵-۲۳)*

این ماژول از دو قطعه تشکیل شده است:

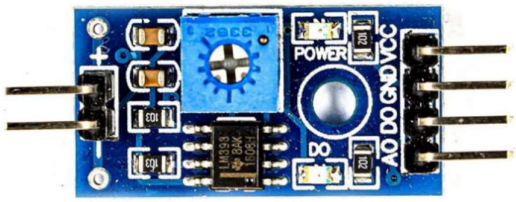
پد حسگر

سنسور باران، دارای یک پد حسگر همراه با مجموعه‌ای از مسیرهای مسی روباز است که در محل‌های باز، روی پشت بام یا محلی که بتواند در معرض بارندگی باشد، قرار داده می‌شود. این مسیرهای مسی معمولا بهم متصل نیستند، اما به‌وسیله قطرات آب به یکدیگر وصل می‌شوند.

ماژول الکترونیکی

علاوه بر پد حسگر، سنسور باران دارای یک ماژول الکترونیکی نیز است که پد حسگر را به آردوینو متصل می‌کند. این ماژول مطابق با مقاومت پد حسگر، یک ولتاژ خروجی تولید می‌کند و این

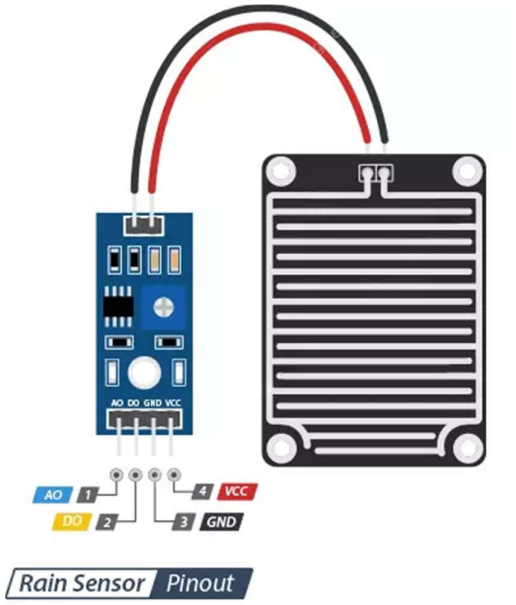
ولتاژ را از طریق پایه خروجی آنالوگ ماژول (AO) مستقیما در دسترس قرار می‌دهد که به‌وسیله آن می توانید شدت بارندگی را تشخیص دهید.
این ولتاژ به مقایسه کننده LM393 نیز فرستاده می‌شود. این مقایسه کننده ولتاژ را به سیگنال دیجیتال تبدیل می‌کند و از طریق پایه خروجی دیجیتال، آن را در دسترس قرار می‌دهد. با خروجی دیجیتال می‌توانید وجود یا عدم وجود باران را تشخیص دهید.




*(شکل ۵-۲۴)*

این ماژول دارای یک پتانسیومتر داخلی برای تنظیم حساسیت خروجی دیجیتال (DO) است. شما با استفاده از این پتانسیومتر می‌توانید یک حد آستانه تنظیم کنید. با این کار، هر زمان که مقدار آب از حد آستانه بیشتر شود، خروجی ماژولLOW می‌شود. اما اگر میزان آب از این حد فراتر نرود، خروجی ماژول HIGH خواهد بود.
در کنار پتانسیومتر، این ماژول دو LED نیز دارد. یکی از آن‌ها، LED تغذیه است که وقتی ماژول به تغذیه وصل باشد، روشن می‌شود LED دیگر وضعیت را نشان می‌دهد که وقتی خروجی دیجیتال LOW شود، روشن خواهد شد.

پین های ماژول سنسور تشخیص باران
A0 (خروجی سیگنال آنالوگ ): این پایه سیگنال آنالوگی بین ۰ تا ۵ ولت به شما می‌دهد.
D0 (خروجی دیجیتال): این پایه، خروجی دیجیتال مدار مقایسه کننده داخلی را می‌دهد. این پایه را می‌توانید به هر کدام از پایه‌های دیجیتال آردوینو یا مستقیما به یک رله ۵ ولت یا تجهیزات مشابه متصل کنید.
GND: این پایه به زمین مدار وصل می‌شود.
VCC: این پایه، تغذیه سنسور را فراهم می‌کند. ولتاژ این پایه بین ۳,۳ ولت تا ۵ ولت است.
لطفا به این نکته توجه داشته باشید که خروجی آنالوگ با توجه به ولتاژ تغذیه تغییر خواهد کرد.




*(شکل ۵-۲۵)*

کالیبره کردن ماژول سنسور باران

برای اینکه بتوانید اطلاعات صحیحی از سنسور باران دریافت کنید، توصیه می‌شود که ابتدا آن را کالیبره کرد. بر روی این سنسور یک پتانسیومتر برای کالیبره کردن خروجی دیجیتال وجود دارد. با چرخاندن پیچ این پتانسیومتر، شما می‌توانید حد آستانه ولتاژ را تنظیم کنید. بنابراین، زمانی‌که آب از حد آستانه مشخص شده، بیشتر شود، LED وضعیت روشن می‌شود و خروجی دیجیتال LOW خواهد شد.

اما برای کالیبره کردن این سنسور، ابتدا کمی آبی بر روی پد حسگر بپاشید و پتانسیومتر را به صورت ساعتگرد بچرخانید تا LED وضعیت روشن شود. سپس، پتانسیومتر را به‌صورت پادساعتگرد بچرخانید تا LED خاموش شود.

راه اندازی ماژول ACS712 با استفاده از آردوینو:

```
#define sensorPower 7 //تعریف پایه های حسگر
#define sensorPin 8

void setup() {
    pinMode(sensorPower, OUTPUT); // تعیین نوع پایه حسگر به عنوان خروجی

    digitalWrite(sensorPower, LOW);     // ابتدا حسگر را خاموش کن


    Serial.begin(9600); // شروع ارتباط سریال با نرخ 9600 بیت بر ثانیه
}
void loop() {
    int val = readSensor();    // مقدار حاصل از تابع زیر را بخوان و چاپ کن

    Serial.print("Digital Output: ");
    Serial.println(val);

    if (val) {     // تشخیص وضعیت باران

        Serial.println("Status: Clear");
    } else {
        Serial.println("Status: It's raining");
    }

    delay(1000);    // هر ثانیه یک بار اطلاعات حسگر را بخوان
    Serial.println();
}

int readSensor() { //  این تابع مقدار خروجی حسگر را برمی‌گرداند

    digitalWrite(sensorPower, HIGH);    // حسگر را روشن کن
    delay(10);                          // صبر کن تا برق ثابت شود
    int val = digitalRead(sensorPin);   // مقدار خروجی حسگر را بخوان
    digitalWrite(sensorPower, LOW);     // حسگر را خاموش کن
    return val;                         // مقدار را برگردان
}
```

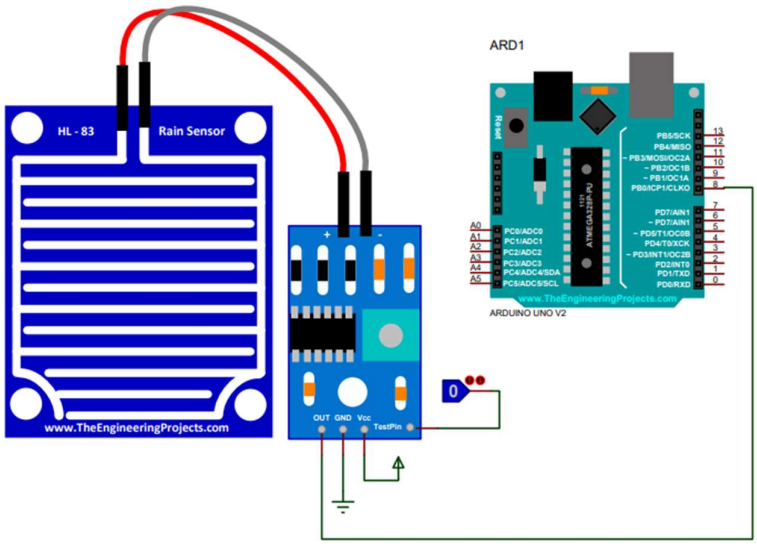
توضیحات کد:

در بخش Setup ، شما باید ابتدا پایه‌های تغذیه سنسور را به عنوان خروجی تعریف کنید و آن را LOW قرار دهید تا در ابتدای کار، سنسور تغذیه نشود. علاوه بر این، شما باید در این بخش سریال مانیتور را نیز تنظیم کرد.

در بخش loop دستورات، باید تابع readSensor() را در فاصله زمانی ۱ ثانیه فراخوانی کرد و مقدار بازگشتی را همراه با وضعیت هوا چاپ کرد.

تابع readSensor() مقدار خروجی دیجیتال سنسور باران را می‌دهد. این تابع سنسور را روشن می‌کند، ۱۰میلی ثانیه صبر می‌کند، مقدار دیجیتال حسگر را می‌خواند، حسگر را خاموش می‌کند و سپس نتیجه را برمی‌گرداند.

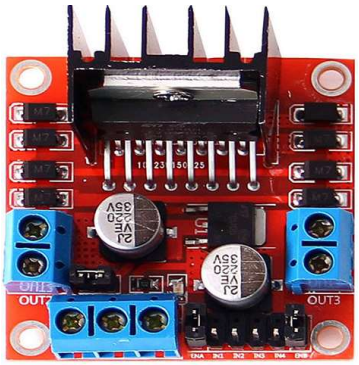
شبیه سازی در نرم افزار پروتئوس




*(شکل ۵-۲۶) اتصالات با آردوینو*

## ۵-۱۱- راه اندازی درایور موتور L298n

یکی از آسان ترین و ارزان ترین روش های کنترل موتورهای DC ، راه اندازی ماژول L298n درایور موتور با آردوینو است. این ماژول می تواند همزمان سرعت و جهت دوران دو موتور DC را کنترل کند.



*(شکل ۵-۲۷) درایور موتور L298n*

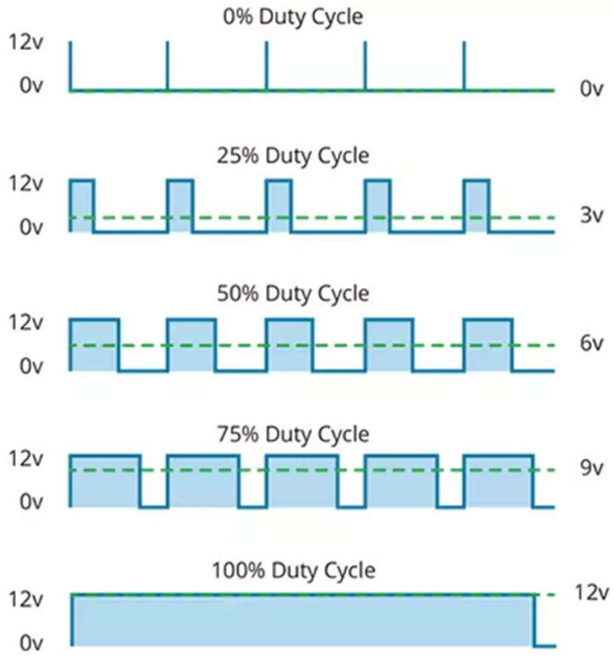
کنترل موتور DC :

برای کنترل کامل یک موتور DC لازم است که هم سرعت و هم جهت دوران آن را کنترل کرد. این نوع کنترل از ترکیب دو تکنیک زیر قابل دستیابی است:

۱- تکنیک PWM : برای کنترل سرعت

۲- تکنیک H-Bridge : برای کنترل جهت چرخش موتور

کنترل سرعت یک موتور DC از طریق تغییر ولتاژ تغذیه dc موتور امکان پذیر است. یک تکنیک مرسوم برای این کار استفاده از روش مدولاسیون عرض پالس (PWM) می‌باشد. در این تکنیک مقدار متوسط ولتاژ ورودی از طریق ارسال یکسری پالس های صفر و یک(ON-OFF)تنظیم می‌شود. مقدار متوسط ولتاژ، متناسب با عرض پالس که duty cycle نامیده می شود، خواهد بود. با افزایش duty cycle مقدار متوسط ولتاژ تغذیه اعمال شده به موتور DCبیشتر خواهد شد و در نتیجه سرعت دوران شافت موتور افزایش خواهد یافت. اما کم کردن duty cycle، سبب کاهش متوسط ولتاژ تغذیه و در نتیجه کاهش سرعت دور موتور می‌شود. در شکل زیر تکنیک PWM و ارتباط مقادیر مختلف duty cycle با مقدار متوسط ولتاژ تغذیه نشان داده شده است:



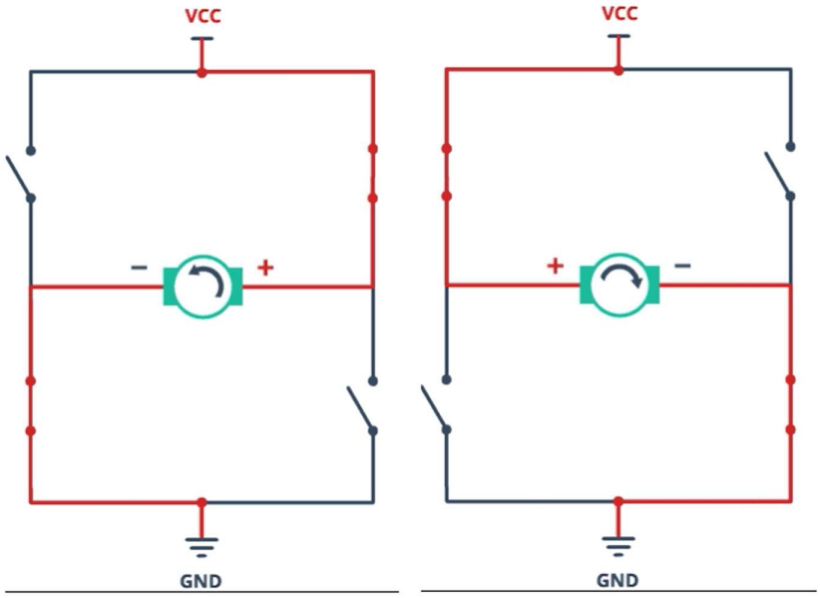

*(شکل ۵-۲۸)*

تکنیک H-Bridge برای کنترل جهت چرخش موتور :

کنترل جهت چرخش شافت یک موتور DC از طریق تغییر پلاریته ولتاژ تغذیه آن امکان پذیر است.

