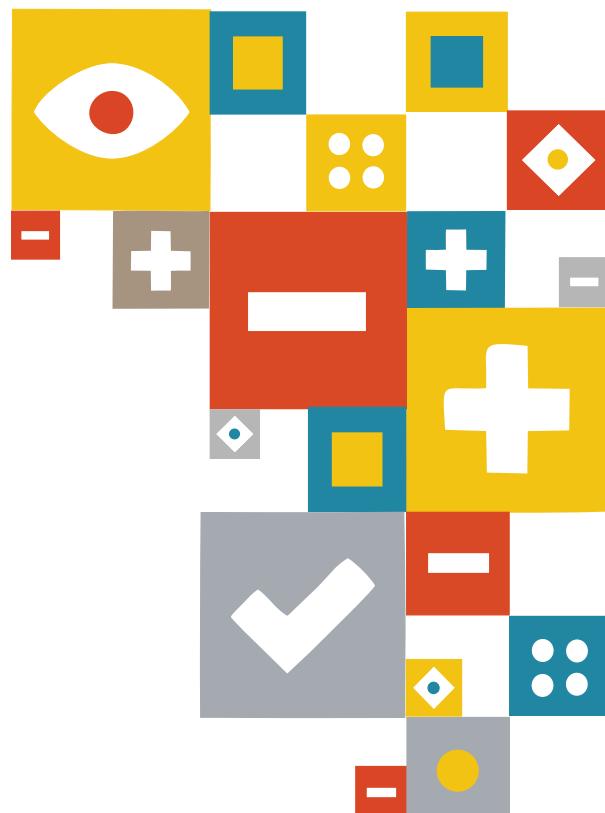


الكترودوينو

آموزش جامع الکترونیک
در قالب ۳۴ پروژه کاربردی جذاب



کتابچه راهنمای + فیلم آموزشی فارسی

محصول شرکت کنکاش تک فرارو

تلفن پشتیبانی: ۰۲۱-۰۷۵۱۶۴۴۱

kankashtech.com/electroduino

یکی از دیدگاه های نوین در حوزه آموزش، تلفیق چهار زمینه علوم، فناوری، مهندسی و ریاضیات (Science, Technology, Engineering, Mathematics) است که به اختصار STEM نامیده می شود. با استفاده از STEM و خلق آموزش هایی جذاب، کودکان و نوجوانان می توانند از طریق بازی های علمی، خلاقیت های خود را پرورش دهند. مجموعه الکترودوینو، از سری محصولات آموزشی دانشمند جوان (Junior Scientist) در زمینه الکترونیک است که بر اساس همین دیدگاه تهیه شده است. در این مجموعه، با استفاده از "برد آردوبینو" مدارهای الکترونیک، به صورت کاربردی آموزش داده خواهد شد. این دفترچه، راهنمای شما در کنکاش علمی در دنیای الکترودوینو خواهد بود. در این دفترچه، اطلاعات موردنیاز برای ساخت و راه اندازی ۳۴ مدار الکترونیکی با استفاده از قطعات و تجهیزات موجود در مجموعه ارائه شده است.

پس از یادگیری این ۳۴ مدار، شما دانش و تجربه کافی برای انجام پروژه ها و آزمایش های مدنظر خود را خواهید داشت.

[www.kankash**tech**.com/electroduino](http://www.kankashtech.com/electroduino)
[t.me/kankash**tech**](https://t.me/kankashtech)

پست الکترونیک:
[info@kankash**tech**.com](mailto:info@kankashtech.com)

تلگرام: *ID*
[@kankash**tech**_admin](https://t.me/kankashtech_admin)



فهرست

مقدمه

۵	آردوینو چیست؟
۷	برد برد چیست؟
۸	مقاومت.
۹	نرم افزار آردوینو
۱۶	نصب درایور اتصال
۱۸	نصب کتابخانه قطعات
۱۹	استفاده از کدهای راهنمای پروژه ها
۲۰	پروژه ۱ - LED چشمک زن
۲۵	پروژه ۲ - کار با چند LED
۳۳	پروژه ۳ - دکمه های فشاری
۳۹	پروژه ۴ - تنظیم سرعت چشمک زدن LED
۴۳	پروژه ۵ - تنظیم شدت نور LED
۵۰	پروژه ۶ - RGB LED
۶۱	پروژه ۷ - مازول RGB LED
۶۳	پروژه ۸ - سون سگمنت
۸۳	پروژه ۹ - سوئیچ زاویه
۸۷	پروژه ۱۰ - سنسور دما LM35
۹۲	پروژه ۱۱ - سنسور دما و رطوبت
۹۷	پروژه ۱۲ - LCD ۱۶۰۲-I2C
۱۰۴	پروژه ۱۳ - سنسور سنجش سطح آب
۱۰۸	پروژه ۱۴ - سنسور تشخیص حرکت



فهرست

۱۱۴	پروژه ۱۵ - سنسور شدت نور
۱۱۸	پروژه ۱۶ - سنسور تشخیص شعله آتش
۱۲۳	پروژه ۱۷ - سنسور سنجش رطوبت خاک
۱۲۷	پروژه ۱۸ - سنسور تشخیص صدا
۱۳۱	پروژه ۱۹ - سنسور تشخیص رنگ TCRT5000
۱۳۵	پروژه ۲۰ - Joystick
۱۴۰	پروژه ۲۱ - ملودی
۱۴۸	پروژه ۲۲ - ماژول رله
۱۵۳	پروژه ۲۳ - سروروموتور
۱۵۷	پروژه ۲۴ - موتور DC
۱۶۲	پروژه ۲۵ - استپ موتور
۱۶۷	پروژه ۲۶ - سنسور فاصله سنج اولتراسونیک
۱۷۲	پروژه ۲۷ - شیفت رجیستر
۱۸۲	پروژه ۲۸ - سون سگمنت ۴ رقم
۱۹۰	پروژه ۲۹ - ماژول نمایشگر TM1637
۱۹۷	پروژه ۳۰ - ماژول ساعت DS1302
۲۰۲	پروژه ۳۱ - ماتریس LED 8x8
۲۱۰	پروژه ۳۲ - کنترل از راه دور IR
۲۱۶	پروژه ۳۳ - صفحه کلید 4x4
۲۲۶	پروژه ۳۴ - ماژول RFID



آردوینو (Arduino) چیست؟

آردوینو یک پلتفرم پردازندۀ منبع باز (Open-Source) است که برای جذاب و ساده کردن تجربیات الکترونیک طراحی شده است. آردوینو دارای زبان برنامه نویسی ساده و مختص خود بوده که با کاربران زیاد و شبکه پشتیبانی وسیع، یک پلتفرم عالی برای افراد مبتدی تا حرفه ای برای ساخت پروژه های زیاد و شبکه پشتیبانی وسیع، یک پلتفرم عالی برای افراد مبتدی تا حرفه ای برای ساخت پروژه های DIY-Do it Yourself است.

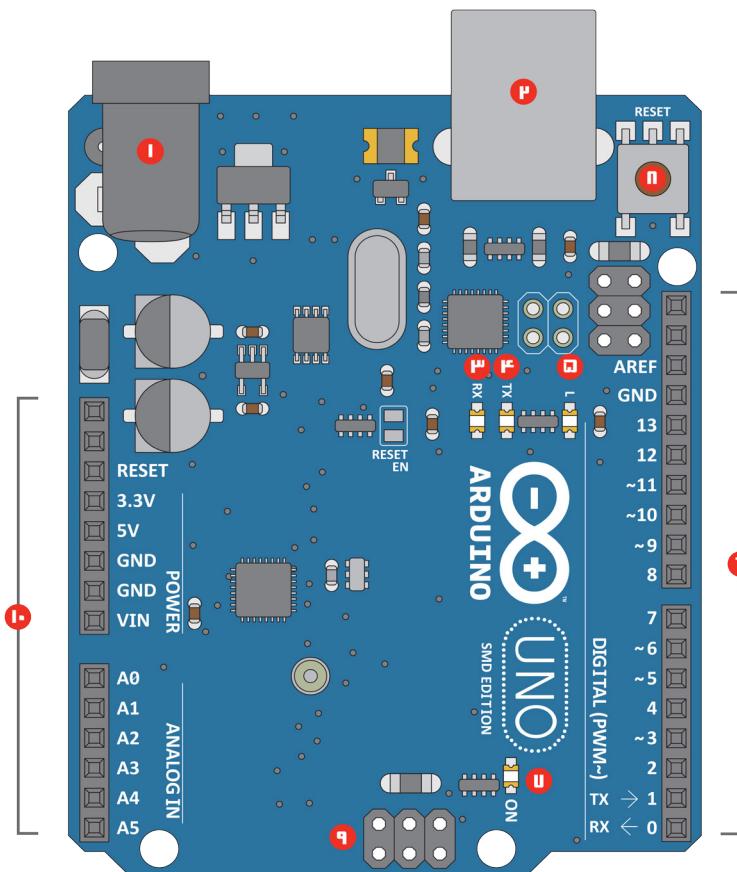
آردوینو یک کامپیوتر کوچک قابل حمل است. این برد قادر به گرفتن داده های ورودی (INPUT) نظیر دکمه های فشاری یا خواندن یک سنسور نور و تفسیر این ورودی ها برای کنترل خروجی های مختلف (OUTPUT) نظیر خاموش و روشن کردن یک LED یا یک موتور الکتریکی است.

اینجاست که عبارت "محاسبات فیزیکی" (Physical Computing) ابداع شده است؛ یک برد آردوینو می تواند دنیای الکترونیک را به صورت واقعی و ملموس به دنیای فیزیکی متصل کند.

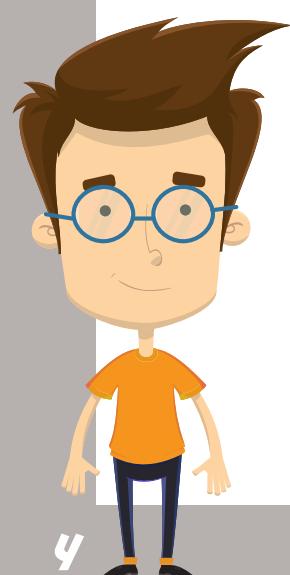
آردوینو Uno یکی از برد های توسعه داده شده بر اساس میکروکنترلر ATmega328 است. در این مجموعه از این برد استفاده می شود زیرا شبکه پشتیبانی و کاربردهای وسیع دارد.

این برد ۱۴ پین ورودی/خروجی (که ۶ پین می تواند خروجی PWM داشته باشند)، ۶ پین ورودی آنالوگ، یک اسیلاتور کریستالی ۱۶ مگاهرتز، یک اتصال USB، یک جک تغذیه، یک هدر ICSP و یک دکمه راه اندازی مجدد (Reset) دارد. نگران نباشید! در ادامه با آیتم های بالا آشنا خواهیم شد. (با توجه به اینکه آردوینو یک پلتفرم منبع باز است، برای اینکه کاربران بتوانند از مطالب مفید کاربران در اینترنت هم استفاده کنند، در این راهنمای این ایتم های سخت افزاری و نرم افزاری استفاده شده است)

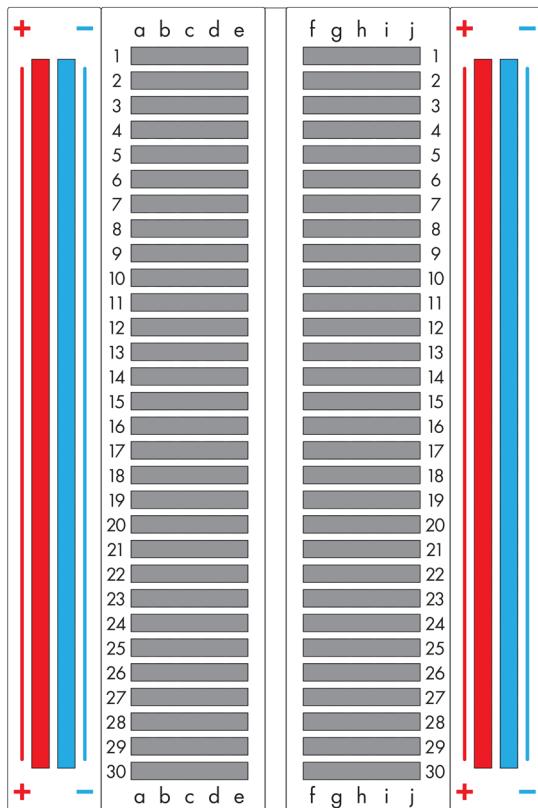
آردوینو (Arduino) چیست؟



۱. جک توان ورودی (Power In) می‌تواند توسط آداپتور ۹ یا ۱۲ ولت یا باتری کار کند.
۲. جک توان ورودی (Power In) توان و ارتباط میان برد آردوینو و کامپیوتر را در هنگام اتصال کابل USB برقرار می‌کند.
۳. چراغ LED RX (علامت RX به معنای دریافت داده): در هنگام دریافت داده - به عنوان مثال در هنگام آپلود کد - چشمک می‌زند.
۴. چراغ LED TX (علامت TX به معنای ارسال داده): در هنگام انتقال داده - به عنوان مثال هنگام اجرای یک برنامه - چشمک می‌زند.
۵. چراغ LED (پین ۱۳: رفع عیب): این LED در کد sketch قرار می‌گیرد تا اطمینان حاصل شود که برنامه به طور صحیح اجرا می‌شود.
۶. پین‌های هدر (Aref, Ground, Digital, Rx, Tx): این پین‌ها می‌توانند برای ورودی، خروجی، توان و زمین استفاده شوند.
۷. چراغ LED: نشان می‌دهد که آردوینو روشن است.
۸. دکمه reset: این دکمه وسیله راه اندازی مجدد آردوینو است که از طریق آن، کد مجدد اجرا می‌شود.
۹. پین‌های ICSP: برای دور زدن boot-loader و Programming استفاده می‌شود.
۱۰. پین‌های هدر (Analog In, Power In, Ground, Power Out, Reset): این پین‌ها می‌توانند برای ورودی، خروجی، توان و زمین استفاده شوند.



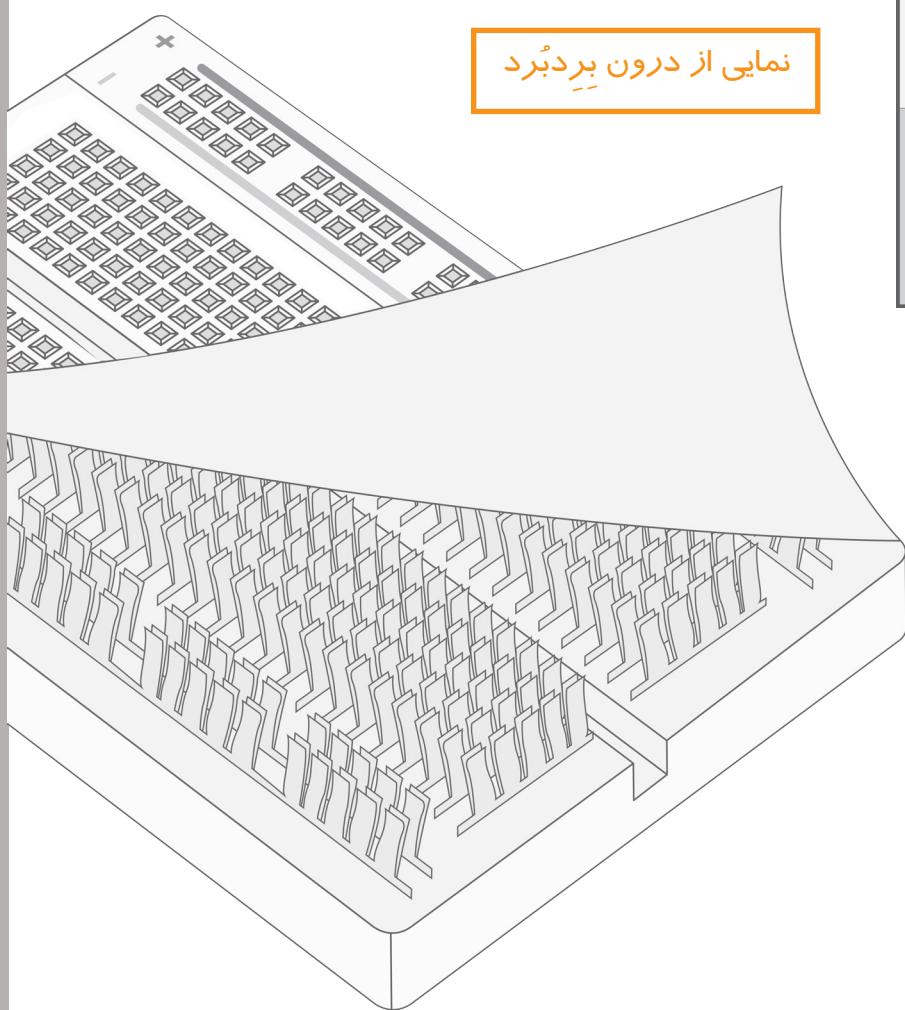
برد برد (Breadboard)



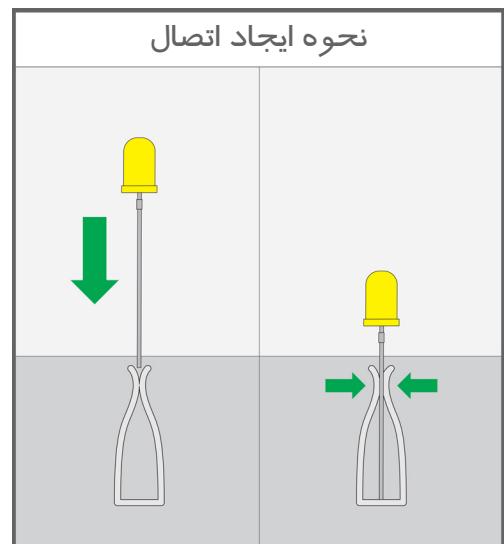
+ دو ستون ۳۰ خانه عمودی در دو سمت برد برد که در تصویر با رنگ قرمز مشخص شده اند، به یکدیگر متصل هستند. در صورتی که به یکی از خانه ها ولتاژ مثبت مصل شود، ۲۹ خانه راستای عمودی آن خانه نیز دارای همان ولتاژ مثبت می شوند.

- خانه های دو ستون عمودی که در تصویر با رنگ آبی مشخص شده اند نیز مانند خانه های قرمز به یکدیگر متصل هستند. در الکترودوینو خانه های آبی به GND آردوینو وصل می شوند.

■ هرکدام از ردیف های افقی که با شماره های ۱-۳۰ و در دو گروه ۵ خانه با حروف a-e و f-j مشخص شده اند نیز به یکدیگر متصل هستند



نمایی از درون برد برد



مقاومت (Resistor)

مقاومت ها اجزا الکترونیکی هستند که جریان عبوری از مدار را محدود می کنند. مقاومت ها جز قطعات غیرفعال (passive) قرار می گیرند بدین معنی که فقط مصرف توان دارند و نمی توانند توان ایجاد کنند.

واحد مقاومت الکتریکی اهم است که توسط حرف یونانی Ω نشان داده می شود. برای مقادیر بالاتر، از کیلو اهم، مگا اهم و گیگا اهم استفاده می شود که به ترتیب 10^3 , 10^6 و 10^9 اهم هستند. همچنین برای مقادیر کوچکتر از میلی اهم استفاده می شود که 10^{-3} اهم است.

مقدار مقاومت توسط رنگ نوارهای روی آن مشخص می شود. تعداد نوارهای رنگی مقاومت بین ۳ تا ۶ متغیر است که در اینجا شیوه تعیین مقاومت انواع ۴ و ۵ نواره توضیح داده می شود.

مقاومت با ۵ نوار رنگی

رنگ	نوار اول	نوار دوم	نوار سوم	ضریب	تلرانس
سیاه	.	.	.	$1 \cdot^{(0)}(1\Omega)$	
قهوه ای	۱	۱	۱	$1 \cdot^{(1)}(1\cdot\Omega)$	$\pm 1\%$
قرمز	۲	۲	۲	$1 \cdot^{(2)}(1\cdot\cdot\Omega)$	$\pm 2\%$
نارنجی	۳	۳	۳	$1 \cdot^{(3)}(1k\Omega)$	
زرد	۴	۴	۴	$1 \cdot^{(4)}(1\cdot k\Omega)$	
سبز	۵	۵	۵	$1 \cdot^{(5)}(1\cdot\cdot k\Omega)$	$\pm 0/5\%$
آبی	۶	۶	۶	$1 \cdot^{(6)}(1M\Omega)$	$\pm 0/25\%$
بنفش	۷	۷	۷	$1 \cdot^{(7)}(1\cdot M\Omega)$	$\pm 0/1\%$
خاکستری	۸	۸	۸	$1 \cdot^{(8)}(1\cdot\cdot M\Omega)$	$\pm 0/05\%$
سفید	۹	۹	۹	$1 \cdot^{(9)}(1G\Omega)$	
طلایی				$1 \cdot^{(-1)}(1\cdot\cdot m\Omega)$	$\pm 5\%$
نقره ای				$1 \cdot^{(-2)}(1\cdot m\Omega)$	$\pm 10\%$

مقاومت با ۴ نوار رنگی

نوار اول از سمت راست نشاندهنده تلرانس (تغییرات) مجاز از مقدار محاسبه شده است.

برای مقاومت های پنج نواره، عدد متناظر با رنگ اولین نوار از سمت چپ صد گان، دومین نوار از سمت چپ ده گان و سومین نوار از سمت چپ یکان عدد پایه و عدد متناظر با نوار چهارم از سمت چپ، ضریب خواهد بود. بدین صورت برای مقاومت ۵ نواره شکل بالا خواهیم داشت:

$$234/4 \text{ M}\Omega @ -0.25\%$$

برای مقاومت های چهار نواره، عدد پایه دو رقمی است و نوار سوم مربوط به ضریب است. در نتیجه برای مقاومت ۴ نواره شکل بالا خواهیم داشت:

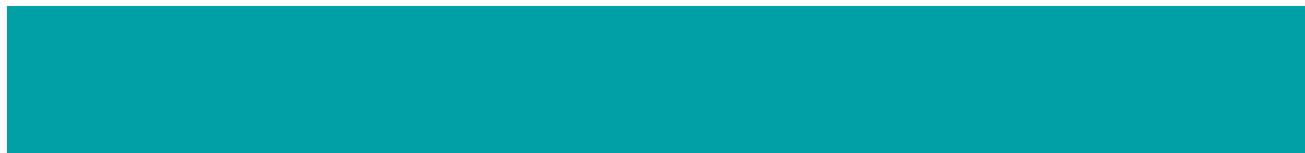
$$220 \text{ }\Omega @ 5\%$$



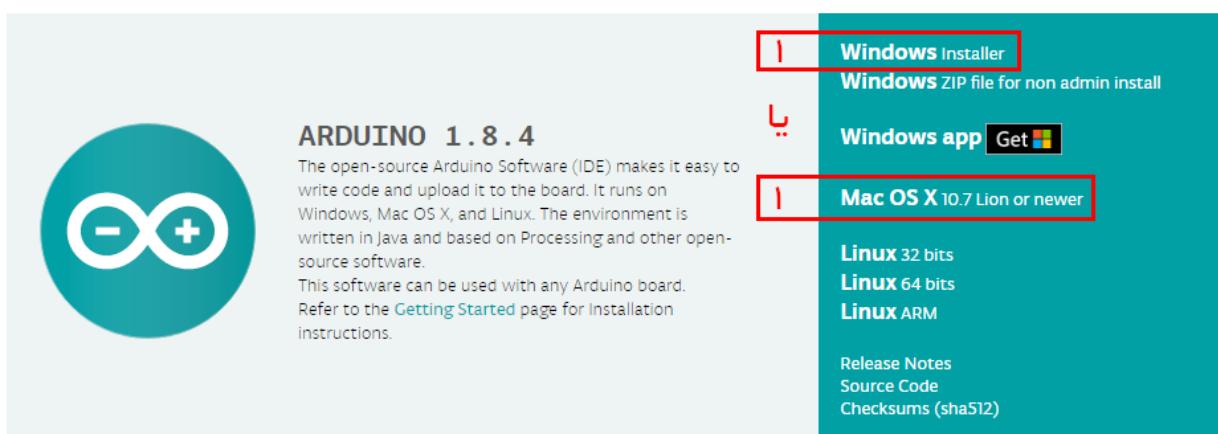
نرم افزار آردوینو (IDE)

برای اینکه بتوانید با آردوینو کار کنید، از نرم افزار آردوینو استفاده کنید. این نرم افزار که (IDE) Integrated Development Environment نام دارد به شما این امکان را می دهد که به آردوینو بگویید دقیقاً چه کاری را انجام دهد. نسخه ۱.۸.۴ این نرم افزار در دی وی دی الکترودینو موجود است. این نرم افزار رایگان بوده ولی در صورت تمایل می توان مبلغی به نویسنده این نرم افزار اهدا کرد. برای دانلود آخرین ویرایش، از طریق مرورگر به آدرس زیر بروید و نسخه مربوط به سیستم خود را دانلود کنید (به حروف کوچک و بزرگ حساس است):

<https://www.arduino.cc/en/Main/Software>



Download the Arduino IDE



Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)

An illustration of a small robot. It has a rectangular body with two legs and two arms. Its head is a blue lightbulb with a plus sign (+) on top. It appears to be made of various electronic components like breadboards and wires.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED
13,553,366 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLE CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

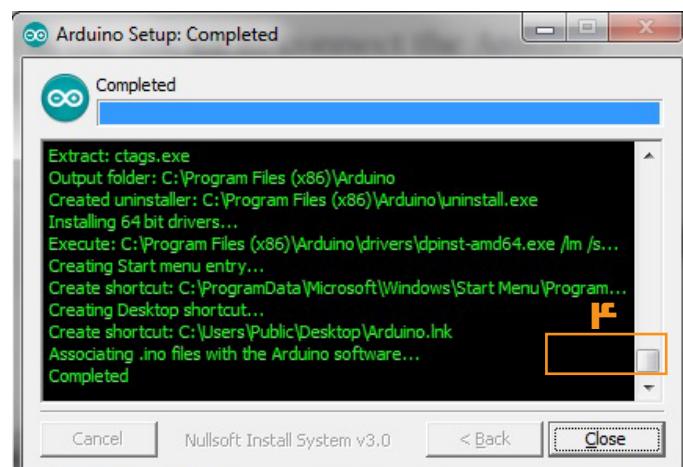
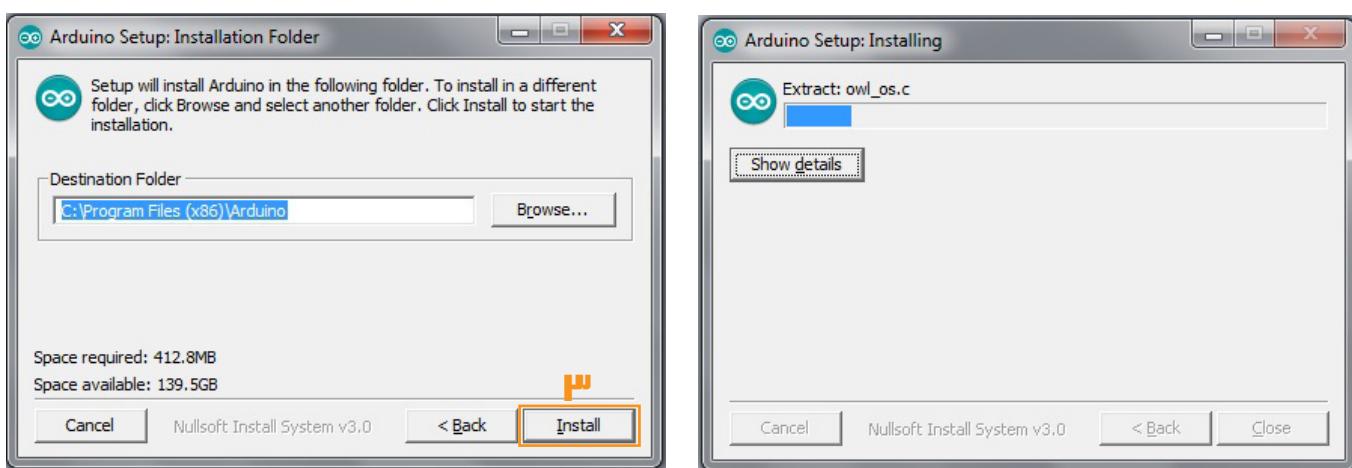
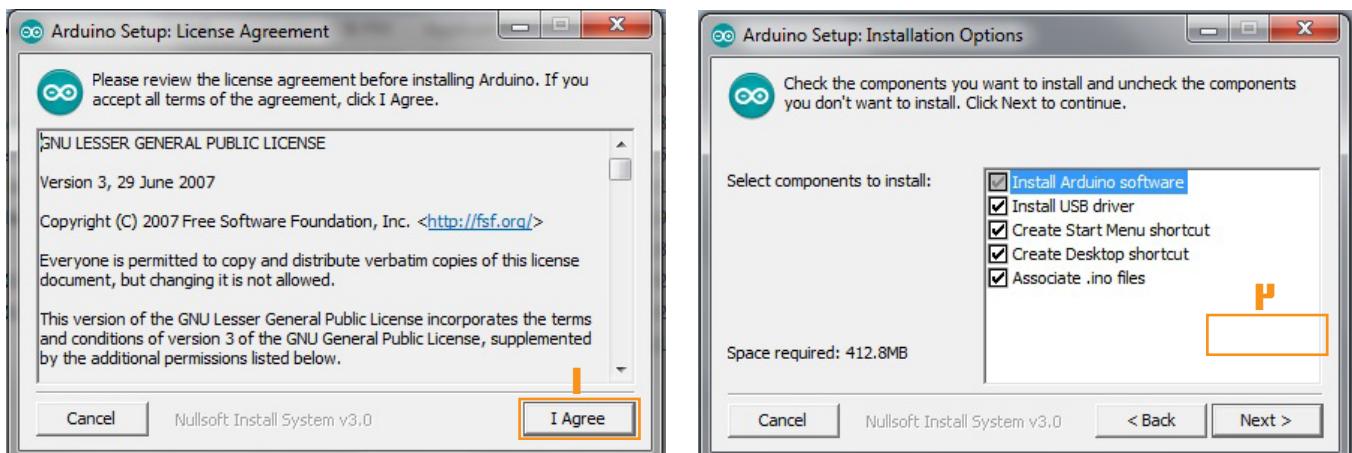
\$3 \$5 \$10 \$25 \$50 OTHER

JUST DOWNLOAD **CONTRIBUTE & DOWNLOAD**

A cartoon character with brown hair, wearing glasses and an orange shirt, stands on the right side of the page.

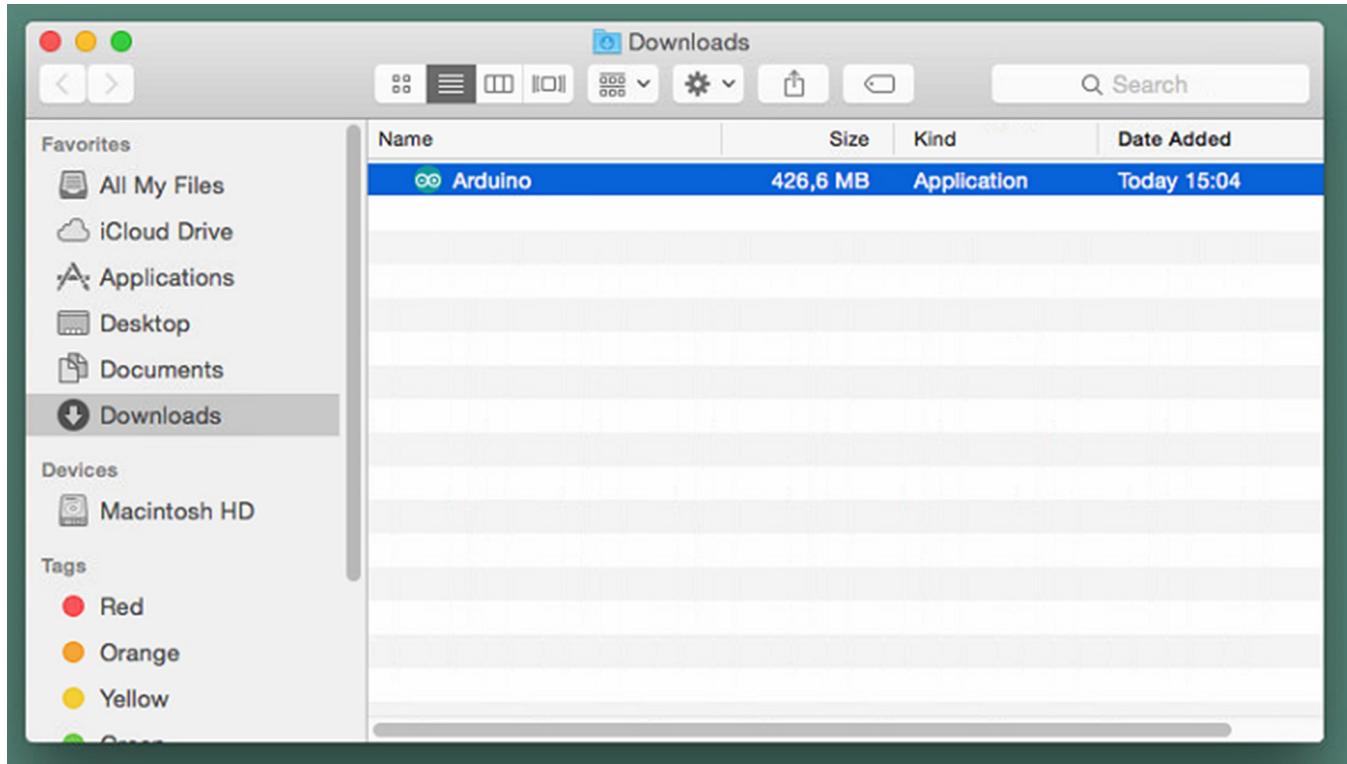
نصب Arduino IDE بر روی ویندوز

پس از دانلود نرم افزار (یا از درون دی دی) فایل arduino-1.8.4-windows.exe (یا فایل با نام متناظر نسخه جدیدتر) را اجرا کنید و طبق تنظیمات پیش فرض، آنرا بر روی کامپیوتر نصب کنید. با توجه به محدودیت های فایروال ویندوزهاي جديد، حتما با کليک راست بر روي آيكون نرم افزار، گزينه Run As Administrator را انتخاب کنيد. بسته به مشخصات کامپیوتر، ممکن است نصب نرم افزار تا ۱۵ دقیقه به طول بینجامد. لطفا تا پایان نصب صبور باشید:

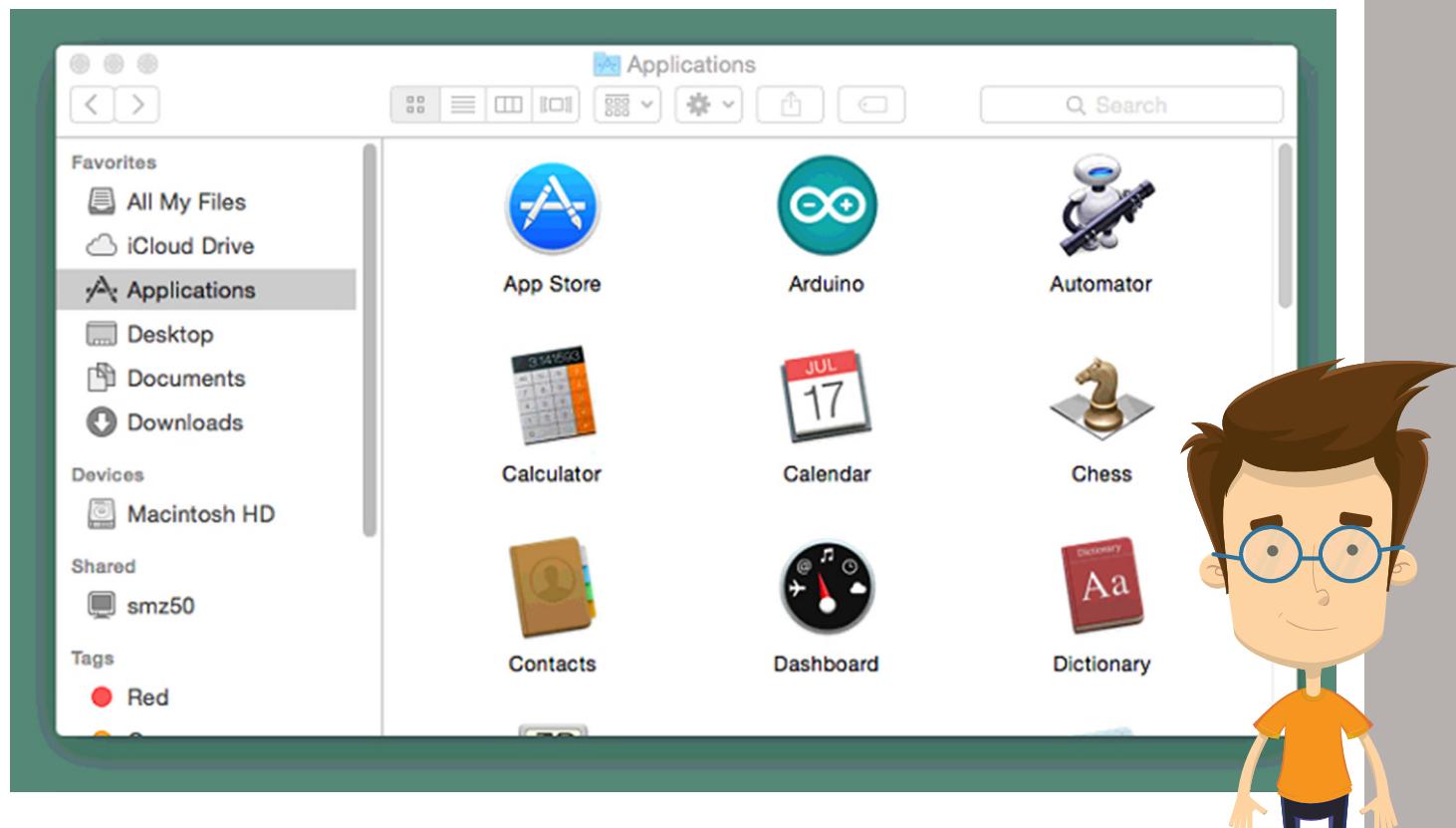


نصب MacOS بر روی Arduino IDE

اگر از مرورگر Safari برای دانلود فایل استفاده کرده باشید، فایل zip دانلود شده به صورت خودکار از حالت فشرده خارج می شود. اگر از مرورگر دیگری استفاده کرده باشید، باید این فایل را به صورت دستی از حالت فشرده خارج کنید.



را به پوشه نرم افزارها (Applications) یا هر پوشه دلخواه دیگر کپی کنید.



Arduino IDE محیط

Arduino IDE را باز کنید. در این مرحله قرار نیست برنامه نویسی انجام شود؛ بلکه هدف آشنایی با محیط نرم افزار و تنظیمات IDE برای شناسایی برد Arduino Uno است.



در ادامه به عملکرد هر کدام از بخش ها آشنا خواهید شد



محیط نرم افزار IDE

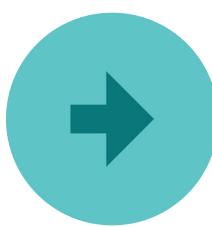
توضیحات محیط گرافیکی (GUI):

۱. دکمه Verify: کد را کامپایل و اعتبارسنجی می کند. در این مرحله ایرادات در نگارش کد مشخص می شود (به عنوان مثال فراموش کردن نقطه-ویرگول یا پرانتز)
۲. دکمه Upload: کد را به برد آردوینو می فرستد. هنگامی که بر این گزینه کلیک می کنید، چراغ های روی برد به سرعت چشمک می زند.
۳. دکمه New: این دکمه یک پنجره کدنویسی جدید باز می کند.
۴. دکمه Open: این دکمه به شما امکان می دهد یک Sketch موجود را باز کنید. (مجموعه کدهای آردوینو با نام Sketch شناخته می شوند)
۵. دکمه Save: این دکمه Sketch فعلی را ذخیره می کند.
۶. دکمه Serial Monitor: این گزینه یک پنجره باز می کند که تمامی اطلاعات سریال که توسط آردوینو مبادله می شود را نشان می دهد. این بخش برای رفع ایرادات از Sketch ها بسیار مفید است.
۷. بخش Sketch Name: این بخش نام Sketch که در حال حاضر بر روی آن مشغول فعالیت هستید را نشان می دهد.
۸. بخش Code Area: اینجا محلی است که کدهای تشکیل دهنده Sketch خود را می نویسید.
۹. بخش Message Area: این محلی است که IDE در صورت وجود خطا، به شما اعلام می کند. همچنین پیغام هایی در هنگام موفقیت کامپایل یا آپلود Sketch به برد در این بخش نشان داده می شود.