تکنیک مرسوم برای این کار استفاده از H-Bridge یا مدار پل H است. مدار پل H متشکل از ۴ عدد سوئیچ می‌باشد که موتور در مرکز آن‌ها قرار می‌گیرد و بدین ترتیب ساختاری شبیه H تشکیل می‌دهد. با بسته شدن همزمان دو سوئیچ خاص می‌توان پلاریته ولتاژ تغذیه اعمالی به موتور و در نتیجه جهت چرخش شافت موتور را تغییر داد.

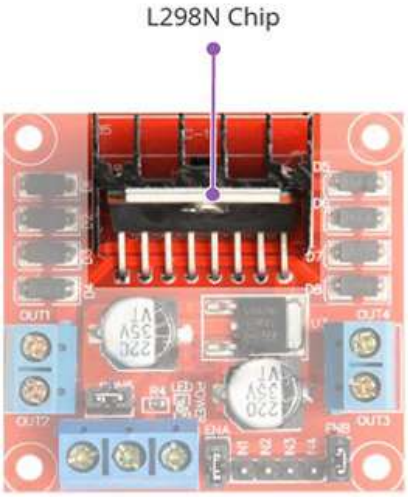
در تصویر زیر نحوه عملکرد مدار پل H نشان داده شده است:




*(شکل ۵-۲۹) عملکرد پل اچ*

آی سی درایور موتور L298n :

L298 بصورت یک چیپ بزرگ مشکی رنگ متصل به یک هیت سینک در مرکز ماژول قرار دارد. این چیپ یک درایور موتور H-bridge دو کاناله است و با استفاده از آن می‌توان دو عدد موتور DC را بصورت مستقل از یکدیگر کنترل نمود. لذا، برای بکارگیری در ربات‌های دوچرخ ایده آل می‌باشد.

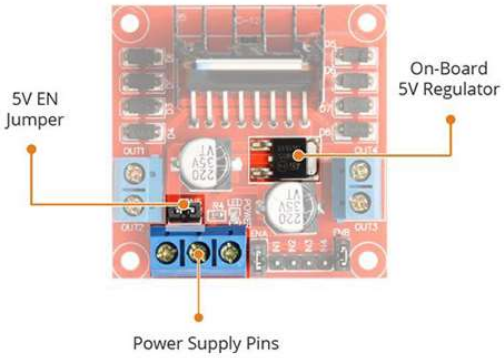



*(شکل ۵-۳۰) آی سی L298n*

منبع تغذیه :

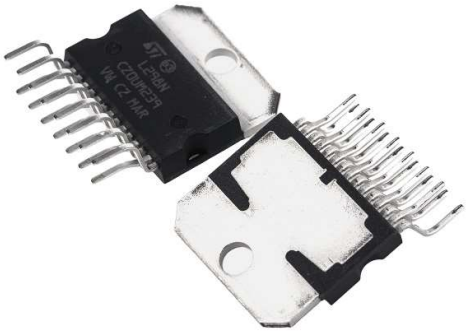
ماژول درایور L298 بوسیله یک ترمینال ۳,۵ میلیمتری سه پین تغذیه می‌شود. یک پین برای تغذیه موتور(Vs) ، یک پین به عنوان زمین (GND) و پین سوم تغذیه ۵ ولت مدارات منطقی (Vss) می‌باشد.

درایور L298 در حقیقت دارای دو پین تغذیه ورودی "Vss" و "Vs" است. از طریق پینVs ولتاژ الکتریکی مورد نیاز برای درایو موتور به مدار پل H داده می‌شود که بین ۵ تا ۳۵ ولت می تواند تغییر کند. برای تغذیه مدارات منطقی استفاده می‌شود که اندازه آن می‌تواند بین ۵ تا ۷ ولت متغیر باشد. پایه GND به عنوان زمین مشترک Vs و Vss می‌باشد.



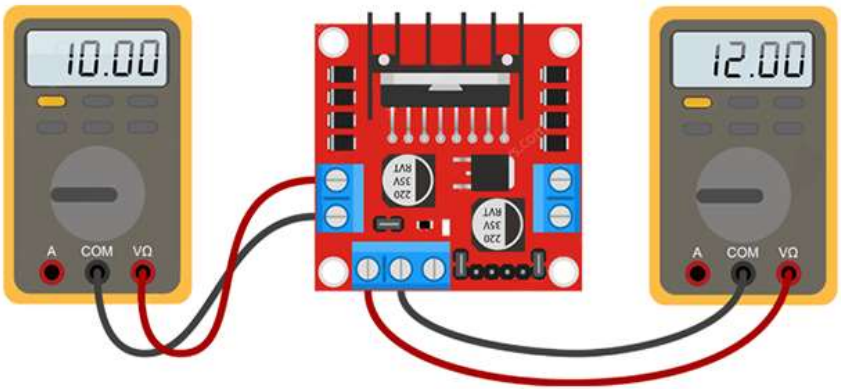

*(شکل ۵-۳۱)*

ماژول L298 دارای یک رگولاتور ۷۸ M05 پنج ولتی STMicroelectronics است که از طریق یک جامپر می‌توان آن را فعال یا غیر فعال کرد. وقتی جامپر در محل خود قرار داشته باشد، رگولاتور پنج ولتی فعال خواهد بود و ولتاژ تغذیه Vss را از ولتاژ تغذیه موتور یعنی Vs تامین می‌کند. در چنین حالتی، پین ۵ ولت ترمینال ورودی به عنوان یک خروجی ۵ ولتی نیم آمپر عمل خواهد نمود که از آن می‌توان برای تغذیه آردوینو و یا هر مدار دیگری که تغذیه ۵ ولتی نیاز دارد، استفاده نمود. برداشتن جامپر سبب غیر فعال شدن رگولاتور ۵ ولتی می‌شود و مجبور خواهیم بود ولتاژ ۵ ولتی جداگانه‌ای از طریق ترمینال ورودی تامین کنیم.



*(شکل ۵-۳۲)*

افت ولتاژ L298n :



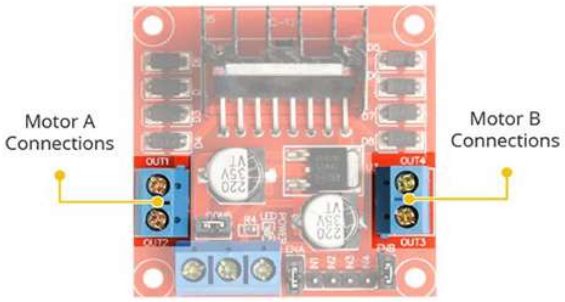

*(شکل ۵-۳۳) تصویر مربوط به افت ولتاژ*

افت ولتاژ درایور L298 در حدود ۲ ولت است که بخاطر افت ولتاژ داخلی ترانزیستورهای سوئیچینگ مدار پل H می‌باشد. بنابراین اگر تغذیه ۱۲ ولتی به ترمینال ورودی ماژول اعمال شود، موتور DC ولتاژی در حدود ۱۰ ولت دریافت می‌کند. لذا یک موتور DC دوازده ولتی

نمی تواند با حداکثر سرعت نامی خود کار کند. به همین دلیل برای دستیابی به حداکثر سرعت نامی موتور، لازم است ولتاژ تغذیه موتور که به ترمینال ورودی ماژول داده می‌شود، حدود ۲ ولت بیشتر از حداکثر ولتاژ نامی موتور در نظر گرفته شود.

بنابراین با در نظر گرفتن افت ولتاژ ۲ ولت، برای یک موتورDC پنج ولتی باید ولتاژ تغذیه ۷ ولت به ترمینال ورودی ماژول درایور داده شود. به همین ترتیب برای یک موتور ۱۲ ولتی باید تغذیه ترمینال ورودی ۱۴ ولت در نظر گرفته شود.

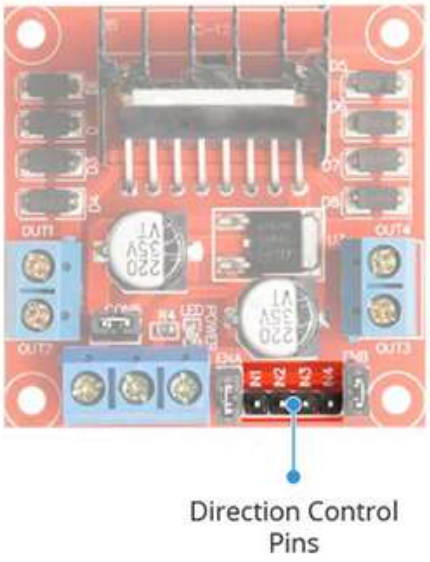
ترمینال های خروجی :




*(شکل ۵-۳۴)*

کانال‌های خروجی درایور برای دو موتور A و B به دو ترمینال ۳,۵ میلیمتری در طرفین ماژول L298 متصل شده‌اند. به این دو ترمینال می‌توان دو موتور الکتریکی مستقل با ولتاژ کاری ۵ تا ۳۵ ولت وصل نمود. هر ترمینال خروجی می‌تواند تا ۲ آمپر جریان برای موتور الکتریکی تامین نماید. هرچند مقدار جریان اعمالی، به منبع تغذیه سیستم بستگی دارد.

پین های کنترلی در ماژول درایور موتور L298 :




*(شکل ۵-۳۵)*

برای هر یک از کانال های ماژول L298 دو نوع پین کنترلی وجود دارد که با استفاده از آن می‌توان به صورت همزمان سرعت و جهت دوران موتور را کنترل نمود:

- پین های کنترل جهت دوران موتورها
- پین های کنترل سرعت موتورها

با استفاده از پین‌های کنترل جهت می‌توان نحوه دوران شافت موتور DC را در دو جهت مستقیم یا عقبگرد کنترل نمود. این پین‌ها در حقیقت سوئیچ‌های مدار پل H آی سیL298 را کنترل می‌کنند. برای هر کانال خروجی دو پین کنترل جهت وجود دارد. پین‌های IN1 و IN2 جهت دوران موتور A و پین‌های IN3 و IN4 جهت دوران موتور B را کنترل می‌کنند. کنترل جهت دوران از طریق اعمال منطق High (۵ ولت) و یا منطق Low (صفر ولت) طبق چارت زیر می‌باشد.

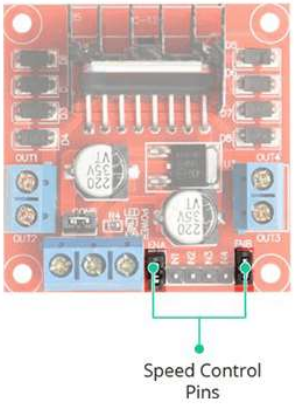
| ورودی ۱ | ورودی ۲ | جهت دوران موتور |
|---|---|---|
| LOW (صفر ولت) | LOW (صفر ولت) | موتور خاموش |
| HIGH (۵ ولت) | LOW (صفر ولت) | دوران در جهت مستقیم |
| LOW (صفر ولت) | HIGH (۵ ولت) | دوران در جهت معکوس |
| HIGH (۵ ولت) | HIGH (۵ ولت) | موتور خاموش |

*(شکل ۵-۳۶)*

پین های کنترل سرعت :

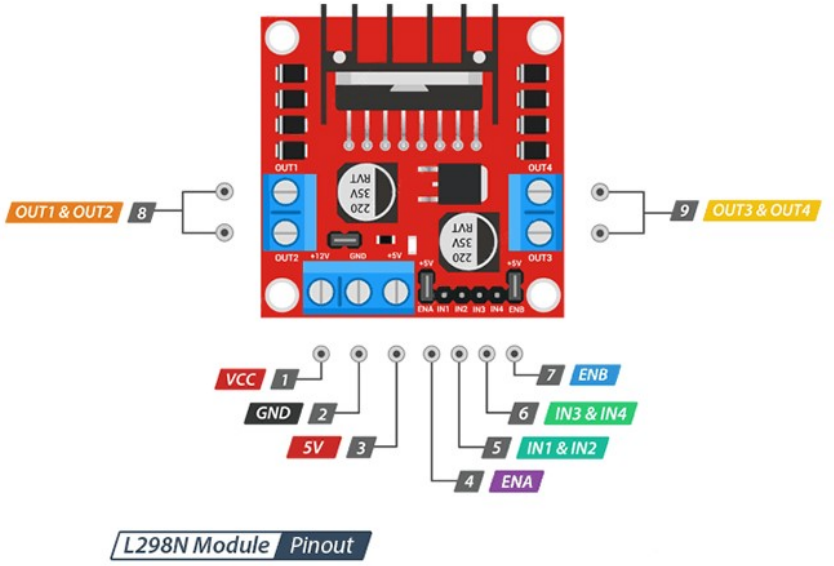
دو پین کنترل سرعت ENA و ENB برای روشن و خاموش کردن موتور و همچنین کنترل سرعت دوران موتورهای A و B بکار گرفته می‌شوند. اعمال منطق High (۵ ولت) سبب روشن شدن موتور و اعمال منطق Low باعث متوقف شدن دوران موتور می‌شود. کنترل تعداد دور از طریق اعمال پالس PWM انجام پذیر می‌باشد.

ماژول درایور L298 معمولا دارای جامپر بر روی پین‌های کنترل سرعت است. هنگامی که جامپر در محل خود قرار دارد، موتور فعال و با حداکثر سرعت دورانی خود کار می‌کند. چنانچه قصد داشته باشید با استفاده از یک برنامه سرعت موتور را کنترل کنید لازم است جامپر از روی پین کنترل سرعت برداشته شود و به پین‌های PWM مربوطه آردوینو متصل شود.




*(شکل ۵-۳۷)*

پین های ماژول درایور موتور L298 :




(شکل ۵-۳۸)

۱-پین VCC :

تغذیه موتور را تامین می کند که مقدار ولتاژ آن می‌تواند بین ۵ تا ۳۵ ولت باشد. یادآوری می‌شود که اگر جامپر 5V در محل خود قرار داشته باشد، برای دستیابی به حداکثر سرعت موتور لازم است مقدار ولتاژ تغذیه به اندازه ۲ ولت بیشتر از ولتاژ نامی موتور اعمال شود.

۲-پین GND : پین زمین مشترک است.

۳-پین 5V :

ولتاژ مورد نیاز مدارات منطقی سوئیچینگ داخلی آی سی L298 را تامین می‌کند. اگر جامپر 5V در محل خود قرار داشته باشد، این پین به عنوان یک پین خروجی عمل می‌کند و از آن می‌توان برای تغذیه آردوینو استفاده نمود. اگر این جامپر برداشته شود، نیاز است که این پین به پین 5V آردوینو متصل شود.