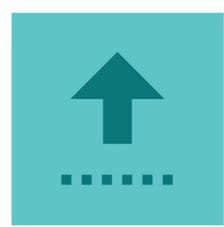
از میان موارد بالا، سه مورد بیشترین کاربرد را دارند:



Verify



Upload



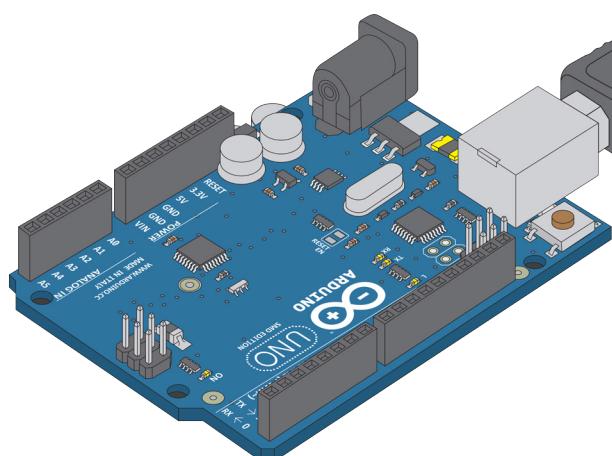
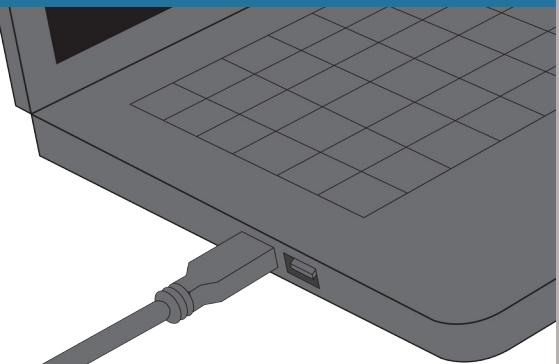
Open



تعریف برد آردوینو در IDE



بُرد آردوینو را توسط کابل USB به کامپیوتر متصل کنید



از منو Tools >Board گزینه Arduino Uno را انتخاب کنید

File Edit Sketch Tools Help

Auto Format Archive Sketch Fix Encoding & Reload Serial Monitor

Board

Arduino Uno

Serial Port

Programmer

Burn Bootloader

Arduino Uno

Arduino Duemilanove w/ ATmega328]

Arduino Diecimila or Duemilanove w/ ATmega168

Arduino Nano w/ ATmega328

Arduino Nano w/ ATmega168

Arduino Mega 2560 or Mega ADK

Arduino Mega (ATmega1280)

Arduino Mini

Arduino Mini w/ATmega168

Arduino Ethernet

Arduino Fio

Arduino BT w/ ATmega328

Arduino BT w/ATmega168

LilyPad Arduino w/ ATmega328

LilyPad Arduino w/ ATmega168

Arduino Pro or Pro Mini (5V, 16 MHz) w/ATmega328

Arduino Pro or Pro Mini (5V, 16 MHz) w/ATmega168

Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328

Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega168

Arduino NG or older w/ ATmega168

Arduino NG or older w/ ATmega8

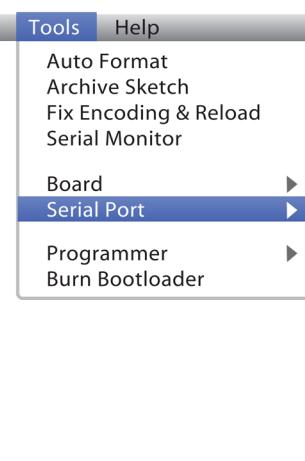
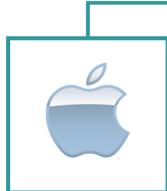


تنظیم پورت آردوینو در IDE



Windows

از منو Tools > Serial Port، پورت مربوط به آردوینو را انتخاب کنید. در سیستم عامل ویندوز، معمولاً این پورت COM3 یا مقداری بالاتر (به عنوان مثال COM1 و COM2) است (COM1 و COM2 برای پورت های سریال سخت افزار کامپیوتر رزرو شده اند). برای اینکه بدانید کدام مورد مربوط به برد آردوینو است، فیش USB آردوینو را جدا کرده و مجدداً منو را باز کنید. موردی که حذف شده است، پورت مورد نظر است.



MacOS

از منو Tools > Serial Port، پورت مربوط به آردوینو را انتخاب کنید. در سیستم عامل MacOS، برای Arduino Uno، این پورت شامل عبارت dev/tty.usbmodem/ است.

Linux

برای استفاده از آردوینو در سیستم عامل لینوکس، به آدرس زیر مراجعه نمایید:

<http://playground.arduino.cc/Learning/Linux>



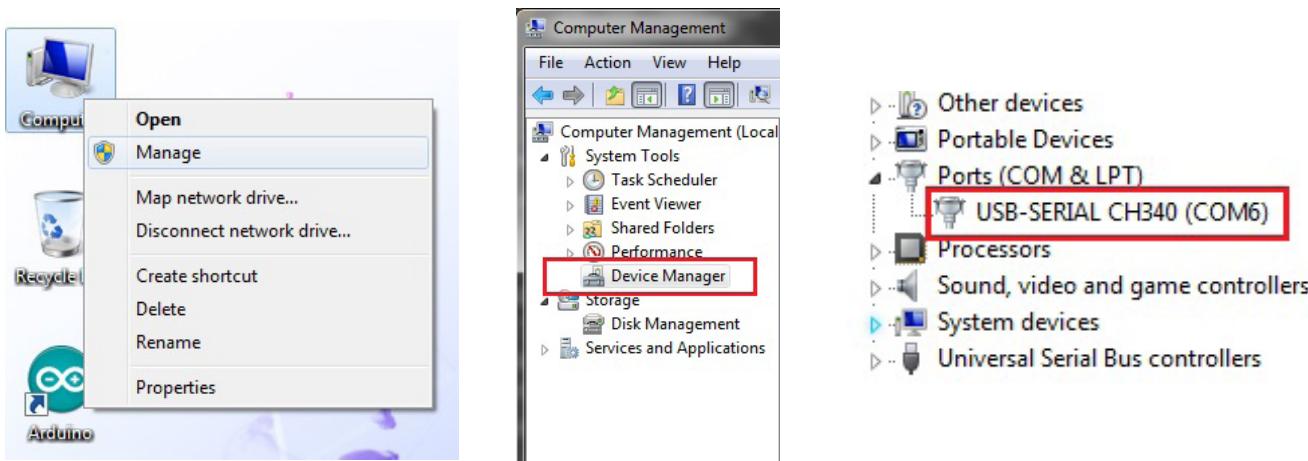
نصب درایور USB - ویندوز

در صورتی که پس از نصب نرم افزار و اتصال آردوینو به کامپیوتر از طریق کابل USB، بخش Port در Arduino IDE فعال نشد، باید درایور USB را نصب کنید. در پوشش Software در DVD همراه مجموعه دو فایل درایور برای USB ویندوز وجود دارد:

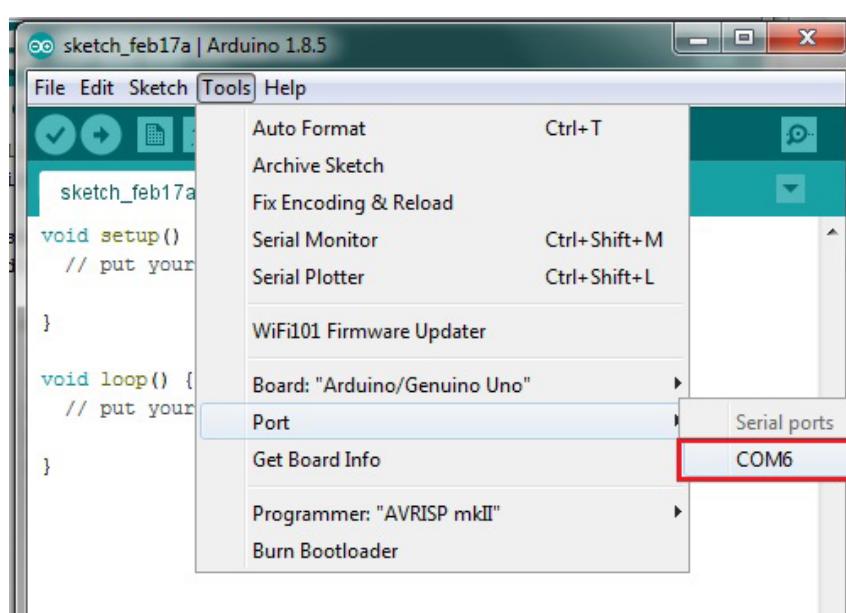
1. CDM_v2.12.00_WHQL_Certified (USB Driver).exe
2. CH34x_Install_Windows_v3_4(USB Driver).EXE

با کلیک راست بر روی فایل ها و انتخاب Run As Administrator درایورها را نصب کنید. برای اطمینان از نصب صحیح درایورها، هنگامی که آردوینو به کامپیوتر متصل است، با کلیک راست بر روی گزینه Manage Computer و سپس از منو سمت چپ گزینه Device Manager را انتخاب کنید. در صورتی که درایور به درستی نصب شده باشد، در زیر منو Port، عبارت زیر نمایش داده خواهد شد: USB-Serial CH340 (COM?)

که در آن ؟ عددی بزرگتر از ۲ خواهد بود (COM1 و COM2 مختص خود سیستم ویندوز هستند):



برای آپلود کد بر روی آردوینو، باید شماره پورت انتخابی در IDE با پورت نمایش داده شده در Device Manager یکسان باشد.



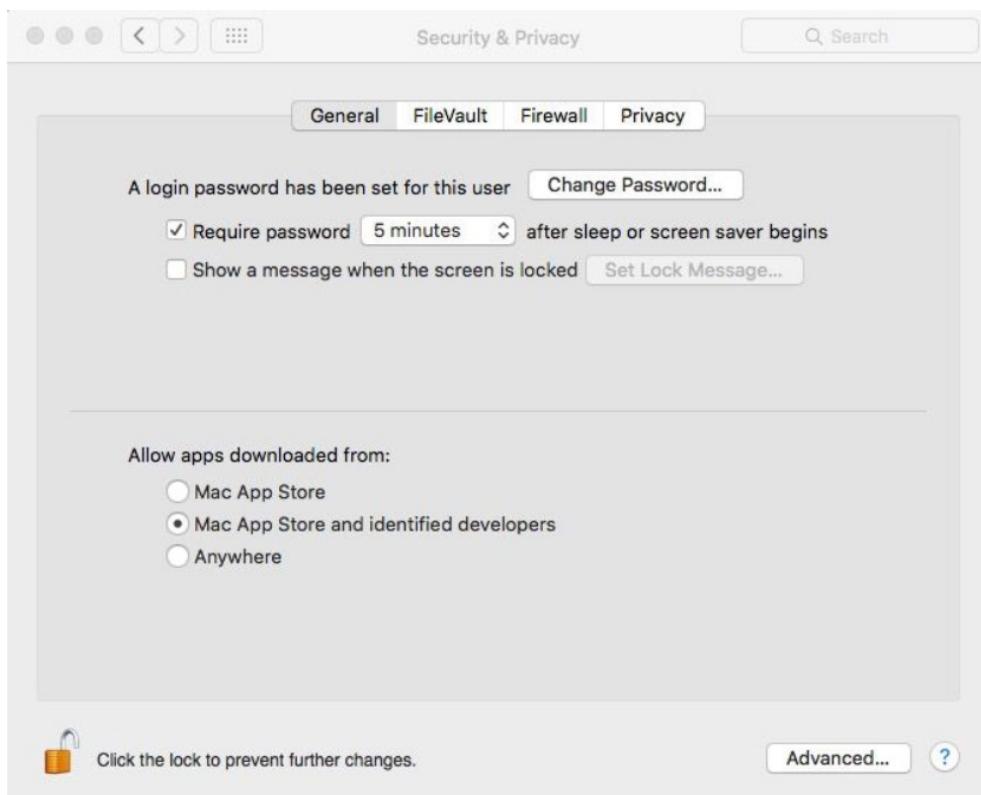
نصب درایور - MacOS

CH34x_Install_V1.4.pkg

بر روی فایل زیر در پوشه Software دبل کلیک کنید:

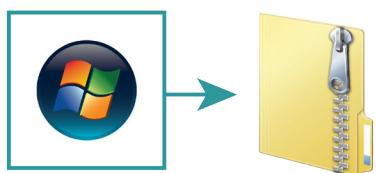
نصب را مطابق با توضیحات مرحله به مرحله انجام دهید. پس از پایان نصب کامپیوتر را ریستارت کنید.
پس از آن باید در device list، یک serial device با نام زیر ببینید:
list(/dev/tty.wchusbserial*)

اگر هنوز پورت را مشاهده نکردید به صورت زیر اقدام کنید:
وارد "System Preferences"→"Security & Privacy"→"General" شده و در زیر بخش مشخص شده "Mac App Store and identified developers" گزینه "Allow apps downloaded from" را توسط انتخاب کنید:

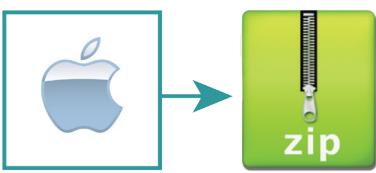


نصب کتابخانه قطعات

همانگونه که در برخی از پروژه ها خواهدی دید، برخی از قطعات، برای عملکرد صحیح نیاز به کتابخانه مخصوص خود دارند. این کتابخانه ها در واقع دستورات و عملکردهای آماده این قطعات است که برای سادگی و سهولت کدنویسی نوشته شده اند. برای نصب کتابخانه ها، محتوای درون پوشه libraries در DVD همراه مجموعه موجود است را در پوشه Libraries در محل نصب نرم افزار Arduino کپی کنید.



فایل libraries را از حالت فشرده خارج کنید.



در ویندوز های ۳۲ بیتی

```
C:\ → Program Files → Arduino → libraries
```

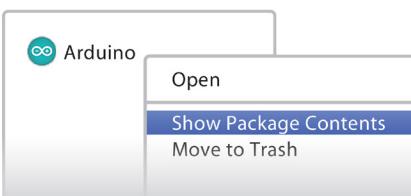
در ویندوز های ۶۴ بیتی

```
C:\ → Program Files(x86) → Arduino → libraries
```

محتوای پوشه libraries را در پوشه libraries کپی کنید



بر آیکون Arduino در پوشه نرم افزار ها راست کلیک (ctrl+click) کنید و گزینه Show Package Contents را انتخاب کنید



محتوای پوشه libraries را در پوشه libraries کپی کنید



استفاده از کدهای راهنمایی

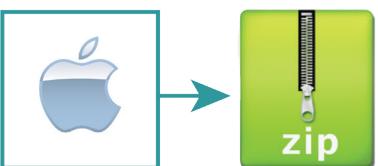
برای راهنمایی و رفع اشکالات احتمالی، کدهای آماده تمامی پروژه‌ها، هم از طریق دی وی دی همراه کیت و هم از طریق وب سایت کنکاش تک در دسترس است. دقت داشته باشید که این کدها تنها برای رفع اشکال است و برای یادگیری بهتر، باید خودتان کدها را مطابق با آموزش‌ها بنویسید. برای دانلود کدها به آدرس زیر مراجعه کنید:

www.kankashtech.com/electroduino

پس از دانلود کدهای راهنمایی، فایل دانلودی را از حالت فشرده خارج کنید. برای استفاده از کدهای راهنمایی باید پوشه Electoduino (که هم در دی وی دی و هم در فایل‌های دانلودی موجود است) را در پوشه آردوبینو کپی کنید. اگر نصب نرم افزار را مطابق با تنظیمات پیش فرض انجام داده باشید:



فایل Electoduino_Examples را از حالت فشرده خارج کنید. یک پوشه به نام Electoduino در آن وجود دارد



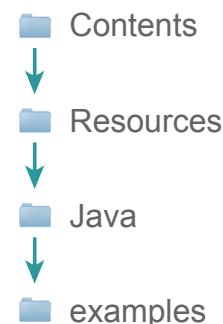
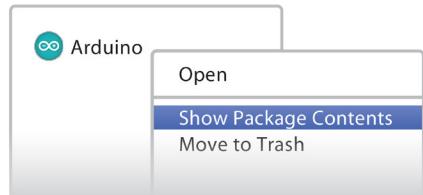
در ویندوز های ۳۲ بیتی
C:\ → Program Files → Arduino → examples

در ویندوز های ۶۴ بیتی
C:\ → Program Files(x86) → Arduino → examples

پوشه examples را در پوشه Electoduino کپی کنید



بر آیکون Arduino در پوشه نرم افزارها راست کلیک (ctrl+click) Show Package گزینه Contents را انتخاب کنید و گزینه Show Package را انتخاب کنید



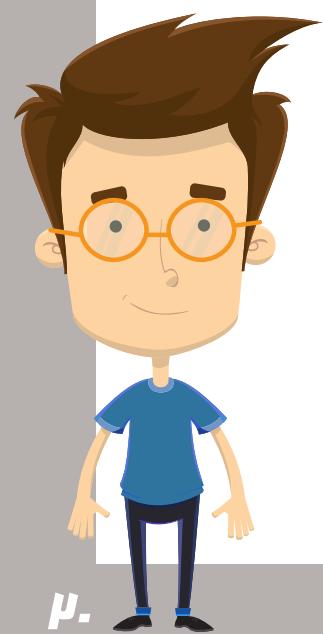
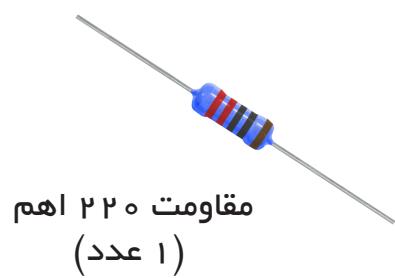
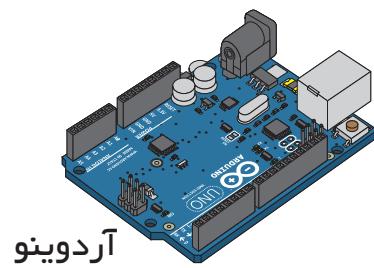
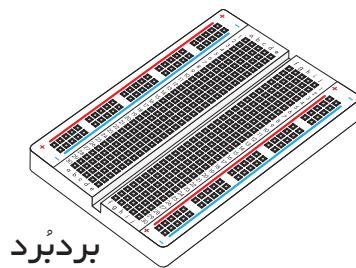
پوشه examples را در پوشه Electoduino کپی کنید



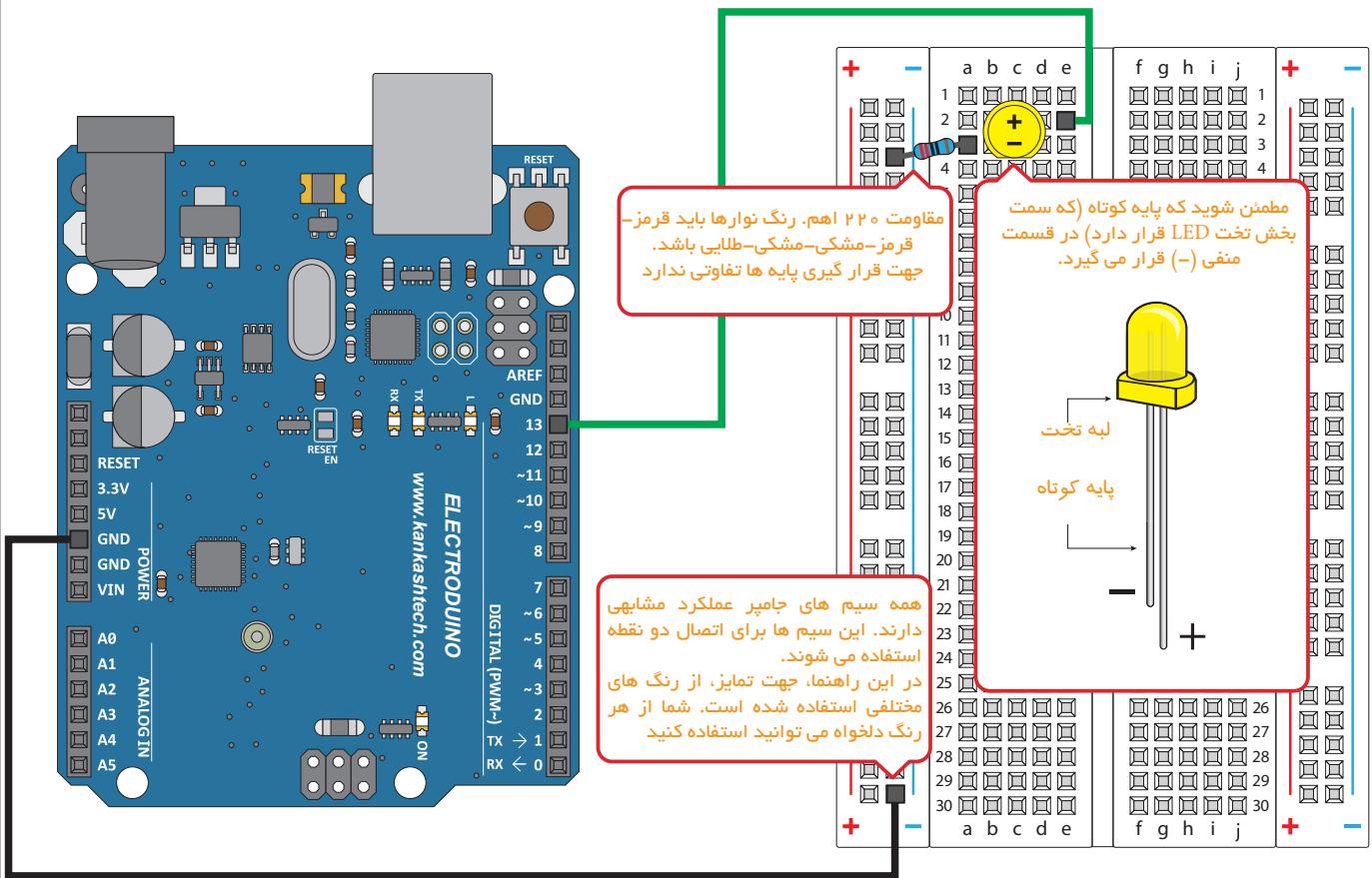
آل ای دی (LED) چشمک زن

چراغ هایی کوچک و پرقدرت هستند که در کاربردهای بسیار متنوعی استفاده می شوند. کار با الکترودوینو، با پروژه LED چشمک زن آغاز می شود. این کار به سادگی خاموش و روشن کردن یک چراغ است. اگرچه ممکن است کار ساده ای به نظر برسد، ولی یادگیری این مدار ساده، زیرساختی است که در پروژه های پیچیده بعدی به کار خواهد آمد.

مطلوبات



آل ای دی (LED) چشمک زن



توضیح مدار:

برای جلوگیری از آسیب دیدن LED در اثر عبور جریان زیاد، عموماً یک مقاومت میان پین های آردوینو و LED قرار می گیرد.

برای بستن مدار پروژه ۱، باید پایه مثبت LED (پایه بلند) را به پین ۱۳ متصل کرد. پایه منفی LED (پایه کوتاه) به یک پایه مقاومت ۲۲۰ اهم (قرمز-قرمز-قهقهه ای - طلایی) یا (قرمز-قرمز-مشکی-مشکی-طلایی) متصل می شود و پایه دیگر مقاومت در یکی از خانه های ستون منفی بردبورد قرار می گیرد. برای تامین اتصال منفی مدار، از یکی از خانه های ستون منفی بردبورد یک سیم جامپر به یکی از پین های GND آردوینو متصل می شود. برخلاف LED که جهت قرارگیری پایه های آنها مهم است، در استفاده از مقاومت این محدودیت وجود ندارد و می توان از مقاومت در هر جهتی استفاده کرد. دقت داشته باشید که هر دو پایه LED را در یک ردیف افقی بردبورد قرار ندهید. در شکل بالا، پایه مثبت LED در ردیف ۲ و پایه منفی در ردیف ۳ قرار گرفته است.



آل ای دی (LED) چشمک زن

چیست؟ Sketch

آردوینو یک کامپیوترا کوچک است که برنامه هایی با نام "sketch" با پسوند ".ino" را اجرا می کند. sketch فایلی متنی است که با استفاده از دستور العمل هایی که کامپیوترا می فهمد، نوشته شده است. sketch حاوی کدهای کامپیوترا و همچنین "comment" هایی است که عملکرد کدها را توضیح می دهد. رنگ comment ها و کدها در IDE متفاوت بوده تا بتوان آنها را از هم تشخیص داد. نرم افزار هر چیزی که بعد از // آمده باشد comment را فرض کرده و از آن چشم پوشی می کند. به عنوان مثال: // This is a comment

comment می تواند بیش از یک خط باشد. در این حالت باید در میان /* */ قرار گیرد. به عنوان مثال: /* This is also a comment.

This comment can be multi-line. */

تابع (Function)

تابع یا function، یک مجموعه کد است که عملکردی خاصی دارد. بسیاری از توابع مفید به صورت پیش فرض در نرم افزار آردوینو وجود دارند که به سادگی می تواند فراخوانده شود. همچنین در صورت نیاز می توان عملکرد خاصی را به صورت تابع نوشت.

همه sketch ها، دو تابع داخلی آردوینو را باید داشته باشند: تابع setup() و تابع loop().

تابع setup()

تابع setup() هنگام راه اندازی آردوینو یا در صورت فشرده شدن دکمه reset فقط یکبار اجرا می شود. این تابع برای انجام کارهایی که اول از همه باید انجام شوند یا برای کارهایی که تنها یکبار باید انجام شوند استفاده می شود.

تابع loop()

پس از پایان کار تابع setup()، تابع loop() به صورت مکرر اجرا می شود و به جز در صورت خاموش شدن آردوینو یا فشرده شدن دکمه reset، این تابع بدون توقف اجرا می شود. معمولا کاری که قرار است آردوینو انجام دهد در این بخش قرار می گیرد.



ال ای دی (LED) چشمک زن

تابع (pinMode)

آردوینو 12 پین دیجیتال ورودی/خروجی دارد. این پین ها یا به صورت ورودی (INPUT) یا خروجی (OUTPUT) تعریف می شوند. تنظیم پین ها توسط دستور داخلی (pinMode) انجام می شود.

تابع (pinMode) دو پارامتر می گیرد که در پرانتز بعد از نام تابع قرار داده می شوند. پارامتر اول شماره پین و پارامتر دوم یکی از دو مقدار INPUT یا OUTPUT است. دقیت داشته باشید که IDE به حروف بزرگ و کوچک حساس است.

`pinMode(13,OUTPUT);`

.....

تابع (digitalWrite)

۱۲ پین دیجیتال آردوینو برای اعمال سیگنال های خاموش/روشن یا "دیجیتال" استفاده می شوند. این سیگنال ها یا ۵ ولت (HIGH) یا صفر ولت (LOW) هستند.

تابع داخلی آردوینو است که برای تنظیم یک پین به حالت HIGH یا LOW استفاده می شود. این تابع دو پارامتر شماره پین و یکی از دو مقدار HIGH یا LOW را به عنوان ورودی می گیرد:

`digitalWrite(13,HIGH);`

.....

تابع (delay)

تابع delay می گیرد که ادامه اجرای دستورات را برای زمان مشخص متوقف می کند. این تابع یک مقدار می گیرد که آن میزان زمان انتظار در واحد میلی ثانیه (یک هزارم ثانیه) است. به عنوان مثال برای نگه داشتن دستورات به مدت یک ثانیه:

`delay(1000);`



آل ای دی چشمک زن (LED)

File > Examples > Electroduino > Circuit_1



Circuit_01



```
void setup() {
```

پین ۱۳ به عنوان خروجی (OUTPUT) تنظیم می شود تا از طریق این پین ولتاژ را ار آردوبینو به LED برسد. این توضیح را می توان به صورت comment در متن sketch قرار داد.

```
pinMode (13,OUTPUT); // Set pin 13 to be an output
```

```
}
```

```
void loop() {
```

از آنجایی که یک LED به پین ۱۳ متصل شده است، اگر خروجی این پین را به صورت HIGH تنظیم شود، LED روشن شده و اگر به صورت LOW تنظیم شود، LED خاموش می شود.

```
digitalWrite(13,HIGH); // Turn on the LED  
delay(1000); // Wait for one second  
digitalWrite(13, LOW); // Turn off the LED  
delay(1000); // Wait for one second
```

```
}
```

پس از اعتبارسنجی (Verify) و آپلود sketch به برد آردوبینو، LED باید شروع به چشمک زدن با فاصله زمانی ۱ ثانیه کند.
عدد مربوط به زمان انتظار (۱۰۰۰ در کد بالا) را تغییر داده و مجدداً کد را به آردوبینو آپلود کرده و تغییرات را مشاهده کنید





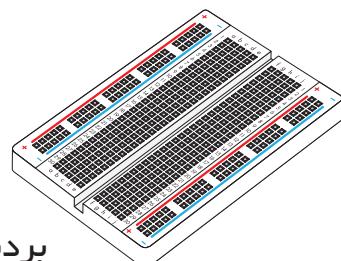
کار با چند LED

کار با چند LED به شیوه ای مشابه کار با یک LED پروژه ۱ انجام می شود.
در این پروژه برای مشاهده شباهت عملکرد پین های آردوینو، از پین های ۲ الی ۹ برای روشن و خاموش کردن ۸ عدد LED استفاده می شود.
کدنویسی این پروژه یکبار با استفاده از دستورات مشابه پروژه ۱ (مدار ۱-۲) و بار دیگر با استفاده از حلقه و آرایه (مدار ۲-۲) انجام می شود.

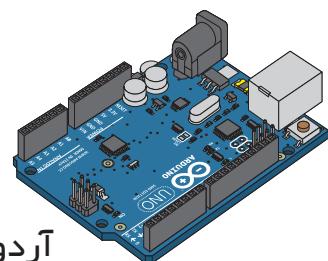
مطلوبات



سیم جامپر نری-نری
(۹ عدد)



بردبرد



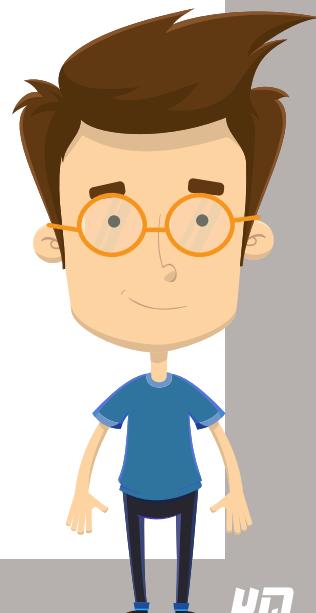
آردوینو



مقاومت ۲۲۰ اهم
(۸ عدد)

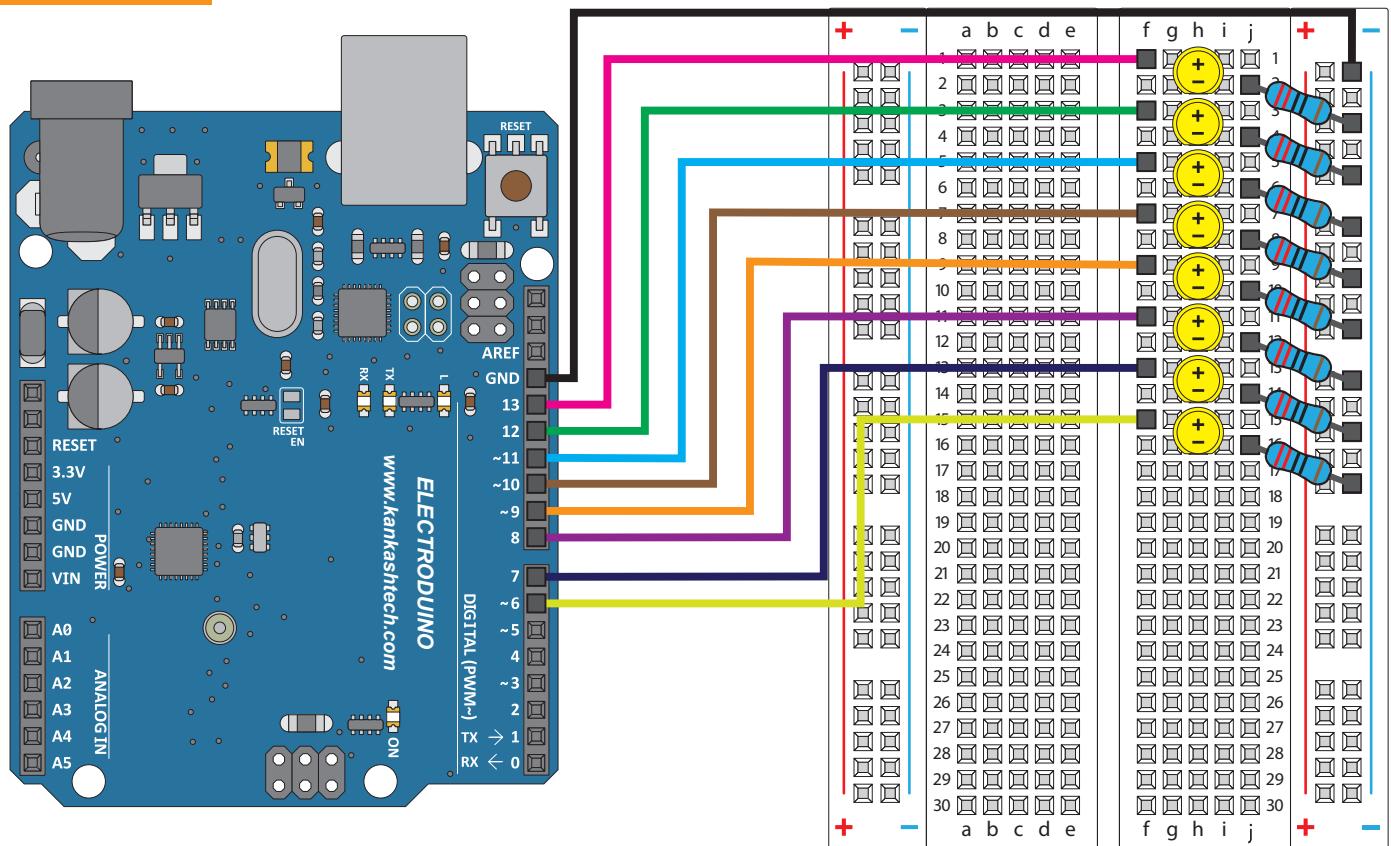


LED
(۸ عدد)



کار با چند LED

P



توضیح مدار:

مانند مدار ۱، برای جلوگیری از آسیب دیدن LED در اثر عبور جریان زیاد، یک مقاومت ۲۰۰ اهم میان پین منفی (GND) آردوینو و پایه های منفی LEDها قرار می گیرد.

برای بستن مدار پروژه ۲، باید پایه مثبت LEDها (پایه بلند) را به پین ۶ الی ۱۳ آردوینو متصل کرد. پایه های منفی LEDها (پایه کوتاه) به یک پایه مقاومت ۲۰۰ اهم (قرمز- قرمز- مشکی- مشکی- طلایی) متصل می شود و پایه دیگر مقاومت در یکی از خانه های منفی برد بُرد قرار می گیرد. از یکی از خانه های ستون منفی برد بُرد، یک سیم جامپر به یکی از پین های GND آردوینو متصل می شود تا اتصال منفی مدار برق را گردد.
دقت داشته باشید که هر دو پایه یک LED در یک ردیف افقی برد بُرد قرار نگیرد.



کار با چند LED

متغیرها (Variables)

متغیرها اعدادی هستند که برای آنها نام تعریف می‌شود. عموماً از متغیرها برای نگهداری مقادیری که در طول برنامه تغییر می‌کنند، مانند اندازه گیری‌های دنیای بیرونی استفاده می‌شود. همچنین برای درک بهتر sketch‌ها نیز از متغیر استفاده می‌شود زیرا خواندن یک نام توصیفی درک بهتری از یک عدد ایجاد می‌کند. متغیرها می‌توانند داده‌هایی با نوع متفاوت داشته باشند. به عنوان مثال اینکه می‌تواند منفی شود؟ می‌تواند اعشار داشته باشد؟ و ...

یکی از پرکاربردترین انواع متغیر، عدد صحیح است که توسط int تعریف می‌شود. این متغیر می‌تواند اعداد صحیح مثبت و منفی بگیرد.

پیش از استفاده از یک متغیر باید آنرا تعریف کرد تا کامپیوتر درباره آن بداند. وقت داشته باشید که نام متغیرها به حروف بزرگ و کوچک حساس است و اگر با خطای " در هنگام کامپایل کد مواجه شدید، نام متغیرهای تعريف شده و استفاده شده را چک کنید.

در پروژه ۱ دیدید که برای تعریف پین به صورت زیر عمل شد:

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

در پروژه‌های ساده، اینگونه کدنویسی مشکلی ایجاد نمی‌کند. ولی در صورتی که مدار پیچیدگی بیشتری داشته باشد، استفاده از شماره پین‌ها ضمن دشواری کدنویسی، احتمال بروز خطا را بیشتر می‌کند.

برای جلوگیری از این ایراد می‌توان به صورت زیر عمل کرد:

```
int ledPin = 13;
```

```
pinMode (ledPin, OUTPUT);
```

کدهای فوق دقیقاً مشابه هم عمل می‌کنند.

نکته دیگر در تعریف متغیرها، محل تعریف است. اگر متغیری بیرون از تابع تعریف شود، این متغیر "global" نامیده می‌شود و در تمام توابع یک sketch می‌توان از آن استفاده کرد و تغییر آن در هر تابع، در تمامی توابع لحظه می‌شود. در صورتی که متغیری درون یک تابع تعریف شود، فقط در آن تابع قابل دسترس است و در صورتی که بدون تعریف مجدد در توابع دیگر استفاده شود، خطا ایجاد می‌شود.

در صورت تعریف صحیح متغیر، می‌توان برای خاموش و روشن کردن LED، به جای شماره پین ۱۳ از متغیر ledPin استفاده کرد:

```
digitalWrite (ledPin, HIGH);
```

در این پروژه از ۸ LED زرد با نام‌های LED1-LED8 استفاده شده است.



کار با چند LED



آرایه (Array)

آرایه این امکان را ایجاد می کند که چندین متغیر را در یک جا ذخیره کرده و با توجه به موقعیت یا ایندکس (index)، متغیر دلخواه را فراخوان کرد.