۴-پین ENA :

برای کنترل سرعت موتور A استفاده می‌شود. اعمال منطق High به این پین (یا بطور معادل نگه داشتن جامپر در سر جای خود) سبب شروع به کار کردن موتور A و برداشتن جامپر سبب متوقف شدن موتور خواهد شد. برداشتن جامپر و اتصال پین به ورودی PWM امکان کنترل سرعت موتور A را فراهم خواهد کرد.

۵-پین های IN1 و IN2 :

برای کنترل جهت دوران موتور A استفاده می شوند. وقتی به یکی از این پین‌ها منطق High (۵ ولت) و به دیگری منطق Low (صفر ولت) اعمال شود، شافت موتور در یک جهت شروع به چرخش می‌کند. اگر به هر دوی این پین‌ها منطق High و یا به هر دو منطق Low اعمال شود، موتور A متوقف خواهد شد.

۶-پین های IN3 و IN4 :

برای کنترل جهت دوران موتور B استفاده می شوند. وقتی به یکی از این پین‌ها منطقHigh (۵ ولت) و به دیگری منطق Low (صفر ولت) اعمال شود، شافت موتور B در یک جهت شروع به چرخش می‌کند. اگر به هر دوی این پین‌ها منطق High و یا به هر دو منطق Low اعمال شود، موتور متوقف خواهد شد.

۷-پین ENB :

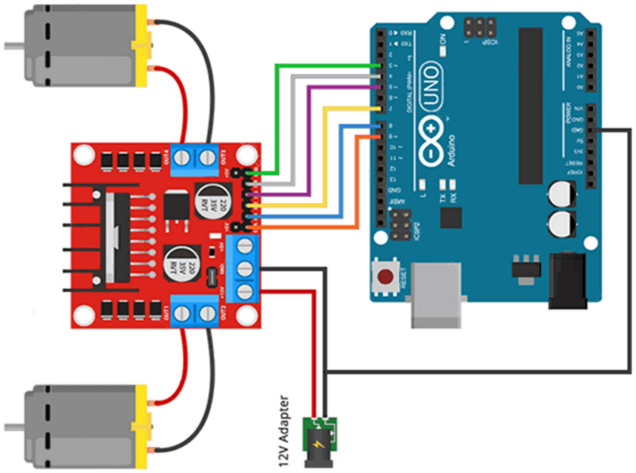
برای کنترل سرعت موتورB استفاده می‌شود. اعمال منطق High به این پین (یا بطور معادل نگه داشتن جامپر در سر جای خود) سبب شروع به کار کردن موتور B و برداشتن جامپر سبب متوقف شدن موتور خواهد شد. برداشتن جامپر و اتصال پین به ورودی PWM امکان کنترل سرعت موتور B را فراهم خواهد کرد.

۸-پین های خروجی OUT1 و OUT2 به موتورA متصل می‌شوند.

۹-پین های خروجی OUT3 و OUT4 به موتور B متصل می‌شوند.

راه اندازی ماژول درایور موتور L298 با آردوینو UNO :

برای این کار در ابتدا منبع تغذیه موتورهای DC را وصل می‌کنیم. در اینجا ما از موتورهای DC گیربکسی(موتورهایTT) که بطور معمول در ربات‌های دوچرخ بکارگرفته می‌شوند، استفاده می کنیم. ولتاژ این موتورها بین ۳ تا ۱۲ ولت است. بنابراین یک منبع تغذیه خارجی ۱۲ ولت به ترمینال VCC متصل می‌کنیم. با توجه به افت ولتاژ داخلی ماژول، موتورها حداکثر ۱۰ ولت دریافت خواهند نمود و حداکثر سرعت دور موتورها کمتر خواهد بود، ولی در این مثال مورد قبول است.




(شکل ۵-۳۹)

در گام بعد لازم است تغذیه ۵ ولتی مدارت منطقی L298 را تامین کنیم. به همین دلیل از رگولاتور داخلی ۵ولت ماژول درایور استفاده می‌شود و لذا جامپر رگولاتور را در سر جای خود قرار داده می‌شود. اکنون باید پین‌های ورودی و فعال‌ساز ماژول درایور L298 (یعنی پین‌های ENA، IN1، IN2، IN3، IN4وENB ) را به ۶ پین خروجی دیجیتال آردوینو (خروجی‌های ۹، ۸، ۷، ۵، ۴ و ۳) متصل کرد. به یاد داشته باشید که پین‌های شماره ۹ و شماره ۳ آردوینو مربوط به PWM هستند.

در مرحله آخر یک موتور به ترمینال A (Out1 و Out2 ) و موتور دیگر به ترمینال B (Out3 وOut4) وصل می شوند.

برنامه نویسی آردوینو (کنترل موتور DC):

در ابتدای کد مشخص شده است که پین‌های آردوینو به کدام یک از پین‌های ماژول درایور L298 متصل هستند.

```
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;
```

همه پین‌های کنترلی موتور به عنوان خروجی‌های دیجیتال تعریف می شوند و مقدار اولیه آن‌ها بصورت Low است می‌شود تا هر دو موتور خاموش شوند.

```
void setup()
{
    // Set all the motor control pins to outputs
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);

    // Turn off motors - Initial state
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
```

در ادامه، دو تابعی که قبلا توسط کاربر تعریف شده اند، درون یک حلقه نرم افزاری با فاصله زمان یک ثانیه (هزار میلی ثانیه) فراخوانی می شوند.

```
void loop()
{
    directionControl();
    delay(1000);
    speedControl();
    delay(1000);
}
```

این توابع عبارتند از:

directionControl(): این تابع هر دو موتور را به مدت ۲ ثانیه با حداکثر سرعت دوران در جهت مستقیم روشن می کند. سپس، به مدت دو ثانیه نیز در جهت معکوس موتورها را می چرخاند و در نهایت دو موتور را متوقف می کند.

```
void directionControl()
{
    // Set motors to maximum speed
    // For PWM maximum possible values are 0 to 255
    analogWrite(enA, 255);
    analogWrite(enB, 255);
    // Turn on motor A & B
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(2000);
```

```
    // Now change motor directions
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    delay(2000);
    // Turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
```

speedControl(): این تابع سرعت دوران هر دو موتور را از مقدار صفر تا حداکثر سرعت افزایش می‌دهد که این کار از طریق تولید سیگنال‌های PWM توسط تابع analogWrite() انجام می‌شود. پس از آن، مجددا سرعت دوران موتورها از مقدار حداکثر تا رسیدن به صفر کاهش می یابد و در نهایت دو موتور خاموش می شوند.

```
void speedControl() {
    // Turn on motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);

    // Accelerate from zero to maximum speed
    for (int i = 0; i < 256; i++) {
        analogWrite(enA, i);
        analogWrite(enB, i);
        delay(20);
    }
```

```
    // Decelerate from maximum speed to zero
    for (int i = 255; i >= 0; --i) {
        analogWrite(enA, i);
        analogWrite(enB, i);
        delay(20);
    }

    // Now turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
```

## ۵-۱۲- راه اندازی سروو موتور

سروموتور یک عملگر چرخشی یا خطی است که امکان کنترل دقیق زاویه یا حرکت خطی، سرعت و شتاب، با استفاده از آن امکان پذیر است. چنانچه بخواهید یک شیء را در یک زاویه خاص بچرخانید یا آن را در یک فاصله مشخص به صورت خطی حرکت دهید، می توانید از سروو موتور استفاده کنید.

*(شکل ۵-۴۰) انواع سروو موتور های صنعتی*


*(شکل ۵-۴۱) برخی از سروو موتور های رباتیک*

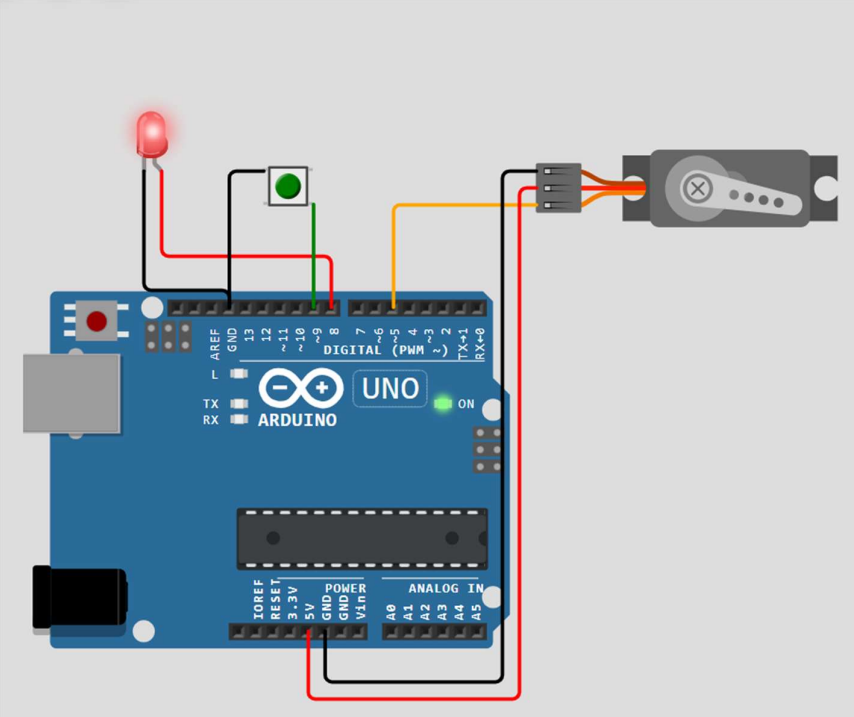
سروو های رباتیکی که در (شکل ۵-۴۱) مشخص است، دارای سه سیم قرمز، قهوه ای و نارنجی که به ترتیب Battery(+) ، Battery(-) و سیگنال یا همان پایه اطلاعات می باشند.

درقسمت زیر مثالی آورده شده است که به کمک یک push button، یک led روشن و همینطور سروو ۹۰ درجه به سمت راست می‌چرخد.

کد برنامه نویسی:

```
#include<Servo.h>   //کتابخانه مربوط به سروو
Servo myservo;      //اسم گذاری برای سروو

const int Button = 9;      //اسم گذاری برای پایه متصل به کلید
const int LED = 8;  //اسم گذاری برای پایه متصل به ال ای دی
int buttonState = 0;       // متغیری برای وضعیت کلید

void setup() {
  myservo.attach(5);       // پایه متصل به سروو
  myservo.write(10);       //مقدار اولیه برای سروو
  pinMode(LED, OUTPUT);
  pinMode(Button, INPUT);
}

void loop() {
  buttonState = digitalRead(Button);

  if (buttonState == HIGH) {
// turn LED on:
    digitalWrite(LED, HIGH);
    myservo.write(100);
  } else {
// turn LED off:
    digitalWrite(LED, LOW);
    myservo.write(0);
  }
}
```

نحوه اتصال با آردوینو و ال ای دی




*(شکل ۵-۴۲)*

## ۵-۱۳- خودآزمایی

۱. برنامه ای بنویسید که بر روی سگمنت از ۰ تا ۹۹۹را نشان دهد.

۲. با التراسونیک برنامه ای بنویسید که در حالت عادی ال ای دی سبز روشن، وقتی جسمی تا ۲۰ سانتی میبیند، ال ای دی قرمز روشن شود.

# ۶

# پروژه های آردوینو

## ۶-۱- اهداف فصل

- بررسی نحوه کار پروژه
- بررسی قطعات مورد نیاز برای آن
- شماتیکی از اتصالات
- کد نویسی و برنامه نویسی آنها

## ۶-۲- مقدمه

بعد از اتمام فصول قبل، وقت آن رسیده که با چند پروژه کاربردی آشنا شده و به نحوه آماده سازی تا اتصال آنها پرداخته شود.

## ۶-۳- ربات تعقیب کننده انسان

در سال های اخیر ما شاهد پیشرفت روزافزون ربات ها هستیم؛ که سبب به وجود آمدن ماشین های هوشمند شده اند و می توانند با محیط اطراف خود تعامل داشته باشند. یکی از این پیشرفت ها ربات تعقیب کننده انسان است .

این ربات ها می توانند انسان ها را شناسایی کنند و شخص را به صورت مستقل ردیابی و دنبال کنند ؛ که در کمک به افراد در مکان های مختلف به کار می آیند. به عنوان مثال چرخ دستی های خرید فروشگاه ها ، چمدان های تعقیب کننده ، کمک کننده های انسان در انبارهای کالا و... از مواردی هستند که این ربات به کمک انسان می آید.

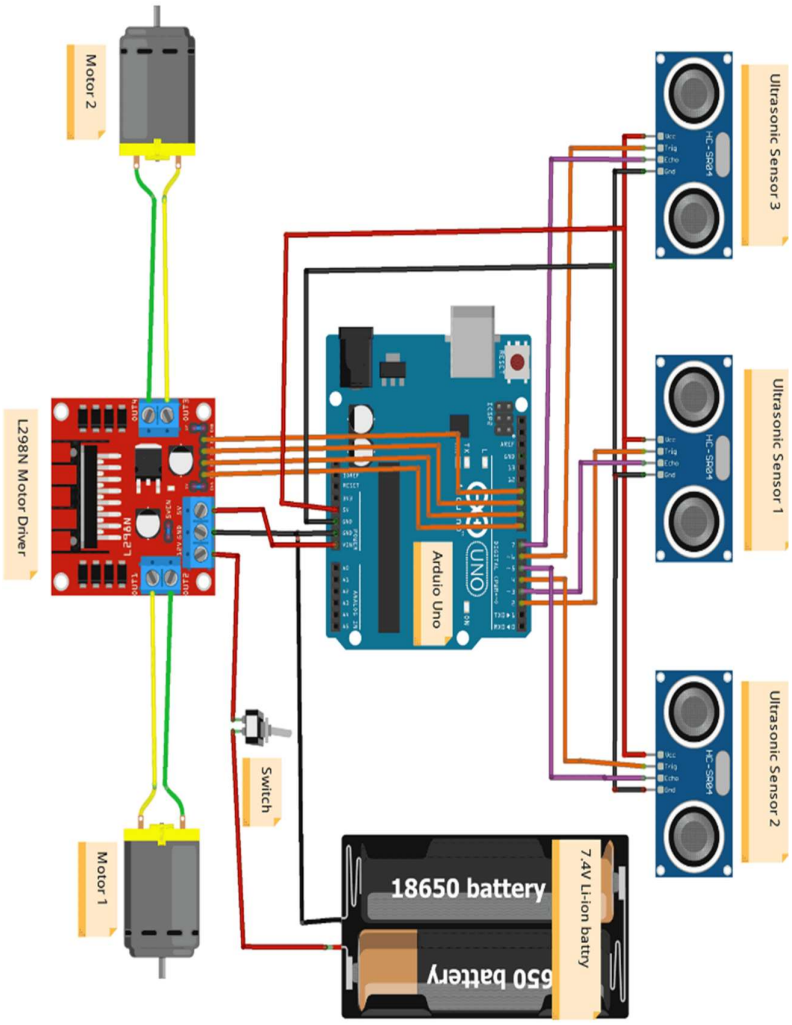
در این پروژه ربات تعقیب کننده ما از ۳ سنسور التراسونیک استفاده می کند و در فاصله سنجی و حرکت، دقت بالایی دارد. این سنسورها فاصله با انسان را اندازه گیری و بر اساس آن شروع به حرکت در جهت مورد نظر می کنند. در فصل دوم این کتاب به معرفی سنسور التراسونیک و ماژول درایور موتور و کار با آن پرداخت شده است.

قطعات مورد نیاز :

- بورد آردوینو UNO
- سنسور اولتراسونیک ۳ عدد
- درایور موتور L298n
- موتورهای گیربکس ۲ عدد
- چرخ ۲ عدد
- باتری لیتیوم یون ۳,۷ ولت ۲ عدد یا باتری معمولی ۴ عدد

- نگهدارنده باتری
- برد بورد
- سیم سوئیچ و جامپر

شماتیک اتصالات ربات تعقیب کننده




*(شکل ۶-۱) شماتیک اتصالات پروژه*

این مدار شامل سه سنسور اولتراسونیک است که امکان اندازه گیری فاصله را در سه جهت جلو، چپ و راست را می‌دهند. این سنسور ها از طریق پین های GPIO آردوینو به برد متصل می‌شود. همچنین این مدار شامل دو موتور DC است که به یک ماژول موتور درایورL298n متصل می‌شود. این ماژول نیز با استفاده از پین های دیجیتال آردوینو کنترل میشود. برای تغذیه کل قطعات، از دو باتری ۳,۷ ولتی لیتیوم یوتی استفاده میشود که با استفاده از یک سوئیچ به موتور درایور متصل هستند ؛ توجه داشته باشید که می‌توان در تغذیه مدار از هر نوع باتری استفاده کرد.

اتصالات بین ماژول سنسور اولتراسونیک و آردوینو :

پایه VCC هر سنسور اولتراسونیک را به پایه ۵ ولت برد آردوینو وصل کنید.

پایه GND هر سنسور اولتراسونیک را به پایه GND برد آردوینو وصل کنید.

پین ماشه (TRIG) هر سنسور اولتراسونیک را به پین های دیجیتال (۲،۴ و ۶) برد آردوینو وصل کنید.

پین اکو (ECHO) هر اولتراسونیک را به پین های دیجیتال (۳،۵ و ۷) برد آردوینو وصل کنید.

اتصالات بین آردوینو و ماژول درایور موتور:

پایه های دیجیتال آردوینو (۸، ۹، ۱۰ و ۱۱) را به پایه های ورودی (IN1، IN2، IN3، وIN4) در درایور موتور متصل کنید.

پین های ENA و ENB ماژول درایور موتور را با اتصالات مناسب High کنید.

پایه هایOUT1 ، OUT2، OUT3 و OUT4 ماژول درایور موتور را به موتورها وصل کنید.

پایه های (+5V) VCC و GND ماژول درایور موتور را به ولتاژ مناسب (Vin) و زمین (GND)در آردوینو وصل کنید.

منبع تغذیه :

ترمینال مثبت منبع تغذیه را به ورودی +۱۲ ولت ماژول درایور وصل کنید.

ترمینال منفی منبع تغذیه را به پایه GND ماژول درایور موتور وصل کنید.

پایه GND آردوینو را به پایه GND ماژول درایور موتور وصل کنید.

برنامه نویسی :

این کد فواصل سه سنسور اولتراسونیک ("frontDistance"، "leftDistance"و "rightDistance") را می خواند. سپس این فواصل را برای تعیین سنسور با کمترین فاصله مقایسه می کند. اگر فاصله زیر حد آستانه باشد، ربات را با استفاده از تابع های کنترل جهت موتور حرکت می دهد ('moveForward()', 'turnLeft()', 'turnRight()'). اگر هیچ یک از فاصله ها زیر حد آستانه نباشد، موتور را با استفاده از "stop()" متوقف می کند.

در ابتدا اتصالات پین سنسورهای اولتراسونیک و کنترل موتور را تعریف می کنیم. متغیرهای S1Trig، S2Trig، S3Trig نشان دهنده پایه های تریگر سه حسگر اولتراسونیک هستند و S1Echo ، S2Echo، S3Echo، نشان دهنده پین‌های اکو مربوط به سنسورهای التراسونیک هستند.

متغیرهای LEFT_MOTOR_PIN1 ، LEFT_MOTOR_PIN2، RIGHT_MOTOR_PIN1 و RIGHT_MOTOR_PIN2 پایه‌هایی را برای کنترل موتورها تعریف می‌کنند.

متغیرهای MAX_DISTANCE و MIN_DISTANCE_BACK آستانه تشخیص موانع را تعیین می کنند.

```
// Ultrasonic sensor pins
#define S1Trig 2
#define S2Trig 4
#define S3Trig 6
#define S1Echo 3
#define S2Echo 5
#define S3Echo 7
// Motor control pins
#define LEFT_MOTOR_PIN1 8
#define LEFT_MOTOR_PIN2 9
#define RIGHT_MOTOR_PIN1 10
#define RIGHT_MOTOR_PIN2 11
// Distance thresholds for obstacle detection
#define MAX_DISTANCE 40
#define MIN_DISTANCE_BACK 5
```

مطمئن شوید که مقادیر «MIN_DISTANCE_BACK» و «MAX_DISTANCE» را با توجه به نیازهای خاص خود و ویژگی‌های ربات خود تنظیم کرده‌اید. مقادیر مناسب برای «MIN_DISTANCE_BACK»و «MAX_DISTANCE» به الزامات و ویژگی‌های خاص ربات دنبال‌کننده انسان شما بستگی دارد. شما باید عواملی مانند سرعت ربات خود، زمان پاسخ سنسورها و ... را در نظر بگیرید.

در اینجا چند دستورالعمل کلی وجود دارد که به شما در انتخاب مقادیر مناسب کمک می کند.

MIN_DISTANCE_BACK: این مقدار نشان دهنده فاصله‌ای است که ربات باید در آن هنگام که مانع یا چیزی مستقیماً در جلو تشخیص داده می‌شود به عقب برود. باید در فاصله ای تنظیم شود که به ربات اجازه دهد بدون برخورد با مانع یا دست، ایمن به عقب برگردد. یک مقدار مناسب می تواند حدود ۵-۱۰ سانتی متر باشد.

MAX_DISTANCE: این مقدار نشان‌دهنده حداکثر فاصله‌ای است که ربات مسیر پیش رو را مستقیم می‌داند و می‌تواند به حرکت به جلو ادامه دهد. باید در فاصله ای تنظیم شود که فضای کافی برای حرکت ربات بدون برخورد با هیچ مانع یا دستی فراهم شود. اگر دست شما و موانع از این محدوده خارج شوند، ربات باید متوقف شود. یک مقدار معمولی می تواند حدود ۳۰-۵۰ سانتی متر باشد.

این مقادیر فقط پیشنهاد هستند و ممکن است لازم باشد آنها را بر اساس ویژگی های خاص ربات خود و محیطی که در آن کار می کند تنظیم کنید.

این خطوط محدودیت سرعت موتور را تعیین می کنند. MAX_SPEED حد آستانه برای سرعت موتور را نشان می‌دهد. مقادیر سرعت معمولاً در محدوده ۰ تا ۲۵۵ هستند و می‌توانند مطابق با نیازهای خاص ما تنظیم شوند.

```
// Maximum and minimum motor speeds
#define MAX_SPEED 150
#define MIN_SPEED 75
```

تابع "setup()" یک بار در شروع کار اجرا می شود. در تابع setup() ، پین‌های کنترل موتور (LEFT_MOTOR_PIN1، LEFT_MOTOR_PIN2، RIGHT_MOTOR_PIN1، RIGHT_MOTOR_PIN2) را به‌عنوان پایه‌های خروجی با استفاده از «pinMode()» تنظیم می‌کنیم. ما همچنین پایه های تریگر (S1Trig، S2Trig، S3Trig) سنسورهای اولتراسونیک را به عنوان پایه های خروجی و پایه های اکو (S1Echo، S2Echo، S3Echo) را به عنوان پایه های ورودی تنظیم کردیم. در نهایت، ما ارتباط سریال را با نرخ باود ۹۶۰۰ برای دیباگ کردن مقداردهی اولیه می کنیم.

```
void setup()
{
// Set motor control pins as outputs
  pinMode(LEFT_MOTOR_PIN1, OUTPUT);
  pinMode(LEFT_MOTOR_PIN2, OUTPUT);
  pinMode(RIGHT_MOTOR_PIN1, OUTPUT);
  pinMode(RIGHT_MOTOR_PIN2, OUTPUT);

//Set the Trig pins as output pins
  pinMode(S1Trig, OUTPUT);
  pinMode(S2Trig, OUTPUT);
  pinMode(S3Trig, OUTPUT);

//Set the Echo pins as input pins
```

```
  pinMode(S1Echo, INPUT);
  pinMode(S2Echo, INPUT);
  pinMode(S3Echo, INPUT);

// Initialize the serial communication for
debugging
  Serial.begin(9600);
}
```

این بلوک کد از سه تابع(”sensorOne()“، ”sensorTwo()“، ”sensorThree()“) تشکیل شده است که مسئول اندازه‌گیری فاصله با استفاده از حسگرهای اولتراسونیک هستند.

تابع ”sensorOne()“ فاصله را با استفاده از اولین سنسور اولتراسونیک اندازه گیری می کند. توجه به این نکته مهم است که تبدیل مدت زمان پالس به فاصله بر اساس این فرض است که سرعت صوت تقریباً ۳۴۳ متر در ثانیه است. با نصف کردن نتیجه حاصل فاصله حدودا درست به دست می آید.

توابع «sensorTwo()» و «sensorThree()» به روشی مشابه کار می کنند اما به ترتیب برای حسگرهای اولتراسونیک دوم و سوم هستند.

```
// Function to measure the distance using an ultrasonic sensor
int sensorOne()
{
  //pulse output
  digitalWrite(S1Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(S1Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S1Trig, LOW);
  long t = pulseIn(S1Echo, HIGH);//Get the pulse
  int cm = t / 29 / 2; //Convert time to the distance
  return cm; // Return the values from the sensor
}
```

```
//Get the sensor values
int sensorTwo()
{
  //pulse output
  digitalWrite(S2Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(S2Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S2Trig, LOW);
  long t = pulseIn(S2Echo, HIGH);//Get the pulse
  int cm = t / 29 / 2; //Convert time to the distance
  return cm; // Return the values from the sensor
}
```

```
//Get the sensor values
int sensorThree()
{
  //pulse output
  digitalWrite(S3Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(S3Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S3Trig, LOW);
  long t = pulseIn(S3Echo, HIGH);//Get the pulse
  int cm = t / 29 / 2; //Convert time to the distance
  return cm; // Return the values from the sensor
}
```

در این بخش، تابع «loop()» با فراخوانی توابع«sensorOne()»، «sensorTwo()»و «sensorThree()» برای اندازه‌گیری فاصله از حسگرهای اولتراسونیک شروع می‌شود. سپس فاصله ها در متغیرهای"frontDistance" ، "leftDistance"و"rightDistance" ذخیره می‌شوند.

در مرحله بعد از "Serial" برای چاپ مقادیر فاصله در سریال مانیتور برای اهداف دیباگینگ استفاده می‌شود.

```
void loop() {

  int frontDistance = sensorOne();
  int leftDistance = sensorTwo();
  int rightDistance = sensorThree();
  Serial.print("Front: ");
  Serial.print(frontDistance);
  Serial.print(" cm, Left: ");
  Serial.print(leftDistance);
  Serial.print(" cm, Right: ");
  Serial.print(rightDistance);
  Serial.println(" cm");
```

در این بخش از شرایط کد بررسی می‌کند که آیا فاصله جلویی کمتر از مقدار آستانه «MIN_DISTANCE_BACK» است یا خیر. اگر این شرط درست باشد، به این معنی است که فاصله جلو بسیار کم است و ربات باید برای جلوگیری از برخورد به عقب حرکت کند.

در این حالت تابع "moveBackward()" فراخوانی می شود.

```
if (frontDistance < MIN_DISTANCE_BACK)
  {
    moveBackward();
    Serial.println("backward");
  }
```

اگر شرط قبلی نادرست باشد، شرط دیگر بررسی می کند که فاصله جلو کمتر از فاصله چپ، کمتر از فاصله سمت راست و کمتر از آستانه «MAX_DISTANCE» باشد. اگر این شرط درست باشد، به این معنی است که فاصله جلو در بین سه فاصله کمتر است و همچنین کمتر از حداکثر آستانه فاصله است. در این حالت، تابع "moveForward()" فراخوانی می شود تا ربات به جلو حرکت کند.

```
else if (frontDistance < leftDistance &&
frontDistance < rightDistance && frontDistance <
MAX_DISTANCE)
```

```
  {
    moveForward();
    Serial.println("forward");
  }
```

اگر شرط قبلی نادرست باشد، این شرط بررسی می شود. بررسی می‌کند که آیا فاصله سمت چپ کمتر از فاصله راست و کمتر از آستانه «MAX_DISTANCE» باشد. این شرط نشان می دهد که فاصله سمت چپ در بین سه فاصله کمتر است و همچنین کمتر از حد آستانه فاصله است. بنابراین، تابع turnLeft() فراخوانی می شود تا ربات به چپ بپیچد.

```
else if (leftDistance < rightDistance &&
leftDistance < MAX_DISTANCE)
  {
    turnLeft();
    Serial.println("left");
  }
```

اگر هیچ یک از شرایط قبلی برآورده نشد، این شرط بررسی می شود. این تضمین می کند که فاصله کمتر از آستانه "MAX_DISTANCE" باشد. این شرط نشان می دهد که فاصله مناسب در بین سه فاصله کمتر است و کمتر از آستانه حداقل فاصله است. سپس تابع "turnRight()"برای چرخش ربات به راست فراخوانی می شود.

```
else if (rightDistance < MAX_DISTANCE)
  {
    turnRight();
    Serial.println("right");
  }
else
  {
    stop();
    Serial.println("stop");
  }
```

اگر هیچ یک از شرایط قبلی درست نباشد، به این معنی است که هیچ یک از فاصله ها شرایط حرکت را برآورده نمی کند. بنابراین، تابع "stop()" برای توقف ربات فراخوانی می شود.