به عنوان مثال به جای ایجاد ۸ متغیر برای ۸ عدد LED متصل به پین های ۶ الی ۱۳ در این پروژه، می توان از یک آرایه به صورت زیر استفاده کرد:

```
int ledPin[] = {13, 12, 11, 10, 9, 8, 7, 6};
```

موقعیت یا index آرایه ها از عدد ۰ شروع می شود. به عنوان مثال در آرایه بالا، عدد ۱۳ در ایندکس ۰، عدد ۱۲ در ایندکس ۱ و ... قرار گرفته اند. آخرین ایندکس آرایه بالا ۷ است که نگهدارنده عدد ۶ است. به عنوان مثال در مدار این پروژه، برای روشن کردن چهارمین LED از بالا (متصل به پین شماره ۰ آردوینو) باید از دستور زیر استفاده کرد:

```
digitalWrite (ledPin[3], HIGH);
```

می توان ایندکس یک آرایه را به صورت یک متغیر تعریف کرد که بتوان در طول برنامه آنرا تغییر داد. به عنوان مثال عملکرد کد زیر مشابه کد بالاست:

```
int index = 3;
```

```
digitalWrite (ledPin[index], HIGH);
```

حلقه for

حلقه for برای تغییر یک متغیر از مقداری به مقدار دیگر و اجرای مجموعه ای کد با مقدار جدید استفاده می شود. در واقع در هر مرحله یک سری کد اجرا شده، متغیر یا متغیرهایی تغییر کرده و کدها مجدداً اجرا می شوند. این حلقه تا زمانی که شرط حلقه برقرار باشد بدون وقفه اجرا می شود. هر حلقه for سه بخش دارد که توسط نقطه ویرگول ";" از هم جدا می شوند:

۱. کاری که پیش از شروع حلقه انجام می شود؛
۲. شرطی که باید بررسی شود؛ تا زمانی که جواب این تست مثبت است، حلقه ادامه می یابد (کدهای میان {} اجرا می شوند)
۳. کاری که بعد از هر حلقه انجام می شود (معمولاً افزایش مقدار یک متغیر است)

به عنوان مثال برای روشن و خاموش کردن به ترتیب LEDها:

```
for (index=0; index<8; index++) {
```

۱

۲

۳

```
    digitalWrite (ledPin[index], HIGH);
    delay(1000);
    digitalWrite (ledPin[index], LOW);
    delay(1000);
}
```



تاربا چند LED



File > Examples > Electroduino > Circuit_2_1

```
int LED1=13;
int LED2=12;
int LED3=11;
int LED4=10;
int LED5=9;
int LED6=8;
int LED7=7;
int LED8=6;

void setup() {

    pinMode(LED1,OUTPUT);
    pinMode(LED2,OUTPUT);
    pinMode(LED3,OUTPUT);
    pinMode(LED4,OUTPUT);
    pinMode(LED5,OUTPUT);
    pinMode(LED6,OUTPUT);
    pinMode(LED7,OUTPUT);
    pinMode(LED8,OUTPUT);

}
```

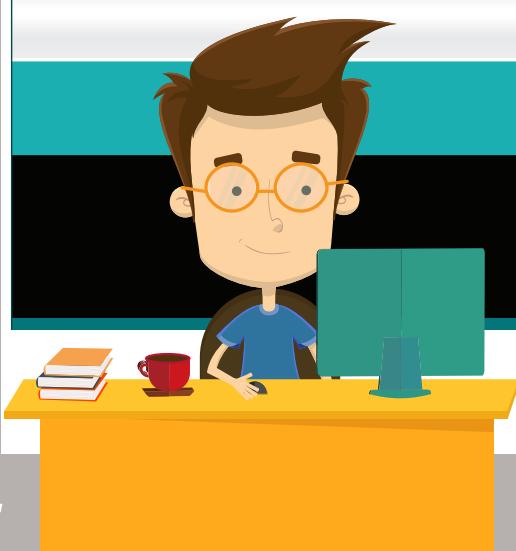
// Set pin 2 to be an output
// Set pin 3 to be an output
// Set pin 4 to be an output
// Set pin 5 to be an output
// Set pin 6 to be an output
// Set pin 7 to be an output
// Set pin 8 to be an output
// Set pin 9 to be an output





Circuit_2_1

```
void loop() {
    digitalWrite(LED1,HIGH);      // Turn on the LED1
    delay(1000);                // Wait for one second
    digitalWrite(LED1, LOW);     // Turn off the LED1
    delay(1000);                // Wait for one second
    digitalWrite(LED2,HIGH);      // Turn on the LED2
    delay(1000);                // Wait for one second
    digitalWrite(LED2, LOW);     // Turn off the LED2
    delay(1000);                // Wait for one second
    digitalWrite(LED3,HIGH);      // Turn on the LED3
    delay(1000);                // Wait for one second
    digitalWrite(LED3, LOW);     // Turn off the LED3
    delay(1000);                // Wait for one second
    digitalWrite(LED4,HIGH);      // Turn on the LED4
    delay(1000);                // Wait for one second
    digitalWrite(LED4, LOW);     // Turn off the LED4
    delay(1000);                // Wait for one second
    digitalWrite(LED5,HIGH);      // Turn on the LED5
    delay(1000);                // Wait for one second
    digitalWrite(LED5, LOW);     // Turn off the LED5
    delay(1000);                // Wait for one second
```



تاربا چند LED



File > Examples > Electroduino > Circuit_2_1



Circuit_2_1

```
digitalWrite(LED6,HIGH);      // Turn on the LED6
delay(1000);                  // Wait for one second
digitalWrite(LED6, LOW);       // Turn off the LED6
delay(1000);                  // Wait for one second
digitalWrite(LED7,HIGH);      // Turn on the LED7
delay(1000);                  // Wait for one second
digitalWrite(LED7, LOW);       // Turn off the LED7
delay(1000);                  // Wait for one second
digitalWrite(LED8,HIGH);      // Turn on the LED8
delay(1000);                  // Wait for one second
digitalWrite(LED8, LOW);       // Turn off the LED8
delay(1000);                  // Wait for one second

}
```



تار با چند LED



File > Examples > Electroduino > Circuit_2_2



Circuit_2_2

```
int ledPin[] = {13, 12, 11, 10, 9, 8, 7, 6};  
int index=0;
```

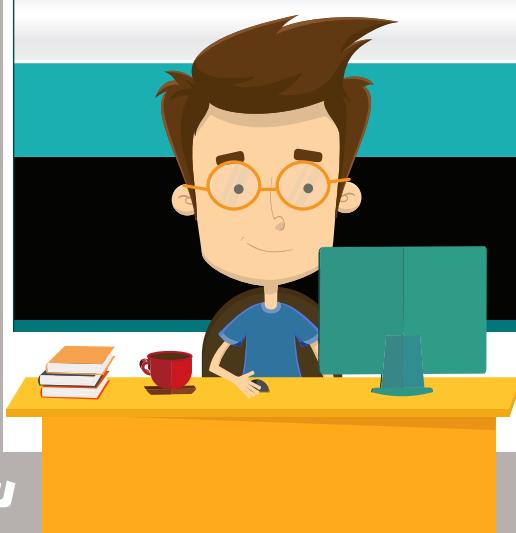
متغیرهای index و ledPin خارج از توابع و به صورت "global" تعریف شده تا در تمام توابع sketch قابل دسترسی و ایجاد تغییر باشند.

```
void setup() {  
    for (index=0; index<8; index++) {  
        pinMode(ledPin[index], OUTPUT);  
    }  
}
```

با استفاده از حلقه for، پین های ۲ الی ۹ آردوینو به صورت خروجی (OUTPUT) تعریف می شوند. همانطور که مشاهده می کنید پیش از آغاز حلقه، متغیر index صفر می شود و در هر مرحله ۱ واحد به آن افزوده می شود.

```
void loop() {  
    for (index=0; index<8; index++) {  
        digitalWrite(ledPin[index], HIGH);  
        delay(1000);  
        digitalWrite(ledPin[index], LOW);  
        delay(1000);  
    }  
}
```

با استفاده از حلقه for، با فاصله زمانی یک ثانیه، هر LED روشن شده و خاموش می شود و پس از ۱ ثانیه LED بعدی روشن و خاموش می شود. پس از رسیدن index به ۸، حلقه for خاتمه می یابد. با توجه به اینکه تابع loop به طور دائم تکرار می شود، مجدداً حلقه for اجرا می شود. (index صفر می شود و در هر مرحله یک واحد به آن افزوده می شود).





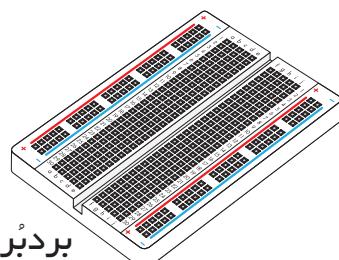
دکمه های فشاری

تاکنون فقط به خروجی‌ها (OUTPUT) پرداخته شد. در این پروژه از قابلیت دریافت ورودی (INPUT) توسط آردوینو و عملکرد متناظر با خروجی دریافتی استفاده می‌شود. برای این کار، از یکی از رایج‌ترین شیوه‌های ایجاد سیگنال ورودی یعنی دکمه‌های فشاری (Push Buttons) استفاده می‌شود. روش استفاده از دکمه فشاری در این پروژه بدین گونه است که با فشردن دکمه اول، LED اول روشن شده و در صورت فشردن دکمه دوم، LED دوم روشن می‌شود. در صورت فشردن همزمان دو دکمه، LED‌ها شروع به چشمک زدن می‌کنند.

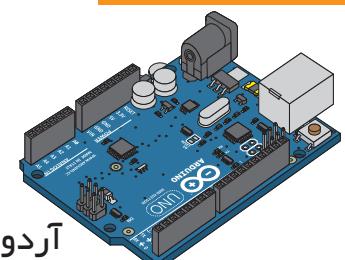
مطلوبات



سیم جامپر نری-نری
(۹ عدد)



برد



آردوینو



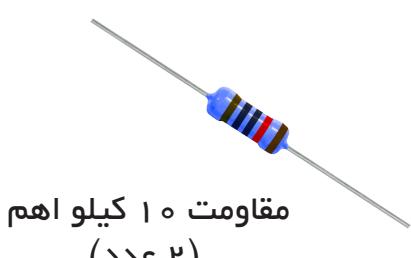
دکمه فشاری
(۲ عدد)



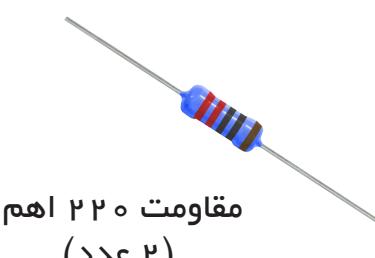
LED قرمز
(۱ عدد)



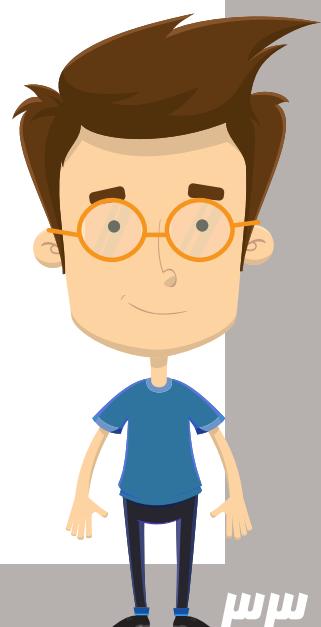
LED زرد
(۱ عدد)

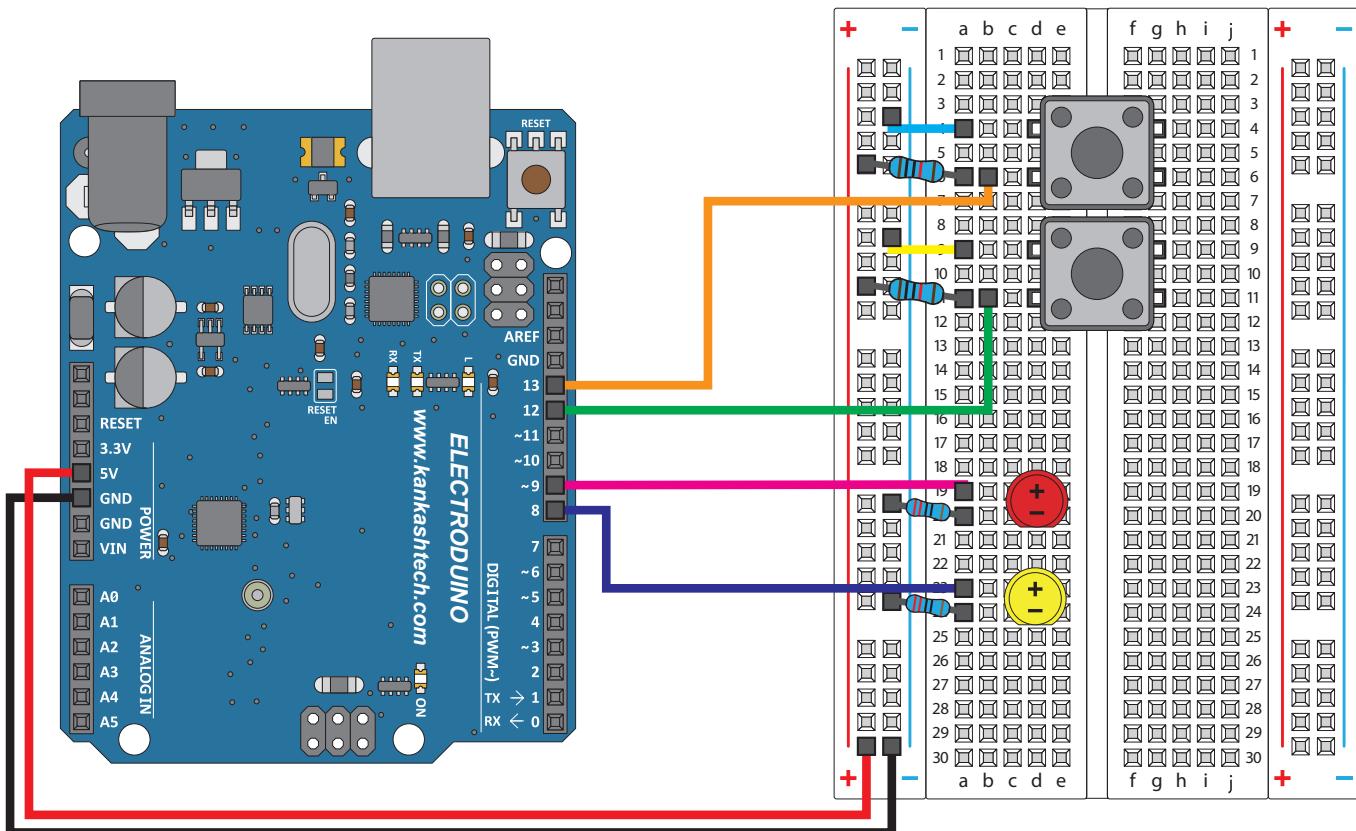


مقاومت ۱۰ کیلو اهم
(۲ عدد)



مقاومت ۲۲۰ اهم
(۲ عدد)

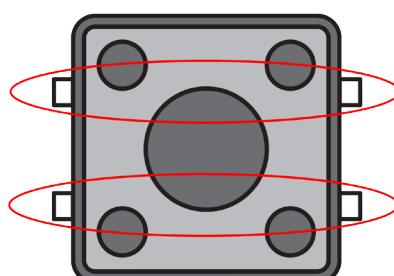




توضیح مدار

دکمه های فشاری در این مدار به عنوان تولیدکننده سیگنال ورودی استفاده می شوند. در واقع روشن و خاموش شدن LED ها به علت قطع و وصل جریان توسط دکمه ها اتفاق نمی افتد بلکه در اثر وضعیت های مختلف دکمه ها، دستورات متفاوتی به بین های متصل به LED ارسال می شود.

دکمه های موجود در الکترودوینو دو جفت پایه دارند که پایه های دور از هم (که با قادر قرمز مشخص شده اند) به صورت دائم به یکدیگر وصل بوده و پایه های نزدیک به هم در هنگام فشرده شدن دکمه به یکدیگر وصل می شوند. در واقع در هنگام فشردن دکمه، دو پایه بالا به دو پایه پایین متصل می شوند.



مدار پروژه ۳ از دو بخش تشکیل شده است، یک بخش حاوی LED و مشابه مدارهای قبلی است که در آن دو LED توسط پین های ۸ و ۹ خاموش و روشن می شوند.

برای بستن این بخش، باید پایه مثبت LED زرد به پین ۸ و LED قرمز به پین ۹ متصل می شود. پایه منفی LEDها به یک پایه مقاومت ۲۰ اهم (قرمز-قرمز-قهقهه ای - طلای) متصل می شود و پایه دیگر مقاومت در یکی از خانه های ستون منفی برد بُرد که به پین GND متصل شده، قرار می گیرد. بخش دوم مدار، داده های ورودی جهت تصمیم گیری را برای آردوینو ایجاد می کند. در این بخش از دو عدد دکمه فشاری استفاده می شود که به پین های ۱۲ و ۱۳ متصل می شوند.

مقاومت Pull-Up

عملکرد دکمه ها در این مدار بدین گونه خواهد بود که در اثر فشرده شدن، پین متصل به دکمه به GND متصل می شود و مقدار LOW را می خواند. حال سوال این است که اگر دکمه فشرده نشده باشد،

و در این حالت چیزی به پین متصل نباشد، پین مورد نظر HIGH می خواند یا LOW؟ برای جلوگیری از این اتفاق، مدار به گونه ای طراحی می شود که پین حتما در یکی از دو وضعیت HIGH یا LOW قرار بگیرد. برای این کار از مقاومت Pull-Up استفاده می شود.

برای ایجاد وضعیت HIGH در وضعیت فشرده نشده دکمه، از یکی از خروجی های ۵ ولت آردوینو، به یکی از ستون های مثبت برد بُرد، ولتاژ رسانده می شود. یک پایه مقاومت ۱۰ کیلو اهم در ستون مثبت و پایه دیگر در ردیف پین ورودی ۱۲ قرار می گیرد. همین کار برای پین ۱۳ تکرار می شود. نتیجه این مدار بدین گونه است:

در حالت فشرده نشده دکمه متصل به پین ۱۲، این پین از طریق مقاومت ۱۰ کیلو اهم به ۵ ولت متصل می شود. پین ۱۲ ولتاژی نزدیک به ۵ ولت می خواند و مقدار HIGH را بر میگرداند. همین اتفاق برای دکمه متصل به پین ۱۳ نیز می افتد.

در صورت فشرده شدن دکمه، پین به GND متصل می شود و جریان به جای حرکت به سمت پین، به سمت GND می رود و از آنجایی که پین به طور مستقیم به GND متصل می شود، مقدار LOW را می خواند.

اگر مقاومت ۱۰ کیلواهم در مدار استفاده نشده باشد، در هنگام فشرده شدن دکمه، پین ۵ ولت آردوینو مستقیما با GND متصل شده و اتصال کوتاه رخ می دهد و ممکن است به برد صدمه وارد شود.





INPUT

پین های دیجیتال آروینو می توانند به عنوان ورودی هم استفاده شوند. برای اینکار لازم است به آردوینو گفته شود که چه استفاده ای از پین می خواهید کنید. به عنوان مثال اگر دکمه فشاری به پین ۱۳ متصل شده باشد:

```
int Button1=13;
pinMode (Button1, INPUT);
```

digitalRead

برای خواندن وضعیت پین های دیجیتال از تابع `digitalRead()` استفاده می شود. اگر به پین ولتاژ ۵ ولت اعمال شده باشد، این تابع مقدار `HIGH` و اگر ولتاژ ۰ به پین اعمال شده باشد، این تابع مقدار `LOW` را بر می گرداند. از آنجایی که آردوینو HIGH را "۱" و LOW را "۰" می بیند، می توان برای استفاده از وضعیت پین ورودی، وضعیت را در یک متغیر از نوع `int` ذخیره کرد:

```
int state1;
state1 = digitalRead(Button1);
```

یک متغیر را می توان در همان زمان تعریف مقدار دهی کرد. به عنوان مثال کد بالا را می توان بدین صورت بازنویسی کرد:

```
int state1 = digitalRead(Button1);
```

if ...else

یکی از دستوراتی که در مدارهای دارای تصمیم گیری بسیار کاربرد دارد، گزاره های شرطی `if...else` است. این دستور از یک شرط و یک سری کد برای حالت برقرار بودن شرط و یک سری کد برای حالت عدم برقراری شرط تشکیل می شود. **البته استفاده از کدهای حالت عدم برقراری شرط اجباری نیست.**

برای نوشتن شرط های مختلف از عملگرهای منطقی استفاده می شود. عبارت های شرطی خروجی `True/False` دارند که `True` برای حالت برقراری و `False` برای حالت عدم برقراری است
به جز عملگرهای ریاضی (`>`, `<`, `=`, `<=`), عملگر های دیگری نیز وجود دارند که از مهمترین آنها:

`==` : برابر؛ شرط `A==B` است اگر `A` و `B` یکسان باشند.