به طور خلاصه، این کد فواصل سه سنسور اولتراسونیک را بررسی می کند و بر اساس ۳ سنسور اولتراسونیک با کمترین فاصله، جهت حرکت خود را تعیین می کند.

و در ادامه توابع حرکت ربات در جهت های مختلف ...

```
// Motor control functions
void moveForward()
{
  analogWrite(LEFT_MOTOR_PIN1, MAX_SPEED);
  analogWrite(LEFT_MOTOR_PIN2, LOW);
  analogWrite(RIGHT_MOTOR_PIN1, MAX_SPEED);
  analogWrite(RIGHT_MOTOR_PIN2, LOW);
}
```

```
void moveBackward()
{
  analogWrite(LEFT_MOTOR_PIN1, LOW);
  analogWrite(LEFT_MOTOR_PIN2, MAX_SPEED);
  analogWrite(RIGHT_MOTOR_PIN1, LOW);
  analogWrite(RIGHT_MOTOR_PIN2, MAX_SPEED);
}
```

```
void turnRight()
{
  analogWrite(LEFT_MOTOR_PIN1, LOW);
  analogWrite(LEFT_MOTOR_PIN2, MAX_SPEED);
  analogWrite(RIGHT_MOTOR_PIN1, MAX_SPEED);
  analogWrite(RIGHT_MOTOR_PIN2, LOW);
}
```

```
void turnLeft()
{
  analogWrite(LEFT_MOTOR_PIN1, MAX_SPEED);
  analogWrite(LEFT_MOTOR_PIN2, LOW);
  analogWrite(RIGHT_MOTOR_PIN1, LOW);
  analogWrite(RIGHT_MOTOR_PIN2, MAX_SPEED);
}
```

```
void stop()
{
  analogWrite(LEFT_MOTOR_PIN1, LOW);
  analogWrite(LEFT_MOTOR_PIN2, LOW);
  analogWrite(RIGHT_MOTOR_PIN1, LOW);
  analogWrite(RIGHT_MOTOR_PIN2, LOW);
}
```

و در پایان این پروژه لازم به ذکر است که ساخت ربات تعقیب کننده انسان با استفاده از آردوینو و سه حسگر التراسونیک، پروژه ای هیجان انگیز و مفید است که برنامه نویسی، الکترونیک و مکانیک را با هم ترکیب می کند. با تطبیق پذیری آردوینو و در دسترس بودن قطعات ارزان قیمت، ساخت آن هم کار ساده ای است.

## ۶-۴- پروژه وینچ

همیشه برای حمل و نقل و جابجایی از یک سری وسایل کمک می گیریم که عمل جا به جایی را برای ما آسان و راحت تر می‌نماید از این رو ما می دانیم که برای حمل و نقل و بلند کردن اجناس و کالا ها میتوان از روش های مانند قرقره و مشابه آن استفاده کرد که هرچه تعداد قرقره ها بیشتر باشد عمل انتقال راحت تر و بهتر می‌شود و به این صورت این روال هر روزه دارای پیشرفت بوده تا به اینکه از گیربکس ها در این امر یاری گرفته شد، گیربکس ها تا حد مشخصی گشتاور را بالا می‌بردند که این امکان را فراهم کنند که با کمترین نیروی وارده قدرتی زیادی را ایجاد نماید و این سبب شد تا از گیربکس ها در امر انتقال و جابجایی استفاده شود که از نظر

گشتاور قابل تحمل متغیر بوده و در صنایع مختلف از قبیل ساختمانی، معدن، دریایی و... مورد استفاده است.

وینچ یا کشنده به سیستمی گفته می شود که به وسیله نیروی محرکه ایجاد شده توسط الکتروموتور یا هیدروموتور یا موتور احتراقی و انتقال نیرو به گیربکس سرعت آن را کم کرده و به قدرت آن می افزاید و از خروجی گیربگس یک شفت رابط به قرقره متصل شده و در عمل گردش درام سیم بکسل به دور آن پیچیده شده و منجر به حرکت رفت و برگشتی می شود و به وسیله این سیستم و سیم بکسلی که به آن متصل است اجسام و سایر موارد را به صورت افقی و عمودی حرکت می دهد. به زبان ساده‌تر وینچ به ابزاری گفته می‌شود که که برای کشیدن و یا بالا بردن اجسام به کار می‌رود. وینچ دو نوع کلی دارد. نوع اول وینچ فقط اجسام را در جهت افقی می‌کشد و نوع دوم آن اجسام را در جهت عمود بالا می‌برد. برای افزایش قدرت مکانیکی وینچ‌ها از قرقره‌های مرکب استفاده می‌کنند. به این صورت که در قرقره‌های مرکب سیم‌ها را چند لا به قلاب متصل می‌کنند و با این عمل میزان قدرت و بهره وری وینچ افزایش پیدا می کند.

از نمونه های استفاده آن میتوان به جرثقیل های سقفی داخل کارخانه ها ، جرثقیل های متحرک کارگاهی ، تاور کرین های ساختمانی ، اتوموبیل های جنگل بانی و جهت کشش کابل های فشار قوی برق و... اشاره کرد. وینچ در کارگاه های صنعتی و کارخانه ها بسیار پر استفاده می باشد. همچنین از وینچ در ساخت نیروگاه های آبی هم استفاده می شود. در این پروژه یک وینچ با قابلیت های مختلفی از جمله تشخیص حرکت، تشخیص فاصله، تشخیص دما، سنجش جریان و کنترل از راه دور ساخته شده است که این قابلیت ها می تواند باعث کاهش ریسک و هزینه های یک کارگاه بشود. برای مثال این قابلیت ها می تواند بر روی میزان خرابی دستگاه تاثیر مثبت بگذارد و موجب صرفه جویی در استفاده از نیروی کار بشود. همچنین قابلیت هایی مانند سنجش جریان و یا تشخیص دما می تواند از اتفاق های ناگوار جلوگیری کند.

مزایای پروژه :

با استفاده از وینچ برای جابجایی و حمل بارهای سنگین و بزرگ، کارایی بسیار بالاتری در انجام کارها به دست می آید. این دستگاه به عنوان یکی از دستگاه های کمکی در صنایع مختلف، برای جابجایی و حمل بارهای سنگین و بزرگ کاربرد دارد و این امر باعث افزایش کارایی و بهبود فرآیندهای تولید و ساخت می شود.

استفاده از وینچ در صنایع مختلف مانند ساختمان سازی و حمل و نقل، باعث کاهش هزینه هایی مانند هزینه نیروی انسانی برای جابجایی و حمل بارهای سنگین و بزرگ می شود. با استفاده از وینچ، بارهای سنگین و بزرگ با سرعت و کارایی بیشتری جابجا می شوند که باعث کاهش هزینه های تولید و ساخت می گردد.

استفاده از وینچ به عنوان یک دستگاه مکانیکی قدرتمند، باعث افزایش ایمنی در محیط های کاری مختلف مانند صنایع ساختمانی و حفاری می شود. وینچ با داشتن سیستم کنترل دقیق، باعث افزایش دقت در جابجایی و حمل بارهای سنگین و بزرگ خواهد شد. این دستگاه به صورت دقیق و با کنترل مناسب، بارهای سنگین و بزرگ را جابجا می کند که باعث کاهش احتمال خطاها و خسارات مالی و کاهش خطراتی مانند صدمات بدنی به کارگران خواهد شد.

وینچ به عنوان یکی از دستگاه های چندمنظوره در صنایع مختلف، در بسیاری از کاربردها مورد استفاده قرار می گیرد. این دستگاه در صنایعی مانند ساختمان سازی، حفاری، حمل و نقل، معدن، کشتی رانی، راه سازی و بسیاری دیگر از صنایع کاربرد دارد. با توجه به این کاربردها، استفاده از وینچ به عنوان یک دستگاه چند منظوره در صنایع مختلف، باعث افزایش کارایی و بهبود فرآیندهای تولید و ساخت می شود.

نرم‌افزارهایی که برای ساخت این پروژه به کار گرفته شده است عبارتند از :

۱. Proteus 8.13
۲. Arduino
۳. AutoCAD
۴. Serial Bluetooth Terminal

ساخت وینچ :

برای شروع پروژه وینچ ابتدا یک وینچ ساده می سازیم و مرحله به مرحله قابلیت های مختلف را اضافه می کنیم. برای ساخت یک وینچ نیاز به یک الکتروموتور و یک گیربکس حلزونی که بر روی شاسی نصب می شود و کلید چپ و راست داریم. الکتروموتور مورد استفاده در این پروژه یک الکتروموتور با توان نیم اسب بخار (0.37KW) می باشد. دور بر دقیقه(RPM) شفت خروجی این الکتروموتور برابر ۱۵۰۰ است. گیربکس مورد استفاده هم یک گیربکس حلزونی با نسبت ۱ به ۱۸ می باشد. الکتروموتور و گیربکس با هم کوبل شده و گیربکس ۱۵۰۰ دور بر دقیقه شفت الکتروموتور را به حدود ۸۳ دور بر دقیقه تبدیل میکند به این معنا که قدرت جابجایی الکتروموتور زمانی که با گیربکس کوبل می شود چندین برابر شده و در نتیجه سرعت کم می شود. از آنجایی که وینچ یک وسیله برای جابجایی اجسام سنگین می باشد در هنگام استفاده از آن نیاز به سرعت بالا نداریم.

(شکل ۶-۲) الکتروموتور

(شکل ۶-۳) گیربکس حلزونی

ساخت شاسی

برای ساخت شاسی ابتدا با استفاده از ابعاد الکتروموتور و گیربکس ابعاد شاسی را مشخص می کنیم. عرض گیربکس تهیه شده در حدود ۱۳ سانتی متر می باشد و طول گیربکس و الکتروموتور در هنگامی که با یکدیگر کوپل شده اند در حدود ۴۰ سانتی متر می باشد پس در نتیجه طول شاسی در حدود ۵۰ سانتی متر و با توجه به آنکه به گیربکس قرقره ای با عرض در حدود ۱۰ سانتی متر برای بلند کردن اجسام متصل می شود که باید آویزان باشد، عرض را در حدود ۳۰ سانتی متر در نظر می گیریم. حال برای آنکه گیربکس به شاسی متصل باشد و قرقره همزمان آویزان باشد، به یک پل در شاسی نیاز داریم که یک طرف گیربکس به آن و طرف دیگر گیربکس به یک طرف شاسی متصل باشد. به دلیل فشار بالایی که در حین کار به شاسی وارد میشود برای ساخت آن از نبشی ۳ سانتی متر که گوشت آن در حدود ۲ میلی متر است، استفاده می کنیم.

همچنین می توانیم ۴ عدد نبشی کوچک به چهار گوشه شاسی جوش کنیم تا با سوراخ کردن نبشی ها و استفاده از پیچ و مهره بتوانیم شاسی را روی سطوح مختلف نصب کنیم.

*(شکل ۶-۴) نبشی های برش خورده*

برای افزایش استقامت شاسی به دو روش می توانیم نبشی ها را به یکدیگر جوش بدهیم. روش اول فارسی بر می باشد، به این صورت که ابتدا و انتهای هر یک از نبشی ها را با زاویه ۴۵ درجه ببریم و آنها را به یکدیگر جوش بدهیم. روش دیگری وجود دارد که در آن لبه ی نبشی ها را به

اندازه عرض نبشی ای که به آن ها جوش داده می شود، برش دهیم تا نبشی ها فیت یکدیگر باشند.

در هر دو روش بالا شاسی استقامت بالایی داشته و جوشکاری به صورت استاندارد انجام می شود اما در روش برش لبه خط جوش بیشتری وجود دارد که موجب استقامت بیشتر می شود. روش های دیگری مانند استفاده از پیچ برای اتصال نبشی ها به یکدیگر و ساخت شاسی وجود دارد اما این روش ها استقامت کمتری نسبت به روش جوشکاری دارند به همین دلیل برای ساخت شاسی وینچ که در حین کار فشار زیادی بر آن وارد می شود از روش جوشکاری استفاده می کنیم.

برای جوشکاری ابتدا باید زاویه نبشی ها را با استفاده از گونیا تنظیم کنیم سپس آن ها را به یکدیگر جوش بدهیم. پس از تنظیم نبشی ها ابتدا یک خال جوش می زنیم طوری که دو نبشی به یکدیگر متصل باشند اما مقداری انعطاف پذیری وجود داشته باشد تا اگر در حین جوشکاری زاویه نبشی ها مقداری تغییر کرد بتوان آن را اصلاح کرد. سپس با استفاده از گونیا یک بار دیگر زاویه نبشی ها را بررسی می کنیم تا از درست بودن آن اطمینان حاصل کنیم.

*(شکل ۶-۵) آماده سازی نبشی جهت انجام جوشکاری*

برای جوشکاری قطعات شاسی با توجه به گوشت نبشی مورد استفاده و ابعاد آن از الکترود شماره ۳ استفاده شد و آمپر دستگاه جوشکاری (اینورتر جوش) روی ۹۰ آمپر تنظیم شد تا هنگام جوشکاری نبشی سوراخ نشود و در عین حال جوش مقاوم باشد. به دلیل فاصله ی زیاد بین پل و ضلع طولی شاسی استقامت این قسمت از شاسی نسبت به قسمت های دیگر آن کمتر می باشد به همین دلیل می توان یک نبشی در راستای عرض شاسی از وسط پل به وسط ضلع طولی شاسی جوش داد تا استقامت شاسی بیشتر شود.

*(شکل ۶-۶) شاسی وینچ*

در انتها ۴ عدد نبشی کوچک بریده شده و به چهار گوشه شاسی جوش شد تا با سوراخ کردن نبشی ها و استفاده از پیچ و مهره بتوانیم شاسی را روی سطوح مختلف نصب کنیم.

*(شکل ۶-۷) نبشی برای پیچ شدن شاسی بر روی سطوح مختلف*

کوپل کردن الکتروموتور و گیربکس

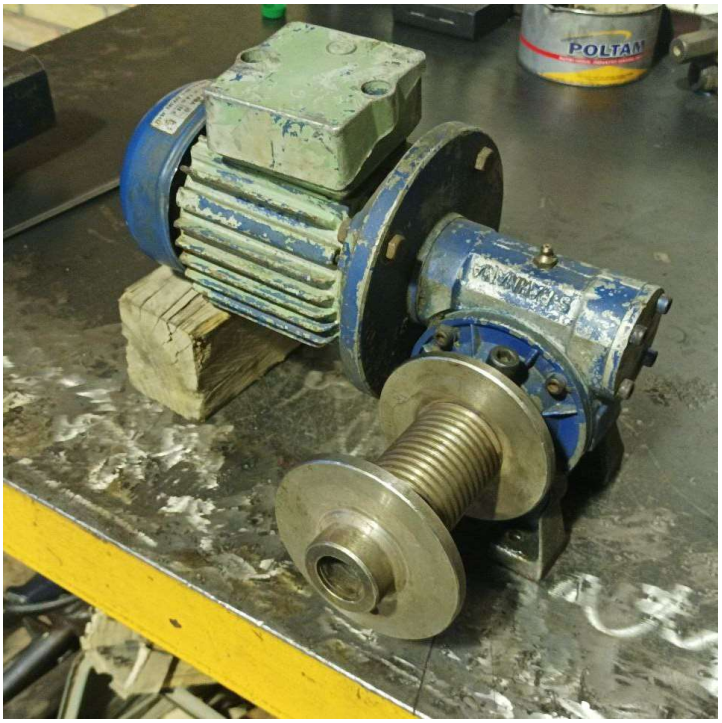
پس از ساخت شاسی گیربکس و الکتروموتور را با یک دیگر کوپل می کنیم. بر روی شفت الکتروموتور یک خار وجود دارد که با استفاده از آن شفت را به درستی سنتر و این دو را با هم کوپل می کنیم. اگر خار موجود بر روی شفت الکتروموتور مشکل داشته باشد یا جای خار که بر روی شفت گیربکس قراردارد مشکلی داشته باشد، در حین کوپل کردن مشکل به وجود می آید. پس از جا رفتن خار و کوپل شدن گیربکس و الکتروموتور پیچ هایی که گیربکس و الکتروموتور را به یکدیگر متصل نگه میدارد را می بندیم تا برای نصب بر روی شاسی آماده باشد.

قرقره

پس از کوپل کردن الکترو موتور و گیربکس قرقره ای برای وینچ تهیه کردیم. شفت گیربکس حلزونی موجود به قطر یک اینچ می باشد که معادل ۲۵ میلی متر است اما قطر سوراخ داخلی قرقره تهیه شده که شفت گیربکس داخل آن قرار می گیرد با قطر شفت گیربکس تفاوت داشت به همین خاطر قرقره به تراشکاری داده شد تا قطر سوراخ داخلی قرقره با قطر شفت گیربکس هم اندازه شود.

*(شکل ۶-۸) قرقره*

در دستگاهی مانند وینچ قرقره و شفت گیربکس باید با یکدیگر هماهنگ باشد، در واقع قرقره همزمان با چرخش شفت گیربکس حلزونی می چرخد؛ به همین دلیل شفت گیربکس و قرقره باید با یکدیگر فیکس باشند. برای فیکس کردن قرقره با شفت گیربکس دو روش وجود دارد. روش اول با استفاده از خار موجود بر روی قرقره (روش خارزنی) و روش دوم استفاده از پیچ ضامن که دو عدد پیچ ضامن در انتهای قرقره وجود دارد و زمانی که قرقره روی شفت و سر جای خود قرار دارد پیچ های ضامن را می بندیم تا قرقره با شفت گیربکس فیکس شود.




*(شکل ۶-۹) فیکس کردن قرقره با استفاده از پیچ ضامن*

## بخش الکترونیکی پروژه

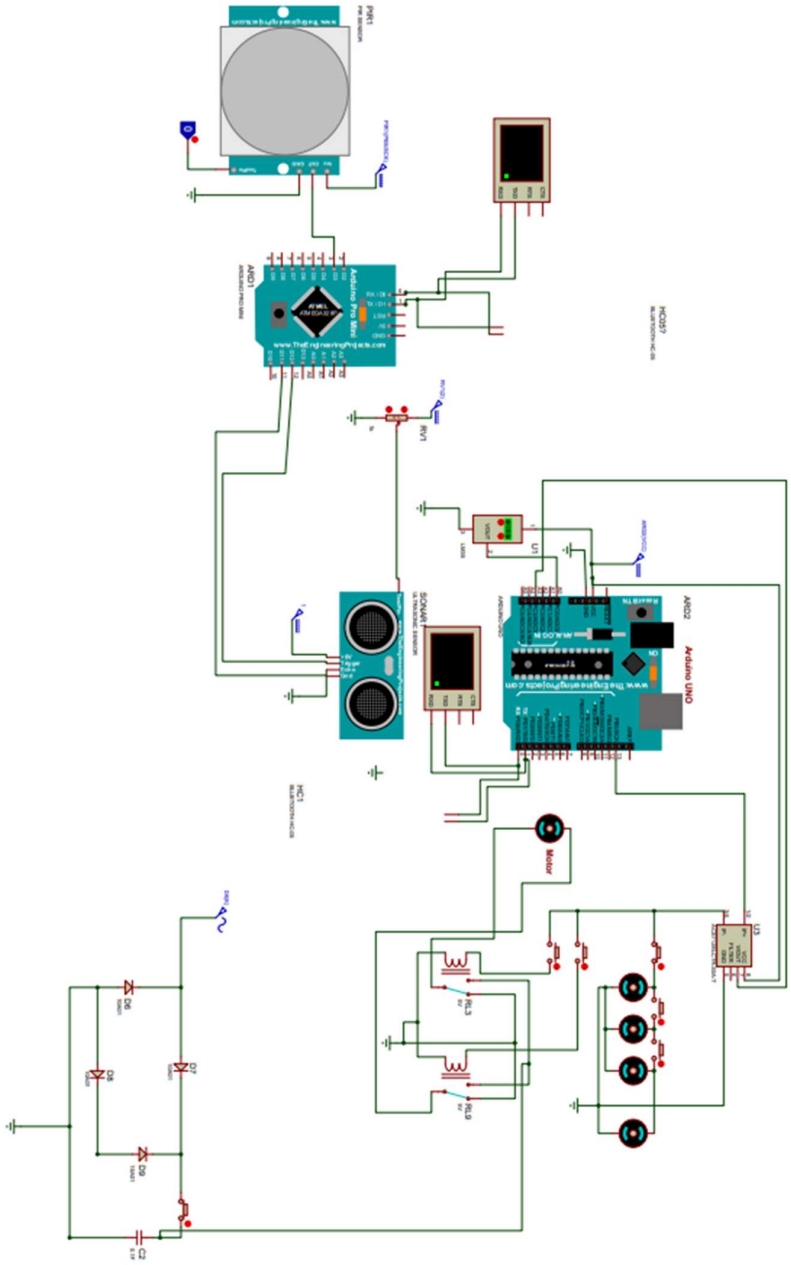
این وینچ دارای قابلیت های تشخیص دما، تشخیص جریان، کنترل از راه دور، تشخیص فاصله و تشخیص حرکت می باشد . در این پروژه با استفاده از آردوینو این قابلیت ها را راه اندازی می کنیم. نحوه کارکرد این پروژه به این صورت می باشد که ابتدا برق ورودی برای الکتروموتور و گیربکس توسط سنسور تشخیص ولتاژ و جریان اندازه گیری می‌شود و سپس وارد رله ای می شود که این رله توسط بلوتوث و آردوینو از راه دورکنترل می شود. در صورتی که دستور از گوشی موبایل صادر شود رله باز می شود و برق وارد الکتروموتور می شود و وینچ شروع به کار می کند. در همین حین سنسور دمایی که بر روی الکتروموتور نصب شده، درجه دما را به آردوینو انتقال می‌دهد و آردوینو به وسیله ی بلوتوث دما را به گوشی انتقال می دهد.

سنسور تشخیص حرکت و تشخیص فاصله هم بر روی الکتروموتور نصب شده و به وسیله یک ماژول بلوتوث دیگر اطلاعات را به گوشی موبایل انتقال می دهد. اکر سنسور تشخیص حرکت، حرکتی را تشخیص بدهد با روشن شدن ال ای دی و پیامی که از طریق بلوتوث به گوشی موبایل ارسال می‌شود متوجه این موضوع می‌شویم. برای پروژه ای مانند پروژه وینچ می توان از ماژول‌ها

و سنسورهای دیگر برای راه اندازی قابلیت های مختلف دیگری مانند سنجش نور، تشخیص رطوبت و باران، تشخیص شعله و .... که در این پروژه راه اندازی نشده اند استفاده کرد.

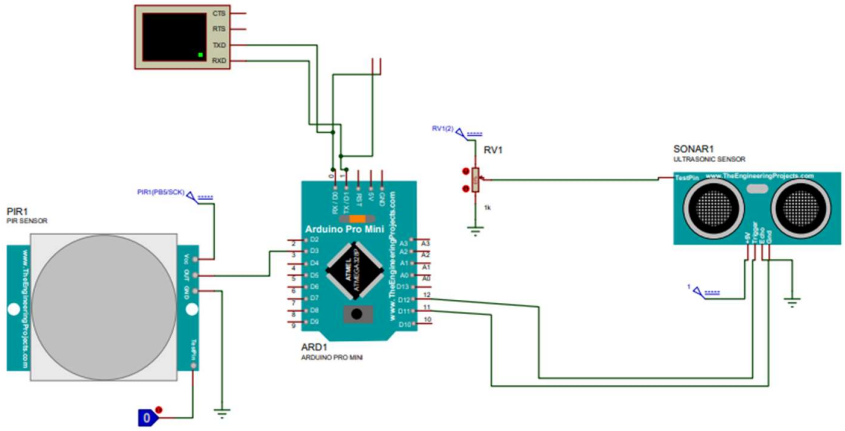
## شبیه سازی در نرم افزار پروتئوس

در ادامه ساخت پروژه وینچ مدار الکترونیکی را در نرم افزار پروتئوس طراحی و شبیه سازی میکنیم. برای این کار باید ابتدا سنسور ها و دیگر قطعات الکترونیکی مورد نیاز پروژه را مشخص کنیم. پس از انتخاب سنسور ها و قطعات الکترونیکی باید کتابخانه قطعاتی که در نرم افزار پروتئوس موجود نیست را دانلود کرده و سپس به کتابخانه نرم افزار پروتئوس اضافه کنیم تا بتوانیم از آن قطعات در شبیه سازی استفاده کنیم. قطعات استفاده شده در این پروژه که کتابخانه آن ها به صورت پیش فرض در نرم افزار موجود نبود عبارتند از: آردوینو، ماژول فاصله سنج SRF04 ، ماژول بلوتوث HC-05 ، سنسور تشخیص حرکت HC-SR501 و سنسور تشخیص دما.




*(شکل ۶-۱۰) مدار شبیه سازی شده پروژه وینچ در نرم افزار پروتئوس*

همانطور که در مدار شبیه سازی شده در نرم افزار پروتئوس مشخص است مدار پروژه از دو بخش مجزا تشکیل شده است. دلیل اینکه مدار به دو بخش تبدیل شده است سختی برقراری ارتباط بین سنسور تشخیص فاصله و سنسور PIR با آردوینو می باشد. سنسور تشخیص فاصله و سنسور PIR در این پروژه باید بر روی قلاب نصب شوند و هنگام استفاده از وینچ قلاب حرکت می کند و امکان استفاده از سیم برای اتصال این دو سنسور به آردوینو نیست در نتیجه برای برقراری ارتباط این دو سنسور با گوشی موبایل می توان از یک آردوینو دیگر که به همراه سنسور سنجش نور و سنسور PIR بر روی قلاب نصب می شود و توسط بلوتوث به گوشی موبایل متصل می شود، استفاده شده است.




*(شکل ۶-۱۱) مدار شبیه سازی شده برد انتقال دهنده اطلاعات در نرم افزار پروتئوس*

## برنامه نویسی در نرم افزار آردوینو

پس از شبیه سازی مدار در نرم افزار پروتئوس ، به کد نویسی در نرم افزار آردوینو می پردازیم. باید توجه داشت در نرم افزار آردوینو همانند نرم افزار پروتئوس برخی از کتابخانه های مورد نیاز به صورت پیش فرض موجود نیستند و ابتدا باید کتابخانه های مورد نیاز را نصب کرد.

کد آردوینو کنترل کننده الکتروموتور »«

تعریف متغیرها: در این بخش، متغیرها و پارامترهای مختلف برای کنترل پورت ها، سنسورها، و متغیرهای مربوط به اندازه‌گیری جریان و ولتاژ تعریف شده‌اند.

```
int first_LED = 13;
int second_LED = 9;
int state;
```

یک متغیر برای ذخیره وضعیتی که از سریال خوانده می‌شود.

```
int flag = 0;
int ThermistorPin = 0;
int Vo;
float R1 = 10000;
```

مقاومت اولیه به عنوان ۱۰ کیلو اهم تعیین شده است.

```
float logR2, R2, T, Tc;
```

متغیرهایی که برای محاسبه دما استفاده می‌شوند.

```
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 =
2.019202697e-07;
```

ضرایب محاسبه دما بر اساس مقاومت.

```
const int sensorIn = A3;
int mVperAmp = 66 ;
```

مقدار تبدیل ولتاژ به جریان (میلی‌ولت بر آمپر) برای سنسور جریان.

```
double Voltage = 0;
```

ولتاژ خوانده شده از سنسور دما.

```
double VRMS = 0;
```

ولتاژ RMS محاسبه شده از جریان AC.

```
double AmpsRMS = 0;
```

ولتاژ RMS محاسبه شده از ولتاژ AC.