`!=` : نابرابر؛ شرط `A!=B` برقرار است اگر `A` و `B` یکسان نباشند.

`&&` : شرط `A&&B` برقرار است اگر هر دو شرط `A` و `B` برقرار باشند.

`||` : شرط `A || B` برقرار است اگر یکی از دو شرط `A` یا `B` برقرار باشد.

`!` : `NOT A`؛ شرط `!A` برقرار است اگر `A` برقرار نباشد و برقرار نیست اگر `A` برقرار باشد.



File > Examples > Electroduino > Circuit_3

```
int LEDR = 9;
int LEDY = 8;
int Button1 = 12;
int Button2 = 13;
void setup() {
    pinMode (LED1,OUTPUT);
    pinMode (LED2,OUTPUT);
    pinMode (Button1,INPUT);
    pinMode (Button2,INPUT);
}

void loop() {
    int state1= digitalRead(Button1);
    int state2= digitalRead(Button2);

    if ((state1 == HIGH) && (state2 == HIGH)) {
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, LOW);
    }
}

// ادامه در صفحه بعد
```





File > Examples > Electroduino > Circuit_3

```
Circuit_3

if ((state1 == HIGH) && (state2 == LOW)) {
    digitalWrite(LED_R, HIGH);
    digitalWrite(LED_Y, LOW);
}

if ((state1 == LOW) && (state2 == HIGH)) {
    digitalWrite(LED_R, LOW);
    digitalWrite(LED_Y, HIGH);
}

if ((state1 == LOW) && (state2 == LOW)) {
    digitalWrite(LED_R, HIGH);
    digitalWrite(LED_Y, LOW);
    delay(200);
    digitalWrite(LED_R, LOW);
    digitalWrite(LED_Y, HIGH);
    delay(200);
}
}
```



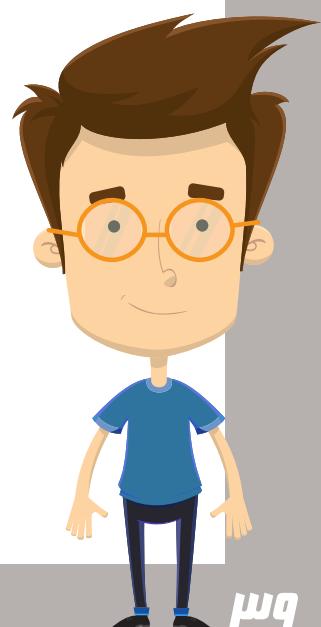
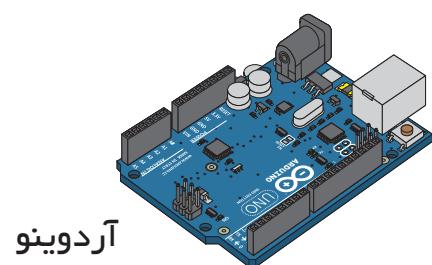
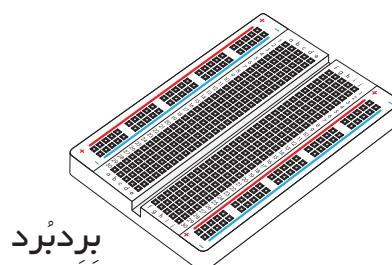
تنظیم سرعت چشمک زدن

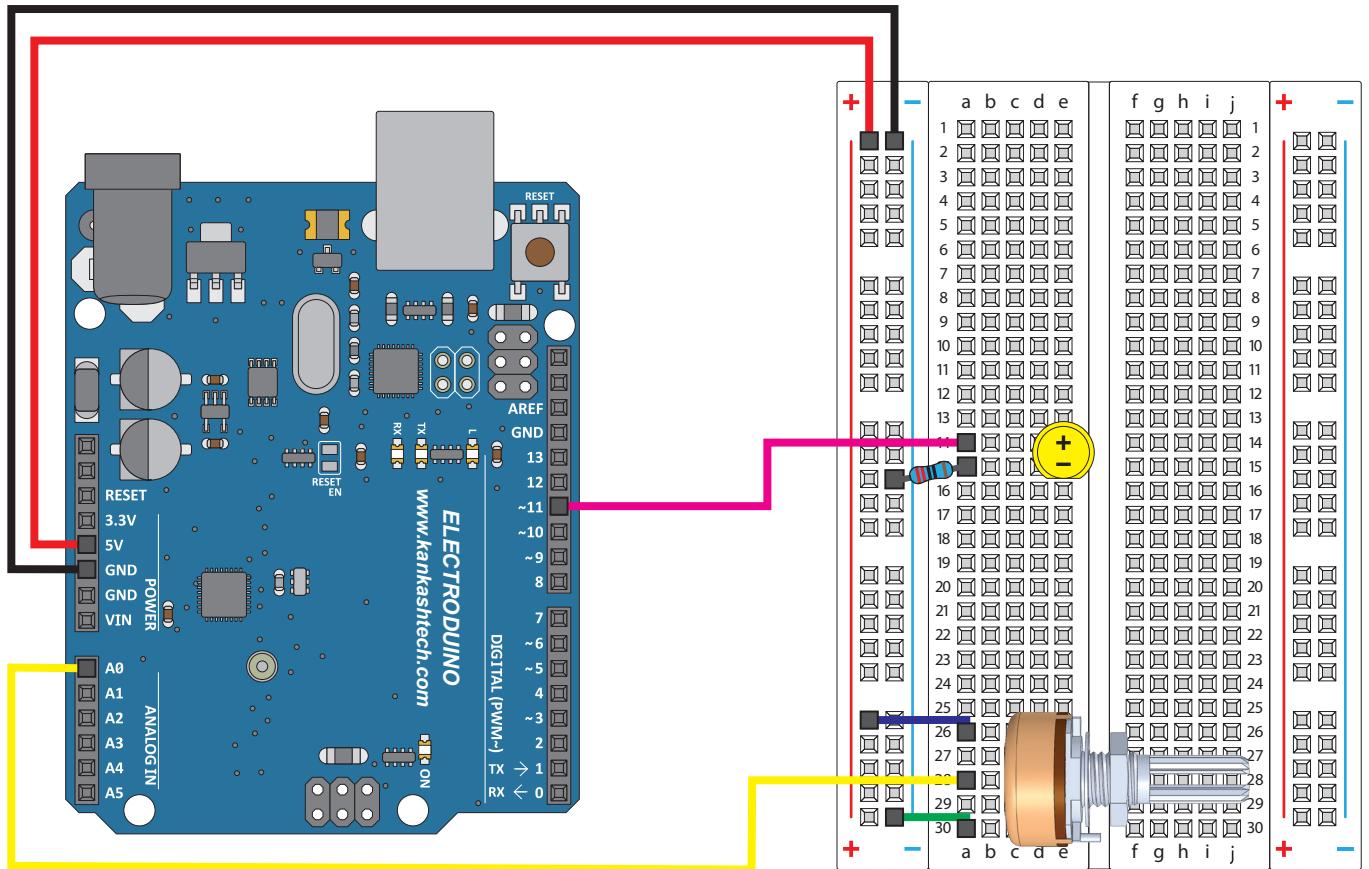
در این پروژه از پتانسیومتر استفاده خواهد شد. پتانسیومتر یک مقاومت متغیر است که سه پایه دارد. اگر به دو پایه کناری آن به ترتیب ۵ ولت و GND متصل شود، بسته به وضعیت ولوم پتانسیومتر، پایه وسط ولتاژی بین ۰ تا ۵ خواهد داشت.

در واقع پتانسیومتر یک مقاومت متغیر بین ۰ الی بی نهایت است که در وضعیت مقاومت ۰، پایه وسط ولتاژ ۵ و در وضعیت مقاومت بی نهایت، پایه وسط ولتاژ ۰ خواهد داشت.

در این پروژه با استفاده از پتانسیومتر، سرعت چشمک زدن یک LED تغییر داده می شود.

مطلوبات





توضیح مدار

مشابه پروژه های پیشین، یک LED به پین ۱۱ متصل می شود. مجددا از یک مقاومت ۲۵۰ اهم برای جلوگیری از آسیب دیدگی LED استفاده می شود.

پتانسیومتر ۳ پایه دارد. پایه وسط به پین A0 متصل می شود و جهت تولید سیگنال کنترلی برای تصمیم گیری آردوینو استفاده می شود.

یکی از پایه های کناری پتانسیومتر به پین ۵ ولت و پایه دیگر به پین GND متصل می شود.



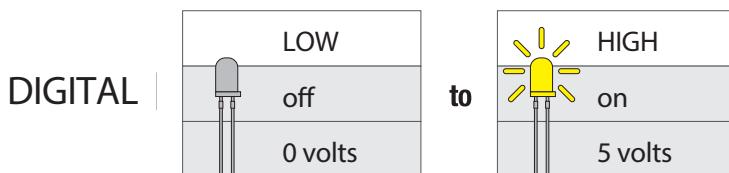
تنظیم سرعت چشمک زدن



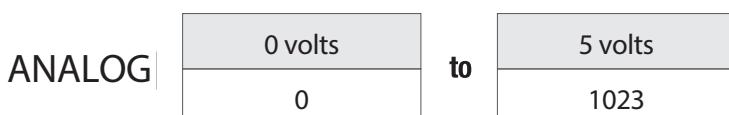
آنالوگ یا دیجیتال

اگر از نزدیک به آردوینو نگاه شود، دیده می شود برقی از پین ها با نام دیجیتال و برقی با آنالوگ مشخص شده اند.

بسیاری از قطعاتی که از آنها استفاده می شود مانند LED و دکمه های فشاری فقط دو وضعیت دارند: روشن که توسط آردوینو به صورت "HIGH" یا ۵ ولت شناخته می شود یا خاموش که توسط آردوینو "LOW" یا ۰ ولت شناخته می شود.



البته قطعات بسیاری وجود دارند که عملکرد آنها تنها خاموش و روشن نیست. دماسنجه، ولوم صدا و ... همه مقادیری بین HIGH و LOW دارند. برای استفاده از چنین قطعاتی، آردوینو ۶ پین ورودی آنالوگ دارد که ولتاژ ورودی را به مقداری میان ۰ (۰ ولت) و ۱۰۲۳ (۵ ولت) ترجمه می کند. پین های آنالوگ برای اندازه گیری مقادیر در دنیای واقعی استفاده می شوند.



پین های آنالوگ نیازی به تعریف به عنوان INPUT ندارند زیرا به صورت پیش فرض به عنوان ورودی استفاده می شوند. برای استفاده از این پین ها از دستور `analogRead` استفاده می شود با توجه به اینکه این دستور مقداری بین ۰ تا ۱۰۲۳ را بر می گرداند، باید مقدار این دستور در یک متغیر ذخیره شود:

```
int sensorValue=analogRead(0)
```

پین های آنالوگ با دو مشخصه شناخته می شوند: شماره پین و کanal پین. شماره پین های آنالوگ از A0 تا A5 است که متناظر با کانال های ۰ تا ۵ هستند.

دستور `analogRead` به صورت پیش فرض با استفاده از کانال پین کار می کند، ولی اگر از شماره پین استفاده شود، به صورت خودکار، شماره پین را به کانال تبدیل می کند. در واقع دستور زیر مشابه دستور بالا است:

```
int sensorValue=analogRead(A0)
```

پین های آنالوگ به روش های متعددی مورد استفاده قرار می کیرند که در پروژه بعد بیشتر به آن پرداخته خواهد شد



تغییر سرعت چشمک زدن

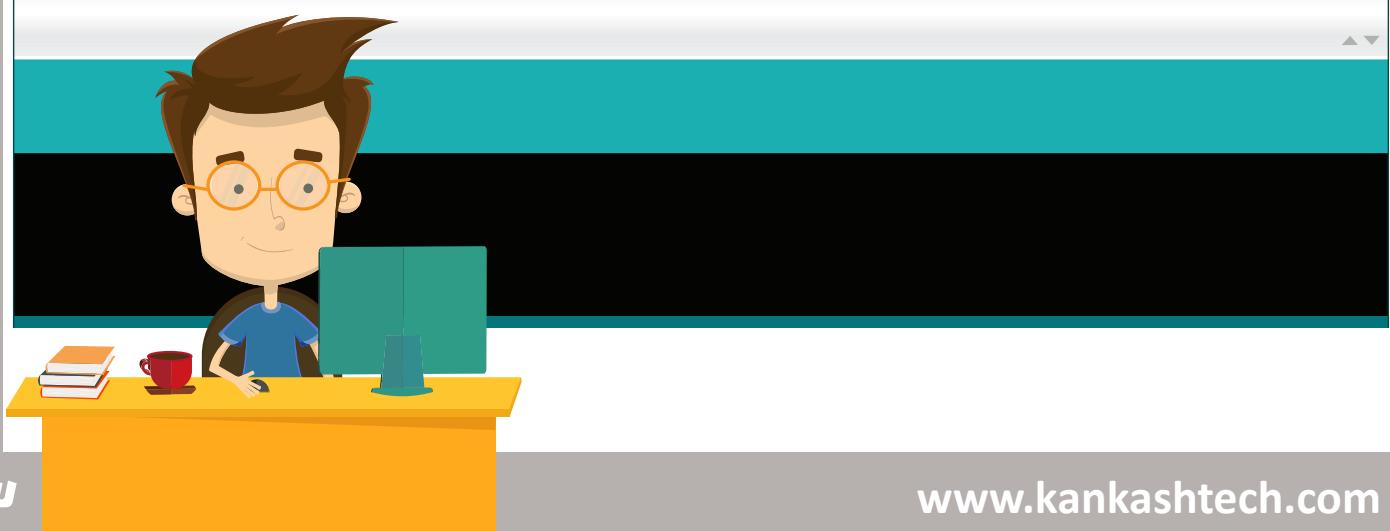


File > Examples > Electroduino > Circuit_4

```
int ledPin =11;
void setup() {
    pinMode(ledPin, OUTPUT);
}
void loop() {
    int sensorPin = A0;
    int sensorValue=analogRead(sensorPin);
    digitalWrite(ledPin, HIGH);
    delay(sensorValue);
    digitalWrite(ledPin,LOW);
    delay(sensorValue);
}
```

متغیر ledPin خارج از توابع و به صورت "global" تعریف شده تا در تمام توابع sketch قابل دسترسی باشد.

کد بالا را در IDE نوشه، کامپایل و به آردوینو آپلود کرده و عملکرد پتانسیومتر را بررسی کنید.



تنظیم شدت نور



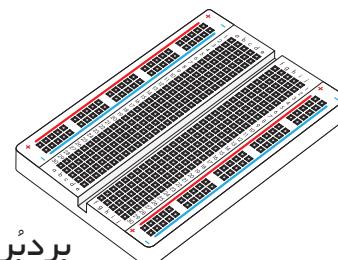
در پروژه قبل مشاهده شد که آردوینو می‌تواند ولتاژ آنالوگ (ولتاژ بین ۰ تا ۵ ولت) را با استفاده از تابع `analogRead` بخواند.

در این درس با استفاده از دستور `analogWrite` و اصل PWM، شدت نور یک LED کنترل خواهد شد.

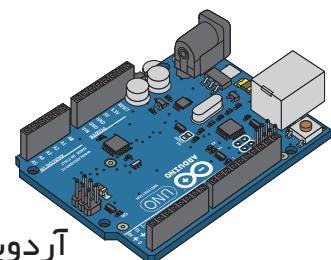
مطلوبات



سیم جامپر نری-نری
(۶ عدد)



بردبرد



آردوینو



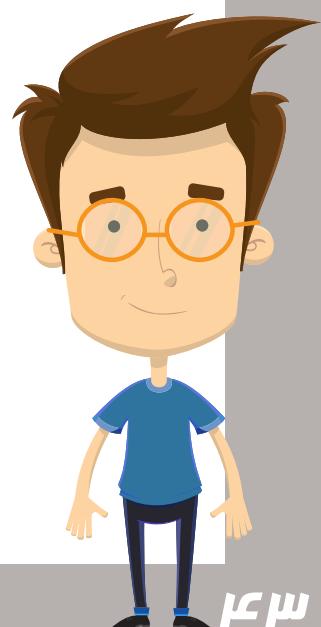
پتانسیومتر
(۱ عدد)

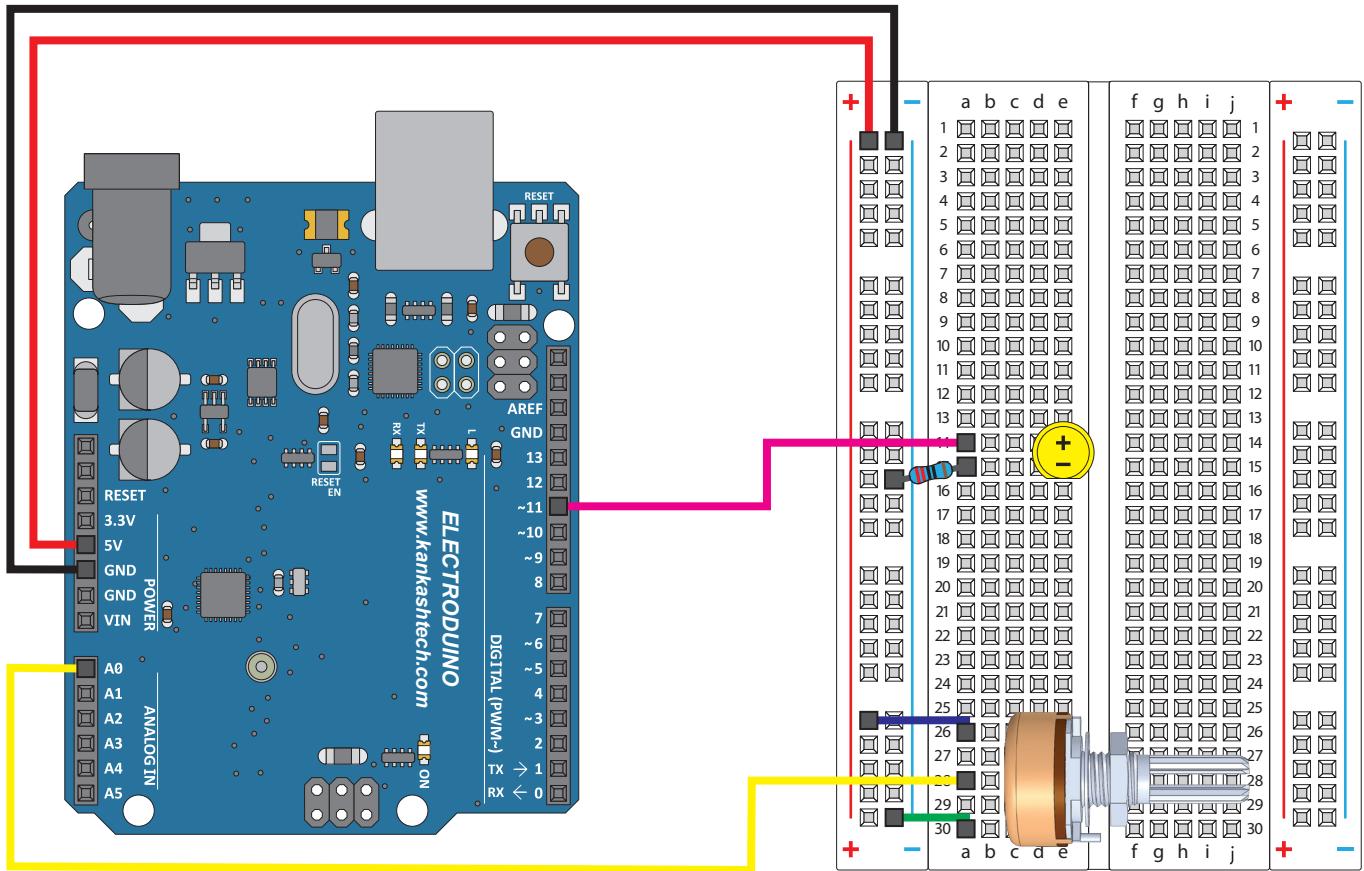


مقاومت ۲۲۰ اهم
(۱ عدد)



LED زرد
(۱ عدد)





توضیح مدار:

مدار این پروژه کاملاً شبیه پروژه ۴ است.