تنظیمات اولیه (setup)

در تابع setup، پین‌ها به عنوان ورودی یا خروجی تعریف شده‌اند و سرعت ارتباط سریال برای ارسال و دریافت داده‌ها به ۹۶۰۰ تنظیم شده است.

```
void setup() {
 pinMode(7, OUTPUT);
 pinMode(second_LED, OUTPUT);
  pinMode(first_LED, OUTPUT);
```

```
 Serial.begin(9600);
}
```

حلقه اصلی (loop):

در حلقه اصلی، دمای محیط اندازه‌گیری شده و در صورتی که بیشتر از ۵۰ درجه سانتیگراد باشد LED روشن می‌شود. سپس اندازه‌گیری جریان و ولتاژ انجام می‌شود و مقادیر به صورت سریال چاپ می‌شوند. در نهایت بر اساس دستوراتی که از طریق سریال دریافت می‌شوند، وضعیت پورت‌ها تغییر می‌کند. در صورتی که آردوینو از طریق بلوتوث عدد ۱ یا ۲ را دریافت کند به ترتیب در پورت شماره ۱۳ و ۹ تغییر ایجاد می کند (HIGH). در صورتی که آردوینو عدد ۰ را از طریق بلوتوث دریافت کند در پورت های ۱۳ و ۹ تغییر وضعیت ایجاد می کند (LOW). پورت های ۱۳ و ۹ در آردوینو به ماژول رله ی ۲ کاناله متصل می باشد که روشن و خاموش بودن و جهت چرخش والکتروموتور را کنترل می کند. همچنین این برنامه کنترل دما و جریان را با استفاده از سنسورها انجام می دهد و از ارتباط سریال برای کنترل و نمایش اطلاعات استفاده می‌کند.

```
void loop() {
 Vo = analogRead(ThermistorPin);
```

مقدار خوانده شده از ترمیستور به وسیله این دستور خوانده می‌شود و در متغیر Vo ذخیره می‌شود.

```
 R2 = R1 * (1023.0 / (float)Vo - 1.0);
```

مقاومت ترمیستور محاسبه می‌شود

```
 logR2 = log(R2);
```

لگاریتم مقاومت ترمیستور محاسبه می‌شود.

```
 T = (1.0 / (c1 + c2 * logR2 + c3 * logR2 * logR2 * logR2));
```

دما با استفاده از فرمول استین هارت محاسبه می‌شود.

```
 Tc = T - 273.15;
```

دما به واحد سانتیگراد تبدیل می‌شود.

```
 Serial.print("Temperature: ");
 Serial.print(Tc);
```

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

چاپ مقدار دما به واحد سانتیگراد.

```
 if (Tc > 50) {
  digitalWrite(7, HIGH);
 }
else {
  digitalWrite(7, LOW);
 }
```

اگر دما بیشتر از ۵۰ درجه باشد، پورت ۷ آردوینو به HIGH تغییر وضعیت پیدا می کند و در غیر این صورت LOW می ماند.

در این بخش ابتدا مقدار خوانده شده از ترمیستور (Vo) اندازه‌گیری می‌شود و با استفاده از فرمول‌های مربوط به ترمیستور، دما به واحد سانتیگراد محاسبه می‌شود. سپس دما از طریق سریال چاپ می‌شود و در صورتی که دما بیشتر از ۵۰ درجه سانتیگراد باشد، در پروت شمارهٔ ۷ تغییر وضعیت ایجاد می شود.

```
Voltage = getVPP();
 VRMS = (Voltage / 2.0) *0.707;
 AmpsRMS = (VRMS * 1000) / mVperAmp;
 Serial.print(AmpsRMS);
 Serial.println(" Amps RMS");
```

در هر چرخه ولتاژ اندازه‌گیری شده و با استفاده از آن مقدار ولتاژ RMS و جریان RMS محاسبه می‌شود.

```
}
float getVPP()
{
 float result;
 int readValue;
 int maxValue = 0;
 int minValue = 1023;
 uint32_t start_time = millis();
```

```
while ((millis() - start_time) < 1000)
{
 readValue = analogRead(sensorIn);
 // see if you have a new maxValue
 if (readValue > maxValue)
 {
  maxValue = readValue;
 }
 if (readValue < minValue)
 {
  minValue = readValue;
 }
}
result = ((maxValue - minValue) * 5.0) / 1023;
return result;
```

این تابع ولتاژ پیک به پیک را از سنسور خوانده و محدوده‌ی ولتاژ را برمی‌گرداند  برای انجام این کار، از یک حلقه  while  استفاده شده که در طول یک ثانیه نمونه برداری می‌کند و حداقل و حداکثر مقادیر را ثبت می‌کند. مقدار ولتاژ پیک به پیک با تفاوت حداکثر و حداقل به دست می‌آید و به ولتاژ معادل میلی ولت تبدیل می‌شود.

در این بخش، از یک حلقه برای اندازه‌گیری مقدار خوانده شده از سنسور جریان استفاده شده است سپس مقدار ولتاژ و جریان محاسبه می‌شود و این مقادیر از طریق سریال چاپ می‌شوند.

```
if (Serial.available() > 0)
 {
 state = Serial.read();
 flag = 0;
 }
```

در صورتی که داده جدیدی از سریال در دسترس باشد، وضعیت  (state)  تغییر می‌کند و flag به صفر تنظیم می‌شود.

```
if (state == '1') {
 digitalWrite(first_LED, HIGH);
 if (flag == 0) {
  Serial.println("Forward");
  flag = 1;
 }
```

اگر وضعیت (state) ۱ باشد در پورت شماره ۱۳ تغییر وضعیت ایجاد می شود و LED اول روشن می‌شود و پیام Forward چاپ می‌شود.

```
} else if (state == '2') {
   digitalWrite(second_LED, HIGH);
   if (flag == 0) {
  Serial.println("Reverse");
  flag = 1;
 }
```

اگر وضعیت (state) ۲ باشد در پورت شماره ۹ تغییر وضعیت ایجاد می شود و LED دوم روشن می‌شود و پیام Reverse چاپ می‌شود.

```
} else if (state == '0') {
 digitalWrite(second_LED, LOW);
 digitalWrite(first_LED, LOW);
 if (flag == 0) {
  Serial.println("Stop");
  flag = 1;
  delay(400);
 }
}
```

اگر وضعیت (state) ۰ باشد، هر دو پورت به حالت LOW تغییر وضعیت می دهند و LED ها خاموش می‌شوند و پیام Stop چاپ می‌شود.

اگر دما بیشتر از ۵۰ درجه سانتیگراد باشد، هر دو LED خاموش می‌شوند و پیام Hot چاپ می‌شود.

در این بخش ابتدا بررسی می‌شود که آیا داده جدیدی از سریال در دسترس است یا خیر. اگر داده در دسترس باشد، وضعیت (state) تغییر می‌کند و flag را برابر با صفر می‌کند و با توجه به وضعیت (state) پورت ها کنترل می‌شوند.

کد آردوینو انتقال دهنده اطلاعات »«

تعریف متغیرها:

این کد تشخیص فاصله و سنسور حرکت (PIR) را کنترل می کند. ابتدا متغیرها و پارامترهای مختلف برای کنترل پورت ها و سنسورها تعریف شده‌اند.

```
int state;
```

یک متغیر برای ذخیره وضعیتی که از سریال خوانده می‌شود.

```
const int TRIG_PIN = 12;
const int ECHO_PIN = 11;
const int LED = 2;
#define button 2
int value = 0;
```

تعریف پایه های TRIG و ECHO برای سنسور تشخیص فاصله انجام شده است.

```
int flag = 0;
int ledPin = 13;
int inputPin = 8;
int pirState = LOW;
```

وضعیت سنسور PIR به صورت اولیه LOW (بدون حرکت).

```
int val = 0;
```

متغیری برای ذخیره مقدار خوانده شده از سنسور حرکت.

تنظیمات اولیه (setup)

در تابع setup، پین‌ها به عنوان ورودی یا خروجی تعریف شده‌اند و سرعت ارتباط سریال برای ارسال و دریافت داده‌ها به ۹۶۰۰ تنظیم شده است.

```
void setup() {
```

```
 pinMode(ledPin, OUTPUT);
 pinMode(inputPin, INPUT);
  pinMode(LED,OUTPUT);
 pinMode(TRIG_PIN,OUTPUT);
 pinMode(ECHO_PIN,INPUT);
 Serial.begin(9600);
}
```

حلقه اصلی (loop)

در حلقه ی اصلی مقدار سنسور تشخیص فاصله خوانده می شود و در صورتی که فاصله از ۲۰ سانتی متر کمتر شود پیام Out of range چاپ می شود و ال ای دی خاموش می شود. در غیر این صورت ال ای دی روشن می ماند. همچنین وضعیت سنسور حرکت خوانده شده و اگر حرکت احساس شود و وضعیت پیشین LOW بوده باشد، پیام Motion detected چاپ می‌شود و وضعیت پورت شماره ۱۳ به HIGH تغییر پیدا می کند. اگر دیگر حرکت احساس نشود و وضعیت پیشین HIGH بوده باشد، پیام Motion ended چاپ می‌شود و وضعیت پورت شماره ۱۳ به LOW تغییر پیدا می کند. همچنین داده‌های موجود در سریال خوانده می شود و چاپ می شود.

```
void loop() {
 int duration, distanceCm;

 digitalWrite(TRIG_PIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG_PIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN, LOW);
 duration = pulseIn(ECHO_PIN,HIGH);
 distanceCm = duration /29.1 / 2;

 if (distanceCm <= 20){
   Serial.println("Out of range");
```

```
  Serial.print("Distance : ");
  Serial.print(distanceCm);
  Serial.print("cm LED : OFF");
  Serial.println();
  digitalWrite(LED , LOW);
 }
 else {
  Serial.print("Distance : ");
  digitalWrite(LED , HIGH);
  Serial.print(distanceCm);
  Serial.print("cm LED : ON");
  Serial.println();
 }

val = digitalRead(inputPin);
```

خواندن وضعیت سنسور حرکت و ذخیره در متغیر val.

```
 if (val == HIGH) {
  digitalWrite(ledPin, HIGH);

  if (pirState == LOW) {
   Serial.println("Motion detected!");
   pirState = HIGH;
  }
 } else {
  digitalWrite(ledPin, LOW);

  if (pirState == HIGH) {
   Serial.println("Motion ended!");
```

```
    pirState = LOW;
   }
  }
```

بررسی وضعیت سنسور حرکت و کنترل LED و چاپ پیام‌های Motion detected یا Motion ended در سریال صورت گرفته است.

```
 delay(500);

 if (Serial.available() > 0) {
  state = Serial.read();
  flag = 0;
 }
}
```

بررسی داده‌های موجود در سریال و چاپ آن ها.

قطعات

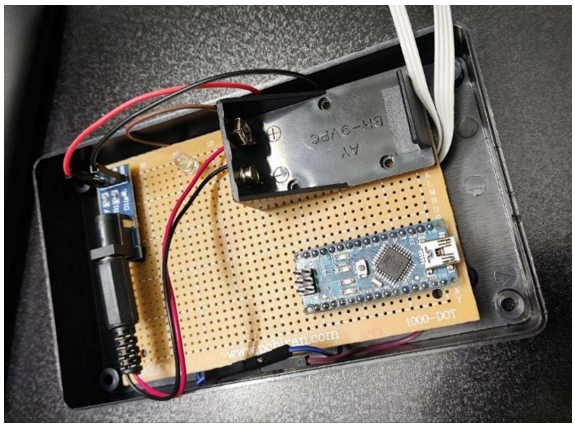
پس از کد نویسی در نرم افزار آردوینو قطعات مورد نیاز برای اجرای پروژه را تهیه می کنیم.
برخی از قطعات الکترونیکی که برای پروژه وینچ تهیه شده عبارت است از:

۱. آردوینو UNO
۲. آردوینو نانو Arduino Nano
۳. ماژول فاصله سنج التراسونیک SRF04
۴. ماژول سنسور جریان ACS712
۵. ماژول تشخیص حرکت مادون قرمز HC-SR501
۶. ماژول بلوتوث HC-05
۷. سنسور دمای NTC 10K
۸. ماژول رله ۵ ولت ۲ کاناله

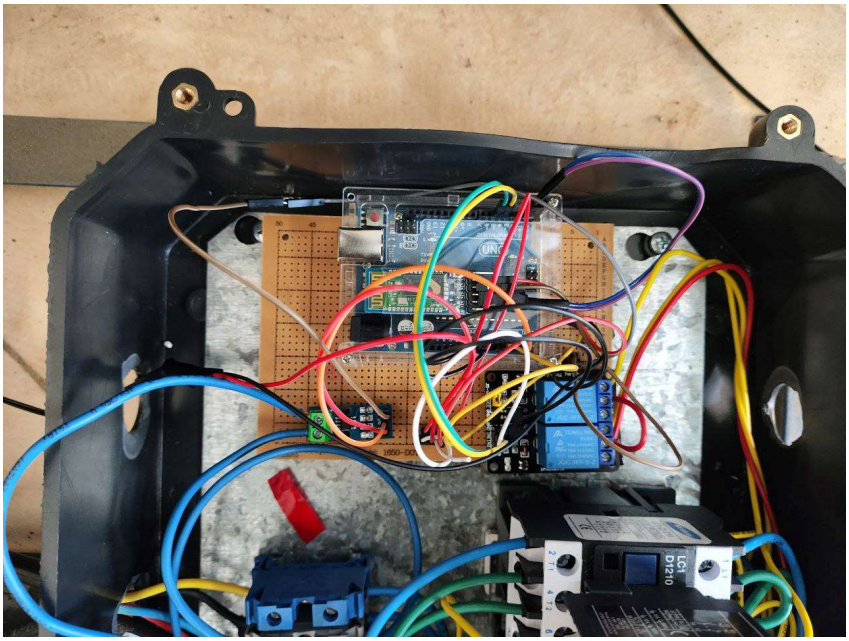
نصب برد های الکترونیکی بر روی شاسی :

در مرحله بعد شروع به ساخت برد های الکترونیکی می کنیم. برای ساخت برد های الکترونیکی این پروژه از برد های سوراخ دار استفاده شده است و قطعات به وسیله برد های سوراخ دار به یک دیگر متصل شده اند. با توجه به کد نوشته شده و پورت های تعیین شده سنسور ها را به آردوینو

ها متصل می کنیم. همچنین چندین لامپ LED برای عیب یابی نصب می‌شود تا زمانی که یکی از قطعات دچار مشکل شد به راحتی عیب یابی شود و قطعه مورد نیاز را تعمیر یا تعویض کرد. همچنین در حین لحیم کاری قطعات باید به این نکته توجه کنیم که هنگام بارگذاری کد در آردوینو پایه های Rx و Tx نباید متصل باشد. اگر این پایه های هنگام آپلود کردن کد متصل باشد کد آپلود نمی شود و با مشکل مواجه می شویم، به همین دلیل برای اتصال بلوتوث پایه های ماژول بلوتوث را لحیم نمی کنیم و از سیم جامپر که یک سر آن مادگی می باشد استفاده می کنیم. همچنین می توانیم برای ماژول های دیگر مانند ماژول تشخیص حرکت هم همین کار را انجام دهیم تا در صورتی که نیاز به تعویض ماژول بود به راحتی این کار را انجام دهیم. از آنجایی که برق مورد نیاز برد باید توسط یک باتری کتابی ۹ ولت تامین شود بر روی هر برد یک جای باتری در نظر گرفتیم. جای باتری باید طوری باشد که باتری به راحتی قابل تعویض باشد. باید توجه داشت که ماژول سنسور جریان و رله دو کاناله بر روی برد طوری نصب شوند که دسترسی به آن ها راحت باشد و کابل برق به راحتی وارد و از آن ها خارج شود. پس از آماده شدن برد های الکترونیکی باید آن ها را بر روی شاسی نصب کنیم. برای محافظت و زیبایی بیشتر جعبه‌هایی برای برد های الکترونیکی تهیه شده که برد ها داخل آن ها قرار می‌گیرند.



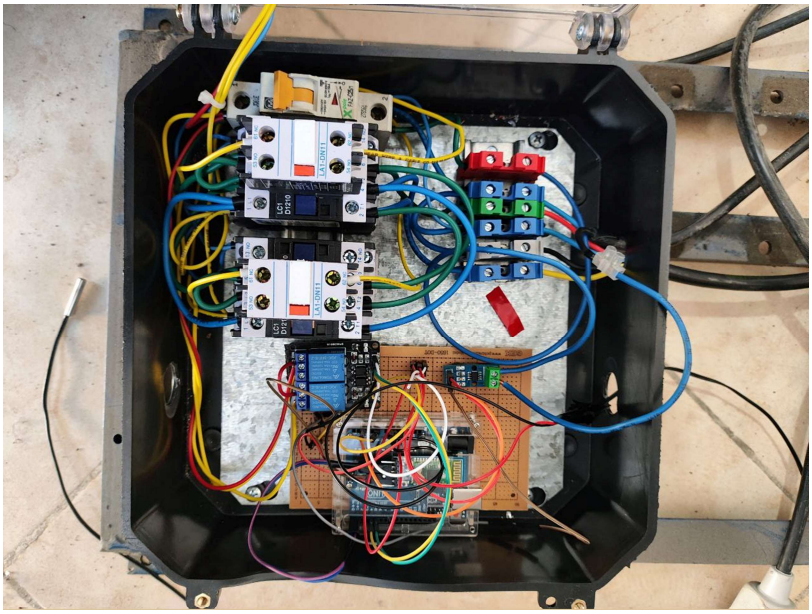
*(شکل ۶-۱۲) برد انتقال دهنده اطلاعات*



*(شکل ۶-۱۳) برد کنترل کننده وینچ*

راه اندازی با استفاده از کنتاکتور:

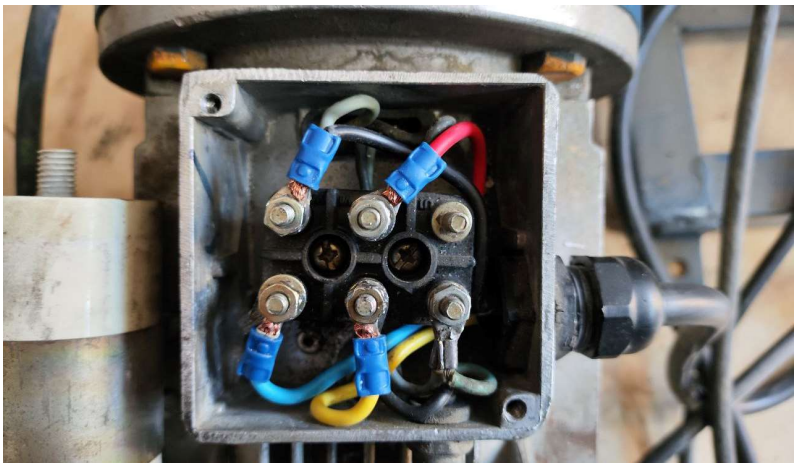
می دانیم از رله دو کاناله فقط می توان یک فاز و یک نول خروجی گرفت و این فاز و نول با هر بار سوییچ کردن رله جابجا می شوند. این جابجایی موجب تغییر جهت چرخش الکتروموتور می شود. با توجه به تخته کلم (ترمینال) الکتروموتور می دانیم با یک فاز و نول که ثابت نیستند نمی توان الکتروموتور را راه اندازی کرد و برای راه اندازی الکتروموتور نیاز به یک فاز و نول دیگر که ثابت هستند داریم. همچنین رله های مورد استفاده ظرفیت تحمل جریان بالایی را ندارد. به همین دلیل از دو کناکتور برای راه اندازی الکتروموتور استفاده می‌شود.



*(شکل ۶-۱۴) تابلو برق شامل برد کنترل کننده وینچ و کناکتور*

تست نهایی

پس از اتصال فاز ها و نول ها به ترمینال الکتروموتور و نصب برد های الکترونیکی بر روی شاسی وینچ آماده تست می باشد. هنگام تست باید حتما به نکات ایمنی توجه داشته باشیم.



*(شکل ۶-۱۵) سربندی الکتروموتور*

در طول پروسه تست نحوه کاکرد تمامی سنسور ها توسط اطلاعاتی که از طریق بلوتوث به گوشی موبایل ارسال می شود بررسی می شود تا مشکلی وجود نداشته باشد. پس از اینکه تست گرفته شد و تمامی قطعات به درستی کار کردند، وینچ آماده می باشد و ساخت پروژه به اتمام رسیده است.

*(شکل ۶-۱۶) وینچ آماده شده طی این پروژه*

## ۶-۵- ربات تشخیص لبه پرتگاه

ربات تشخیص لبه، رباتی است که تنها به وسیله دو IR سنسور، میتواند لبه ها را شناسایی کند و از آن دوری کند.

این ربات توسط یک کنترل کننده اداره شده و نکته جالب توجه این است که حتی از روی عمد هم به پایین پرت نمی شود.

قطعات مورد نیاز:

- یک برد آردوینو (مدل UNO مرسوم تر است)
- ۴ عدد موتور (به همراه چرخ)
- شیلد درایور موتور l293d
- باتری لیتیوم یون ۳,۷ ولت ۲ عدد
- نگهدارنده باتری
- ۲ عدد سنسور IR
- سیم و جامپر
- ماژول HC-05 (بلوتوث)

کد برنامه نویسی:

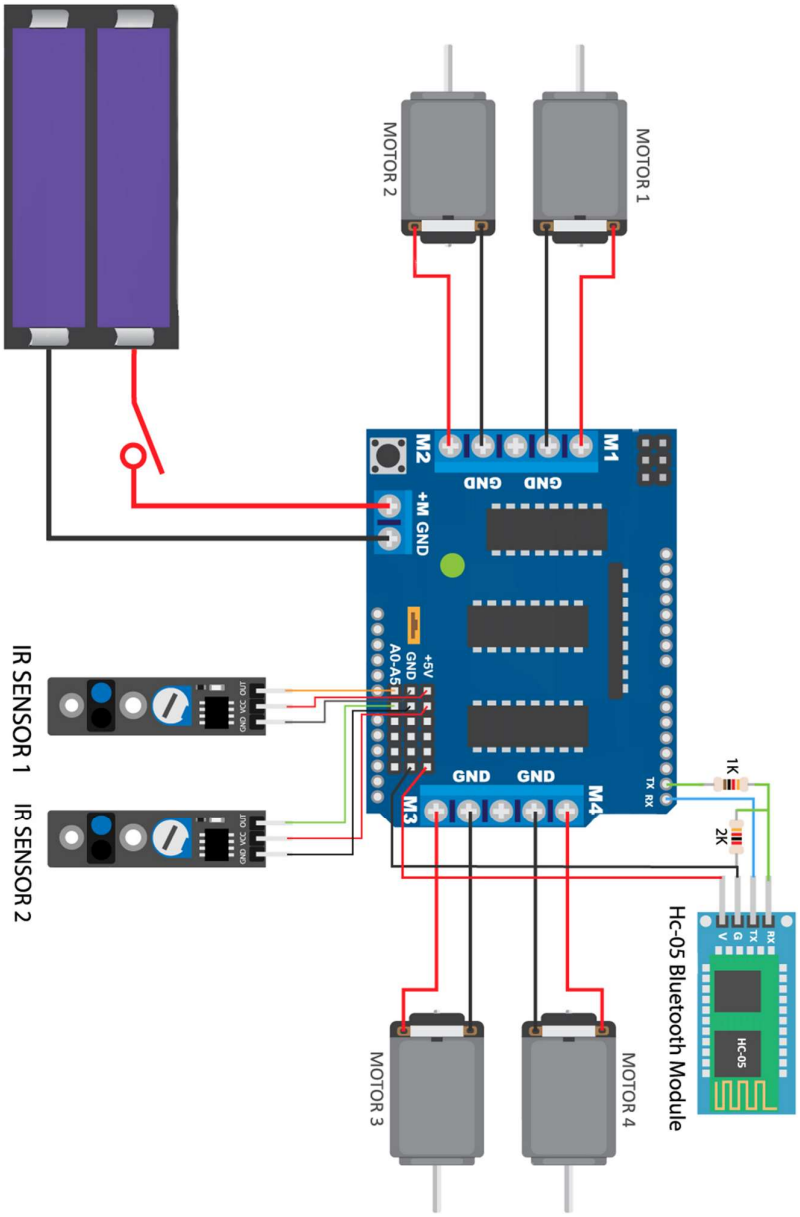
```
#include <AFMotor.h>  //L293 برای راه اندازی شیلد


#define IR1 A0
#define IR2 A1

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);

char command;
void setup()
{
 Serial.begin(9600);
 pinMode(IR1, INPUT);
 pinMode(IR2, INPUT);
}
```

```
void loop(){
 int Right = digitalRead(IR1);
 int Left = digitalRead(IR2);

 if(Serial.available() > 0){
  command = Serial.read();
  Stop();
  Serial.println(command);

  switch(command){
  case 'F':
   forward();
   break;
  case 'B':
    back();
   break;
  case 'L':
   left();
   break;
  case 'R':
   right();
   break;
  }
  if(Right == 1 || Left == 1) {
   motor1.setSpeed(255);
   motor1.run(BACKWARD);
   motor2.setSpeed(255);
   motor2.run(BACKWARD);
   motor3.setSpeed(255);
   motor3.run(BACKWARD);
   motor4.setSpeed(255);
   motor4.run(BACKWARD);

  }else if(Right == 0 || Left == 0) {
   command = Serial.read();
  }
 }
}
```

```
void forward()
{
 motor1.setSpeed(150);
 motor1.run(FORWARD);
 motor2.setSpeed(150);
 motor2.run(FORWARD);
 motor3.setSpeed(150);
 motor3.run(FORWARD);
 motor4.setSpeed(150);
 motor4.run(FORWARD);
}
void back()
{
 motor1.setSpeed(150);
 motor1.run(BACKWARD);
 motor2.setSpeed(150);
 motor2.run(BACKWARD);
 motor3.setSpeed(150);
 motor3.run(BACKWARD);
 motor4.setSpeed(150);
 motor4.run(BACKWARD);
}
void left()
{
 motor1.setSpeed(200);
 motor1.run(BACKWARD);
 motor2.setSpeed(200);
 motor2.run(BACKWARD);
 motor3.setSpeed(200);
 motor3.run(FORWARD);
 motor4.setSpeed(200);
 motor4.run(FORWARD);
}
void right()
{
 motor1.setSpeed(200);
 motor1.run(FORWARD);
 motor2.setSpeed(200);
```

```
 motor2.run(FORWARD);
 motor3.setSpeed(200);
 motor3.run(BACKWARD);
 motor4.setSpeed(200);
 motor4.run(BACKWARD);
}
void Stop()
{
 motor1.setSpeed(0);
 motor1.run(RELEASE);
 motor2.setSpeed(0);
 motor2.run(RELEASE);
 motor3.setSpeed(0);
 motor3.run(RELEASE);
 motor4.setSpeed(0);
 motor4.run(RELEASE);
}
```

شماتیک اتصالات ربات:




*(شکل ۶-۱۷) اتصالات قطعات*

## واژه نامه‌ی فارسی به انگلیسی

| | | | |
|---|---|---|---|
| آردوینو | Ardouino | درگاه | Port |
| آلتراسونیک | Ultrasonic | ریز تراشه | Microchip |
| اتصال دهنده | Connector | زمان | Time |
| برد الکترونیکی | Electrical board | سون سگمنت | Seven-segment |
| بلوتوث | Bluetooth | غلط | False |
| تنظیم کننده | Regulator | متعادل | Balance |
| حساسیت | Sensitive | متغیر | Value |
| حفاظ | Shield | متن باز | OPEN-SOURCE |
| درست | True | | |

## واژه نامه‌ی انگلیسی به فارسی

| | | | |
|---|---|---|---|
| آردوینو | Ardouino | تنظیم کننده | Regulator |
| متعادل | Balance | حساسیت | Sensitive |
| بلوتوث | Bluetooth | سون سگمنت | Seven-segment |
| اتصال دهنده | Connector | حفاظ | Shield |
| برد الکترونیکی | Electrical board | زمان | Time |
| غلط | False | درست | True |
| ریز تراشه | Microchip | آلتراسونیک | Ultrasonic |
| متن باز | OPEN-SOURCE | متغیر | Value |
| درگاه | Port | | |

## منابع و مأخذ

1.http:// docs.arduino.cc/hardware/uno-rev3. Date: 5/9/2023
2.www.uio.no/studier/emner/matnat/ifi/IN1060/v21/arduino/arduino projects-book.pdf Date: 3/6/2023
3. Boloor, Adith Jagadish (2015). *Arduino by example*. Packt Publishing Ltd.
4. Yarnold, Stuart (2015). *Arduino in easy steps*. In Easy Steps.
5. Barrett, Steven F (2022). *Arduino microcontroller processing for everyone!*. Springer Nature.
6. https://arduino.ir/ Date: 6/3/2023
7. Margolis, Michael, Brian Jepson, and Nicholas Robert Weldin (2020). *Arduino cookbook: recipes to begin, expand, and enhance your projects*. O'Reilly Media.
8. Smythe, Richard J (2021). *Advanced Arduino Techniques in Science*. Apress.
9. سفاهن، علیرضا (۱۴۰۲). *مرجع تخصصی ARDUINO به همراه پروژه های کاربردی*، تهران، آروین نگار.
۱۰. طالبی، اشکان، صفورا قاسمیان (۱۳۹۸). *آموزش آردوینو با محوریت رباتیک*، تهران ، دیباگران تهران.
۱۱. زارعی علی آبادی، ابراهیم (۱۳۹۸). *ریز پردازنده AVR*، چاپ اول، تهران، انتشارات حافظ پژوه.
۱۲. زارعی علی آبادی، ابراهیم (۱۳۹۸). *۱۹ پروژه کاربردی با میکروکنترلر AVR* ، چاپ اول، تهران، انتشارات حافظ پژوه.
۱۳. زارعی علی آبادی، ابراهیم (۱۴۰۰). *کار با میکروکنترلر AVR* ، چاپ اول، تهران، انتشارات حافظ پژوه.