تنظیم شدت نور

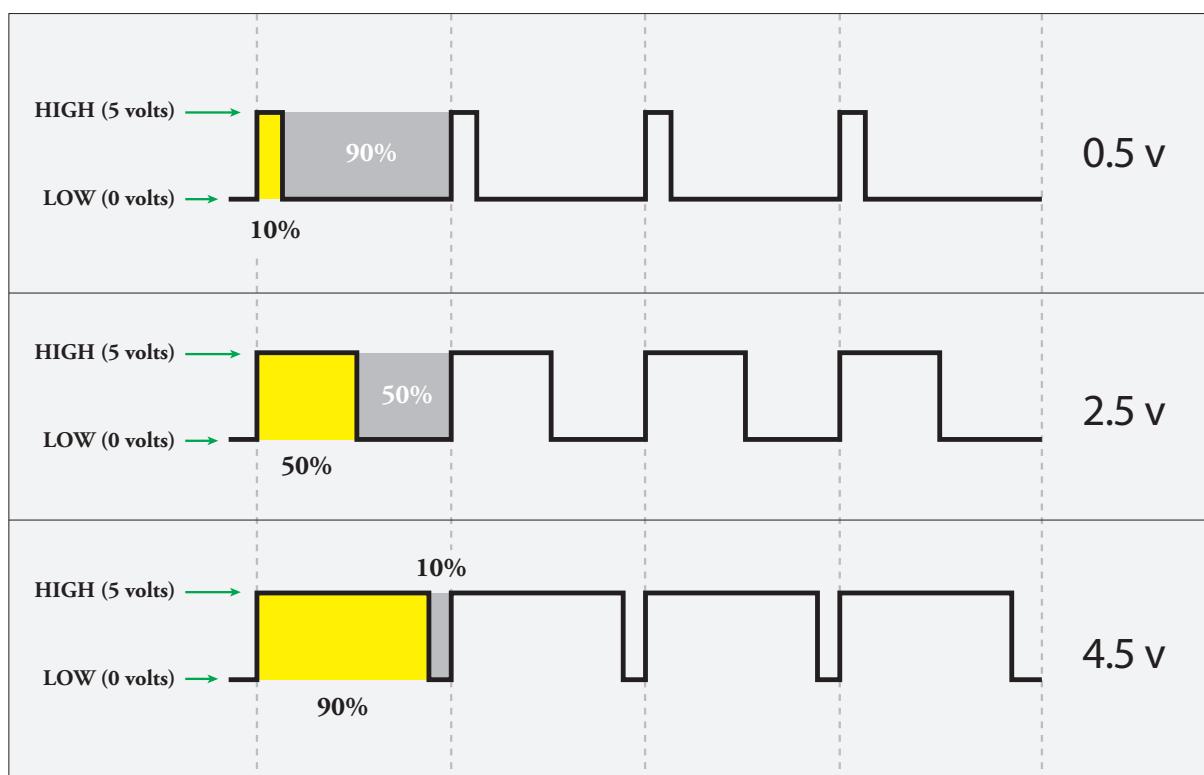


PWM

در پروژه قبل مشاهده شد که آردوینو می تواند با استفاده از دستور `analogRead`، ولتاژی بین ۰ تا ۵ را خوانده و از آن در تصمیم گیری و عملکرد پین ها استفاده کند.

سوالی که پیش می آید این است که آیا آردوینو می تواند خروجی ولتاژ آنالوگ داشته باشد؟ پاسخ هم خیر و هم بله است. آردوینو خروجی ولتاژ آنالوگ واقعی ندارد. ولی به علت عملکرد سریع آردوینو می توان با استفاده از PWM که مخفف Pulse Width Modulation این کار را انجام داد.

آردوینو به حدی سریع است که می تواند در یک ثانیه هزاران بار پین را خاموش و روشن کند. اگر این امکان را ایجاد می کند که آردوینو میزان زمان روشن و خاموش بودن یک پین را کنترل کند. اگر در بیشتر اوقات، پین مورد نظر HIGH باشد، LED متصل به پین درخشان دیده می شود. اگر در بیشتر زمان ها پین LOW باشد، LED متصل به پین کم نور دیده می شود. از آنجایی که این چشمک زدن در سرعتی بیشتر از قابلیت تشخیص چشم انسان صورت می گیرد، یک خروجی آنالوگ دیده می شود.



تنظیم شدت نور



AnalogWrite

برای استفاده از PWM از دستور **analogWrite** استفاده می‌شود. ابتدا پین مورد نظر باید به عنوان خروجی تعریف شود. برای استفاده از دستور **analogWrite**، باید پین مورد نظر و میزان ولتاژ را به صورت عددی بین ۰ (۰ ولت) و ۲۵۵ (۵ ولت) مشخص کرد. در دستور زیر پین ۱۱ به میزانی معادل ۲.۵ ولت روشن می‌شود:

```
pinMode (11, OUTPUT);  
analogWrite (11, 127);
```

دستور	ورودی	خروجی
digitalRead	شماره پین	HIGH/LOW
digitalWrite	شماره پین ۹	-
analogRead	شماره پین	۰ - ۱۰۲۳
analogWrite	شماره پین ۹	-

توضیحات تکمیلی - عملکرد پین ها

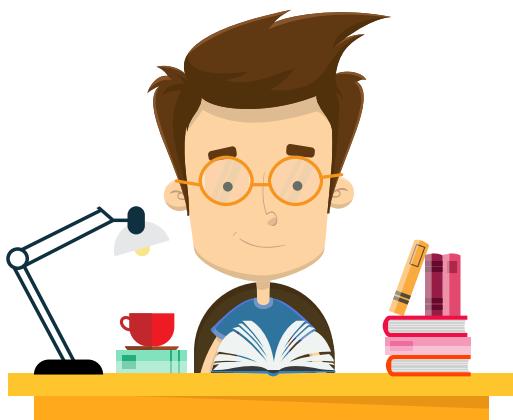
پین های آنالوگ ورودی می‌توانند هم به عنوان ورودی دیجیتال و هم به عنوان خروجی (آنالوگ و دیجیتال) استفاده شوند. نکته مهم این است که در غیر از حالت ورودی آنالوگ، این پین ها حتماً باید به صورت A0, A1, ... استفاده شوند. این قابلیت آردوینو این امکان را می‌دهد که اگر برای پروژه ای به پین هایی بیش از پین های دیجیتال نیاز بود، از پین های آنالوگ هم استفاده شود.

(**digitalRead**): بر روی همه پین ها کار می‌کند. این دستور، مقدار آنالوگ دریافتی را اگر بزرگتر یا مساوی ۱۲۷ بود به ۱ و اگر کوچکتر از ۱۲۷ بود به صفر گرد می‌کند.

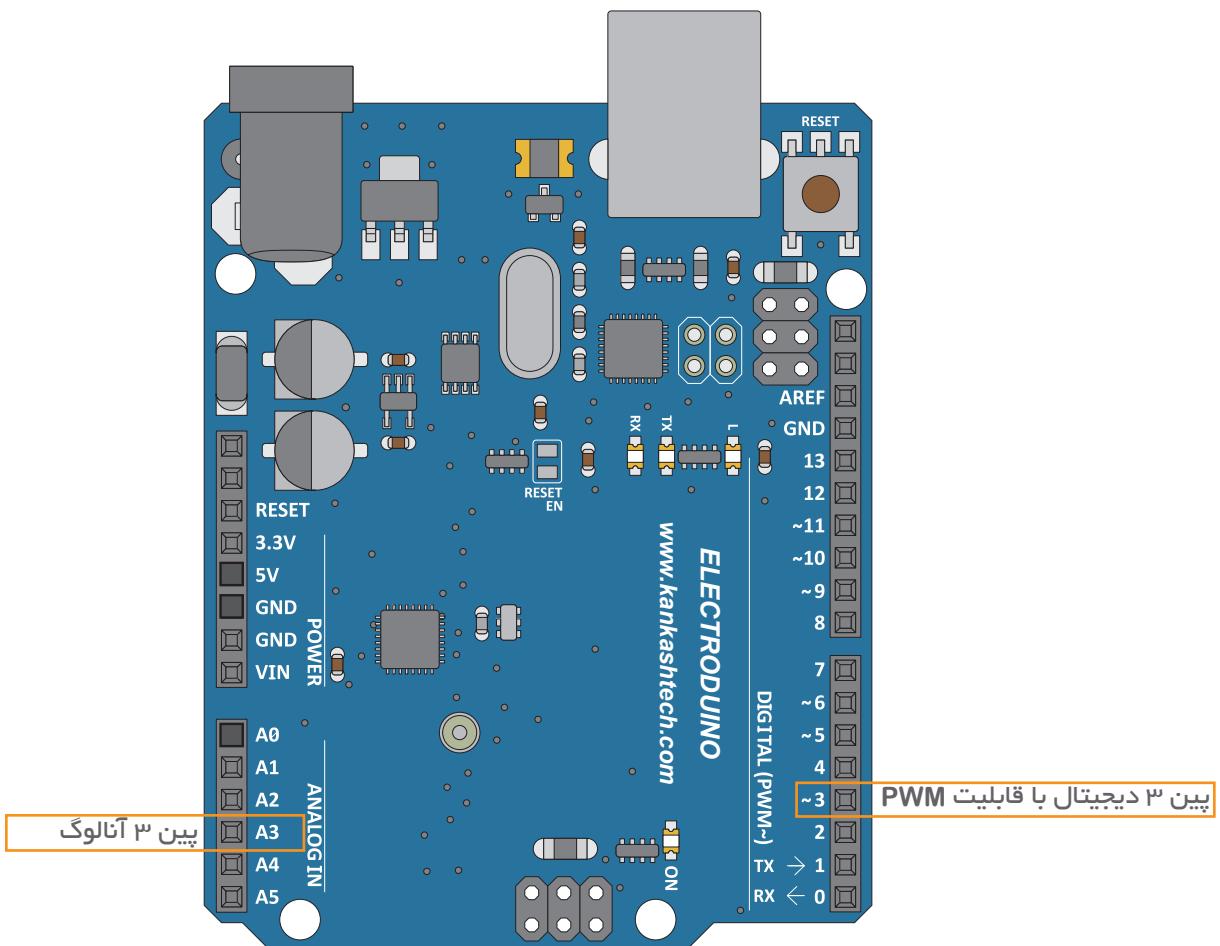
(**digitalWrite**): بر روی همه پین ها کار می‌کند و مقدار مجاز مورد استفاده در این دستور، اعداد ۰ و ۱ هستند. **digitalWrite(A0,0)** دقیقاً مانند **analogWrite(A0,0)** و **digitalWrite(A0,1)** دقیقاً مانند **analogWrite(A0,255)** عمل می‌کند.

(**analogRead()**): فقط بر روی پین های آنالوگ عمل کرده و می‌تواند مقداری بین ۰ تا ۱۰۲۳ بگیرد

(**analogWrite()**): بر روی همه پین های آنالوگ و پین های دیجیتال PWM کار می‌کند و می‌تواند مقادیری بین ۰ تا ۲۵۵ را به پین مورد نظر اعمال کند.



تنظیم شدت نور



دقت شود که دستور `analogRead` تنها بر روی پین های آنالوگ کار می کند.

دستور	عملکرد
<code>digitalRead (3);</code>	وضعیت پین ۳ دیجیتال را می خواند
<code>digitalRead (A3);</code>	وضعیت پین ۳ آنالوگ را می خواند
<code>digitalWrite (3,HIGH);</code>	پین ۳ دیجیتال را روشن می کند
<code>digitalWrite (A3, HIGH);</code>	پین ۳ آنالوگ را روشن می کند
<code>analogRead (3);</code>	مقدار پین ۳ آنالوگ را می خواند
<code>analogRead (A3);</code>	مقدار پین ۳ آنالوگ را می خواند
<code>analogWrite (3, 255);</code>	پین ۳ دیجیتال را روشن می کند
<code>analogWrite (A3,255);</code>	پین ۳ آنالوگ را روشن می کند



تنظیم شدت نور



دستور **map**

دستور **analogRead** مقداری میان ۰-۱۰۲۳ را متناظر با ولتاژ ۵-۰ خروجی می‌دهد. همانگونه که پیشتر ذکر شد، دستور **analogWrite** مقداری بین ۰-۲۵۵ می‌گیرد. برای تبدیل مقادیر خوانده شده در یک بازه مشخص به بازه جدید، از دستور **map** استفاده می‌شود. عملکرد این دستور بدین صورت است:

`x=map (y, a, b, c, d)`

در دستور بالا، `y` مقداری در بازه `a` تا `b` است. با استفاده از دستور بالا، مقدار `y` به تناسب فاصله از حدود بالا و پایین (`a` و `b`) به بازه جدید `c` تا `d` تصویر می‌شود.
به عنوان مثال، برای تبدیل عدد خوانده شده توسط پتانسیومتر (`sensorValue`) و تبدیل آن به مقدار مجاز دستور **analogWrite** (یعنی `(intensity)`):

`intensity=map (sensorValue, 0, 1023, 0, 255);`

لزومی به بزرگتر بودن `b` از `a` از `c` از `d` وجود ندارد و دستور **map** می‌تواند برای معکوس کردن یک بازه نیز به کار گرفته شود. به عنوان مثال اگر:

`intensity=map (sensorValue, 0, 1023, 255, 0);`

در این حالت، جهت عملکرد پتانسیومتر معکوس می‌شود.
دستور **map** مشکلی با مقادیر منفی نداشته و بسته به حدود بالا و پایین مشخص شده، هر بازه ای را می‌تواند به بازه دیگر تبدیل کند.

در موارد ساده که فقط نیاز به تبدیل حد بالا از ۰-۱۰۲۳ به ۰-۲۵۵ است، می‌توان عدد خوانده شده را بر ۴ تقسیم کرد.



File > Examples > Electroduino > Circuit_5

```
int ledPin=11;

void setup() {
    pinMode (ledPin,OUTPUT);
}
void loop() {
    int sensorPin = A0;
    int sensorValue=analogRead(sensorPin);
    int intensity = map (sensorValue, 0, 1023, 0, 255);
    //int intensity = sensorValue/4;
    analogWrite(ledPin, intensity);
    delay(1000);

}
```



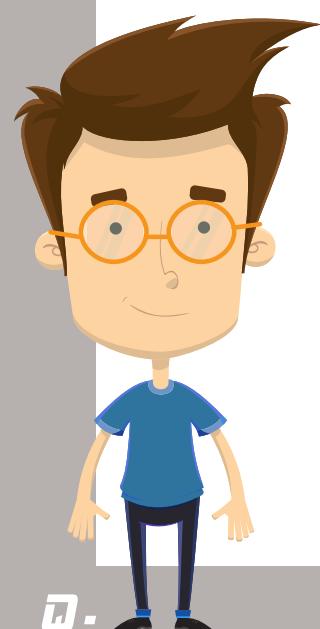
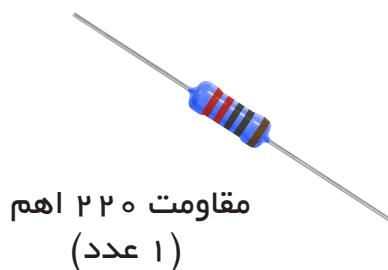
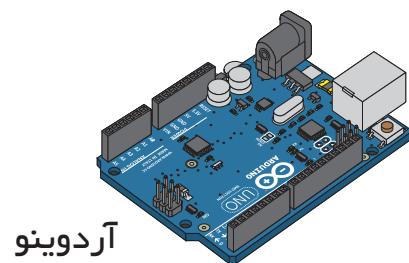
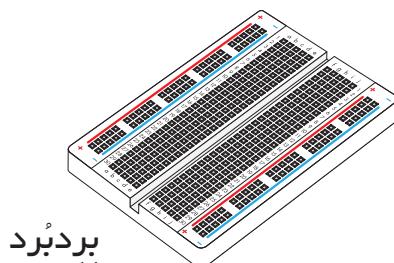
(RGB) سه رنگ LED

۷

پس از انجام چند پروژه مختلف با LED ساده و خاموش و روشن کردن و تغییر میزان درخشندگی آن، در این پروژه از LED رنگی یا RGB استفاده خواهد شد. LED های RGB، مخفف Red-Green-Blue به معنی قرمز - سبز - آبی هستند. در واقع این LEDها از سه قرمز و سبز و آبی تشکیل شده اند. با تنظیم خاموش و روشن شدن این سه LED و نیز تعیین میزان درخشندگی هر کدام، می توان اکثر رنگ ها را ایجاد کرد.

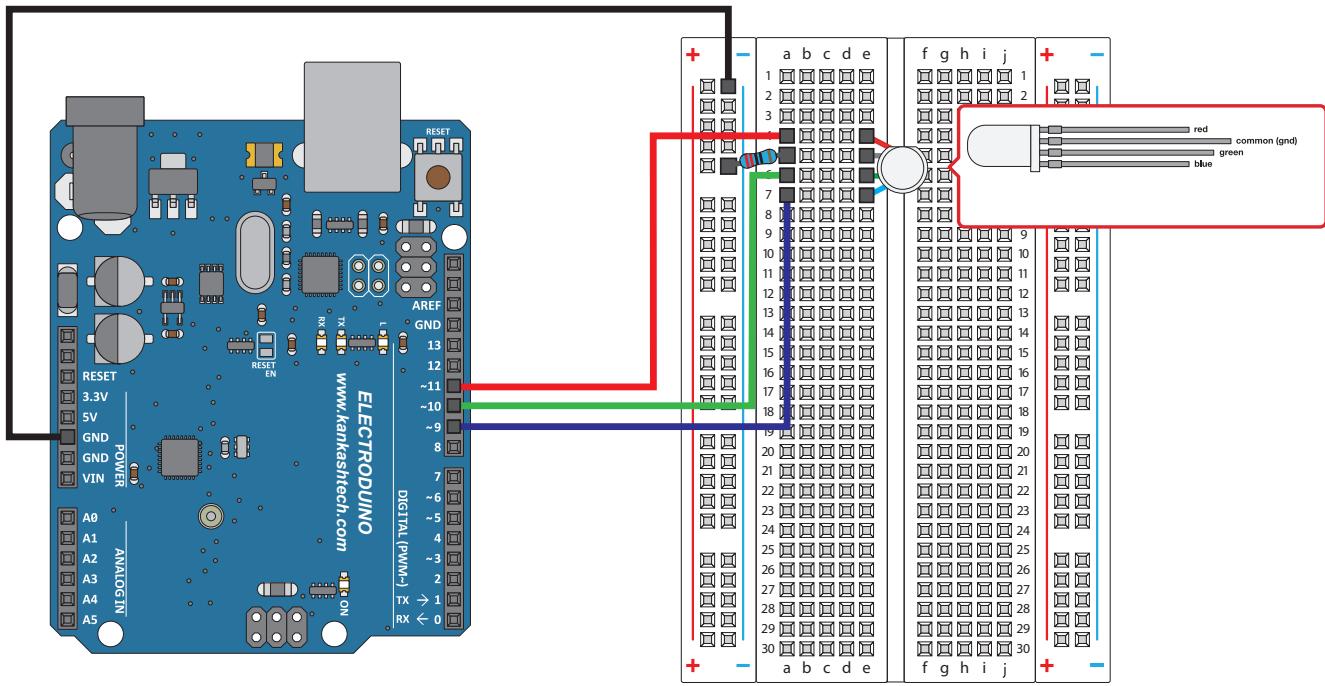
سه LED رنگ در دو نوع آند مشترک و کاتد مشترک هستند. کلیات استفاده مشابه است، با این تفاوت که در نوع کاتد مشترک، پایه مشترک به GND وصل شده و برای روشن شدن هر یک رنگ باید پایه آن رنگ HIGH شود؛ در صورتی که در نوع آند مشترک، پایه مشترک به ۵V متصل شده و برای روشن شدن رنگ ها، پایه مربوط به آن رنگ باید LOW شود.

مطلوبات



LED رنگ سه مشترک (RGB)-کاتد (Common Anode)

۷



توضیح مدار:

LED های RGB چهار پایه دارند که پایه بلند پایه منفی مشترک و هر کدام از سه پایه دیگر مطابق شکل مربوط به یک رنگ هستند. برای جلوگیری از آسیب دیدن LED از یک مقاومت میان GND آردوینو و پایه منفی استفاده می شود. برای تنظیم رنگ به تنظیم میزان درخشندگی هر پایه نیاز است؛ در نتیجه از سه پین دارای قابلیت PWM یعنی ۱۱ و ۱۰ و ۹ و ۸ و ۷ و ۶ و ۵ و ۴ و ۳ و ۲ و ۱ و ۰ به ترتیب برای پایه های قرمز، سبز و آبی استفاده می شود.

در برخی از نمونه های این قطعه، پایه های سبز و آبی برعکس این مدار است.



LED سه رنگ (RGB)-کاتد مشترک

۱

توابع

پیش از این با دو تابع داخلی آردوینو یعنی `setup` و `loop` آشنا شدید. در این پروژه برای کدنویسی بهتر و مرتب تر، حالت های مختلف عملکردی LED را در قالب توابعی نوشته و در هنگام اجرای تابع `loop` فراخوانی می شود. ابتدا و پیش از تابع `setup`، پین های مربوط به رنگ ها به صورت `global` تعریف می شوند تا توسط تمام توابع قابل دسترسی و اعمال تغییرات باشند.

```
int bluePin=9;  
int greenPin=10;  
int redPin=11;
```

در تعریف توابع زیر از کلمه `void` استفاده شده و درون پرانتزهای مقابل نام تابع چیزی نوشته نمی شود زیرا در هنگام فراخوانی تابع و پس از اجرای آن، متغیر و مقداری بین توابع مبادله نمی شود.

تابع قرمز

```
void RED () {  
    digitalWrite(redPin, HIGH);  
    digitalWrite(greenPin, LOW);  
    digitalWrite(bluePin, LOW);  
}
```

تابع سبز

```
void GREEN () {  
    digitalWrite(redPin, LOW);  
    digitalWrite(greenPin, HIGH);  
    digitalWrite(bluePin, LOW);  
}
```

تابع آبی

```
void BLUE () {  
    digitalWrite(redPin, LOW);  
    digitalWrite(greenPin, LOW);  
    digitalWrite(bluePin, HIGH);  
}
```

تابع طیف

برای ایجاد طیف رنگ از حلقه `for` استفاده شده است. در این حلقه، درخشندگی هر رنگ به تدریج تغییر داده شود. بدین منظور ابتدا سه متغیر شدت درخشندگی برای هر پایه تعریف می شود.

در این پروژه ابتدا رنگ آبی روشن شده و به تدریج به حداقل درخشندگی می رسد. سپس در وضعیتی که آبی کاملاً روشن است، رنگ سبز و پس از آن رنگ قرمز به تدریج روشن می شود.

پس از اینکه هر سه رنگ روشن شدند، ابتدا رنگ سبز و سپس آبی به تدریج خاموش می شود. این ترکیب، طیفی رنگی ایجاد می کند.



ریزگار LED-تاد مشرک (RGB) (LED شرکت)

File > Examples > Electroduino > Circuit_6

```
int bluePin=9;
int greenPin=10;
int redPin=11;
void setup() {
    pinMode(bluePin,OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(redPin,OUTPUT);
}
void loop() {
    GREEN();
    delay(1000);
    BLUE();
    delay(1000);
    RED();
    delay(1000);
    SPECTRUM();
    delay(1000);
}
void RED () {
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
}
void GREEN() {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, LOW);
}
```



لەگ، ۋە LED - (RGB) چاتىد مشتىرى

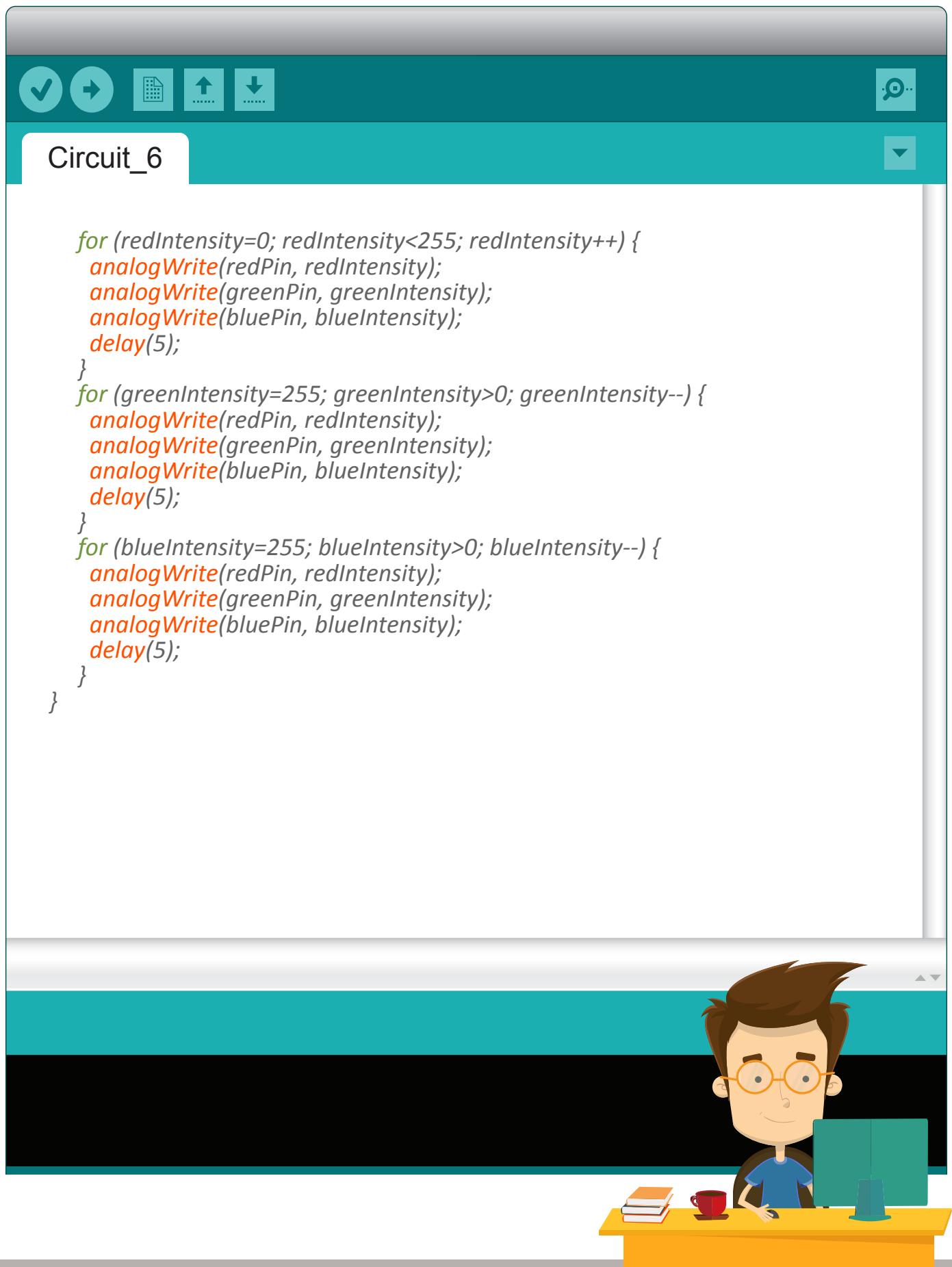
File > Examples > Electroduino > Circuit_6

```
void BLUE() {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, HIGH);
}
void SPECTRUM() {
    int redIntensity;
    int greenIntensity;
    int blueIntensity;
    for (blueIntensity=0; blueIntensity<255; blueIntensity++) {
        analogWrite(redPin, redIntensity);
        analogWrite(greenPin, greenIntensity);
        analogWrite(bluePin, blueIntensity);
        delay(5);
    }
    for (greenIntensity=0; greenIntensity<255; greenIntensity++) {
        analogWrite(redPin, redIntensity);
        analogWrite(greenPin, greenIntensity);
        analogWrite(bluePin, blueIntensity);
        delay(5);
    }
}
```



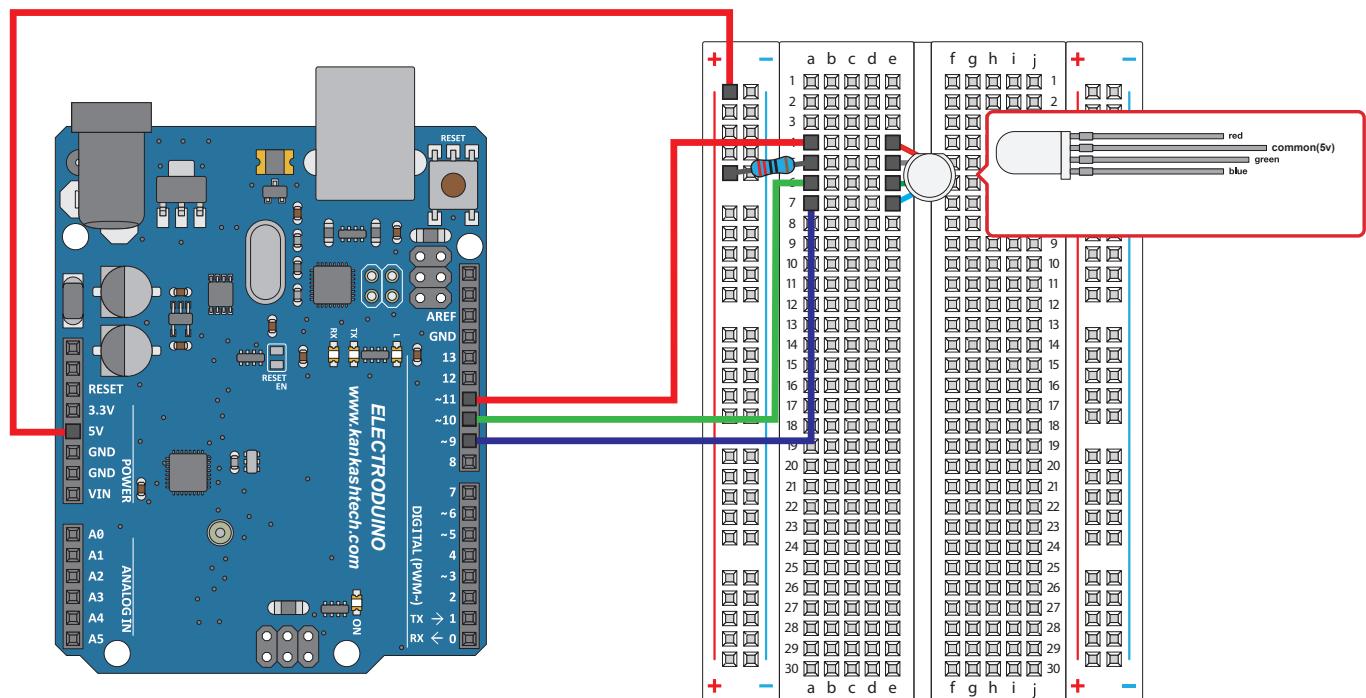
LED رنگی (RGB)-تاتد مشترک

File > Examples > Electroduino > Circuit_6



```
for (redIntensity=0; redIntensity<255; redIntensity++) {
    analogWrite(redPin, redIntensity);
    analogWrite(greenPin, greenIntensity);
    analogWrite(bluePin, blueIntensity);
    delay(5);
}
for (greenIntensity=255; greenIntensity>0; greenIntensity--) {
    analogWrite(redPin, redIntensity);
    analogWrite(greenPin, greenIntensity);
    analogWrite(bluePin, blueIntensity);
    delay(5);
}
for (blueIntensity=255; blueIntensity>0; blueIntensity--) {
    analogWrite(redPin, redIntensity);
    analogWrite(greenPin, greenIntensity);
    analogWrite(bluePin, blueIntensity);
    delay(5);
}
```

۷ آند مشترک LED رنگ RGB- آند مشترک



توضیح مدار:

LED های آند مشترک، عملکردی مشابه گونه کاتد مشترک دارند، با این تفاوت که به جای اینکه پایه مشترک به GND متصل شود، در نوع آند مشترک پایه مشترک (پایه بلند) به 5v متصل می شود.

در برخی از نمونه های این قطعه، جای پایه های سبز و آبی برعکس این مدار است.



سه رنگ (RGB)-آند مشترک



تابع

ساختار کلی برنامه LED های آند مشترک، با LED های کاتد مشترک شباهت دارد. همانگونه که در قسمت قبل دیده شد، در گونه کاتد مشترک، پایه مشترک هر سه رنگ به منفی (GND) متصل می شود. در نتیجه برای روشن کردن هر رنگ، باید پایه دیگر مربوط به آن رنگ HIGH شود. از آنجایی که در گونه آند مشترک، پایه مثبت هر کدام از رنگ ها به ۵ ولت متصل شده است، برای روشن شدن هر کدام از رنگ ها، باید پایه مربوط به آن رنگ در وضعیت LOW قرار داده شود. در نتیجه توابع رنگ به صورت زیر اصلاح می شود:

تابع قرمز

```
void RED () {  
    digitalWrite(redPin, LOW);  
    digitalWrite(greenPin, HIGH);  
    digitalWrite(bluePin, HIGH);  
}
```

تابع سبز

```
void GREEN () {  
    digitalWrite(redPin, HIGH);  
    digitalWrite(greenPin, LOW);  
    digitalWrite(bluePin, HIGH);  
}
```

تابع آبی

```
void BLUE () {  
    digitalWrite(redPin, HIGH);  
    digitalWrite(greenPin, HIGH);  
    digitalWrite(bluePin, LOW);  
}
```

تابع طیف

برای ایجاد طیف رنگی با LED سه رنگ آند مشترک، تابع طیف نیز نسبت به وضعیت کاتد مشترک تغییر می کند. از آنجایی که پایه مشترک به ۵ ولت متصل است، هر چقدر پایه های مربوط به رنگ ها به ۰ نزدیک تر باشد، اختلاف ولتاژ دو سر LED بیشتر بوده و درخشندگی آن بیشتر می شود. در نتیجه اگر مقدار آنالوگ یک پایه ۲۵۵ تنظیم شود، آن پایه خاموش و اگر ۰ تنظیم شود، آن پایه در وضعیت حداکثر درخشندگی قرار می گیرد.



LED رنگی (RGB)-آند مشترک

File > Examples > Electroduino > Circuit_6_2

```
int bluePin=9;
int greenPin=10;
int redPin=11;
void setup() {
    pinMode(bluePin,OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(redPin,OUTPUT);
}
void loop() {
    GREEN();
    delay(1000);
    BLUE();
    delay(1000);
    RED();
    delay(1000);
    SPECTRUM();
    delay(1000);
}
void RED () {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, HIGH);
}
```



LED رنگی (RGB) میله مشترک

File > Examples > Electroduino > Circuit_6_2

```
Circuit_6_2

void GREEN () {
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, HIGH);
}

void BLUE () {
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, LOW);
}

void SPECTRUM() {
    int redIntensity;
    int greenIntensity;
    int blueIntensity;

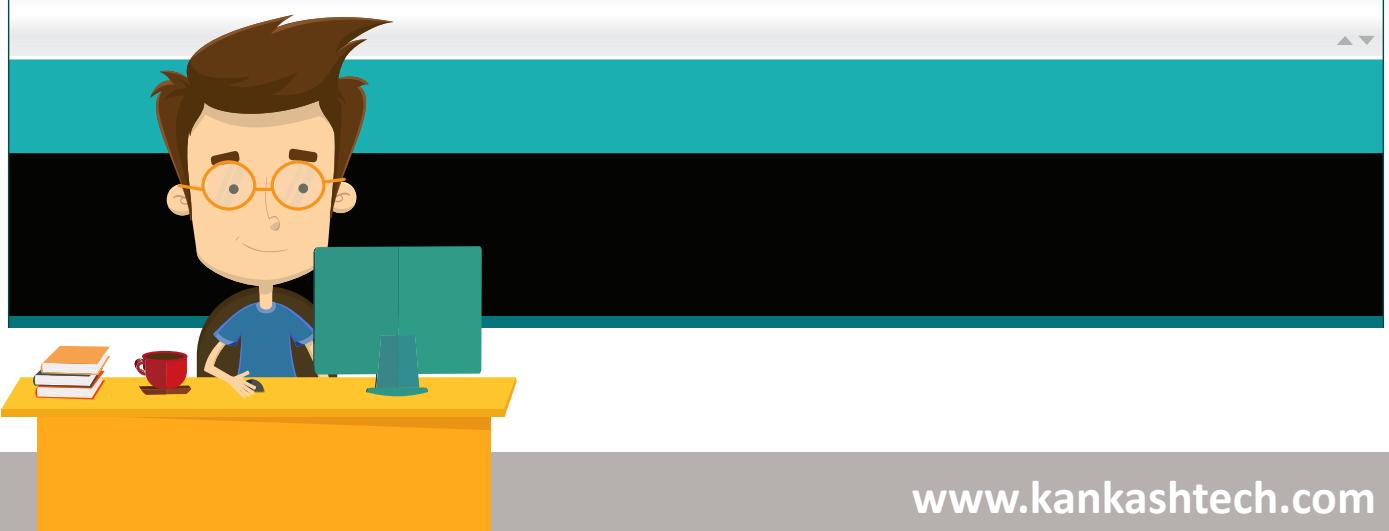
    for (blueIntensity=255; blueIntensity>0; blueIntensity--) {
        analogWrite (redPin, redIntensity);
        analogWrite (greenPin, greenIntensity);
        analogWrite (bluePin, blueIntensity);
        delay (10);
    }
    for (greenIntensity=255; greenIntensity>0; greenIntensity--) {
        analogWrite (redPin, redIntensity);
        analogWrite (greenPin, greenIntensity);
        analogWrite (bluePin, blueIntensity);
        delay (10);
    }
}
```



LED رنگی (RGB)-آند مشترک

File > Examples > Electoduino > Circuit_6_2

```
for (redIntensity=255; redIntensity>0; redIntensity--) {
    analogWrite (redPin, redIntensity);
    analogWrite (greenPin, greenIntensity);
    analogWrite (bluePin, blueIntensity);
    delay (10);
}
for (greenIntensity=0; greenIntensity<255; greenIntensity++) {
    analogWrite (redPin, redIntensity);
    analogWrite (greenPin, greenIntensity);
    analogWrite (bluePin, blueIntensity);
    delay (10);
}
for (blueIntensity=0; blueIntensity<255; blueIntensity++) {
    analogWrite (redPin, redIntensity);
    analogWrite (greenPin, greenIntensity);
    analogWrite (bluePin, blueIntensity);
    delay (10);
}
```

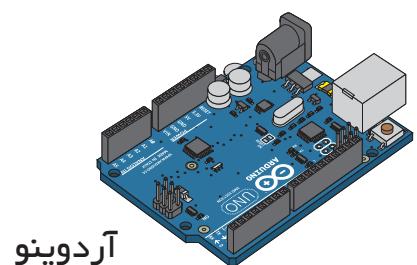
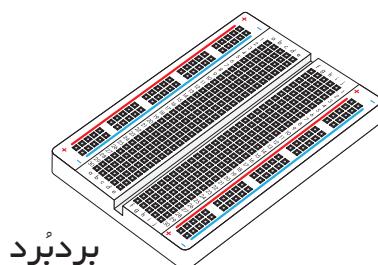


ماژول LED سه رنگ (RGB)

ماژول‌ها مدارهایی آماده هستند که جهت سهولت به کارگیری قطعات الکترونیکی استفاده می‌شوند. هر قطعه الکترونیکی بسته به نوع و نحوه به کارگیری، جهت عملکرد صحیح به مداری شامل قطعات پایه نظیر مقاومت خازن، دیود و ... نیاز دارد که توسط سازنده در قالب دیتابشیت ارائه می‌شود. ماژول‌ها شکل ساده شده این مدارها هستند که قطعات مورد نیاز عملکرد صحیح قطعه را به صورت مجتمع دارند.

در این پروژه از ماژول LED سه رنگ RGB با کدی مشابه کد پروژه ۶ استفاده می‌شود.

مطلوبات

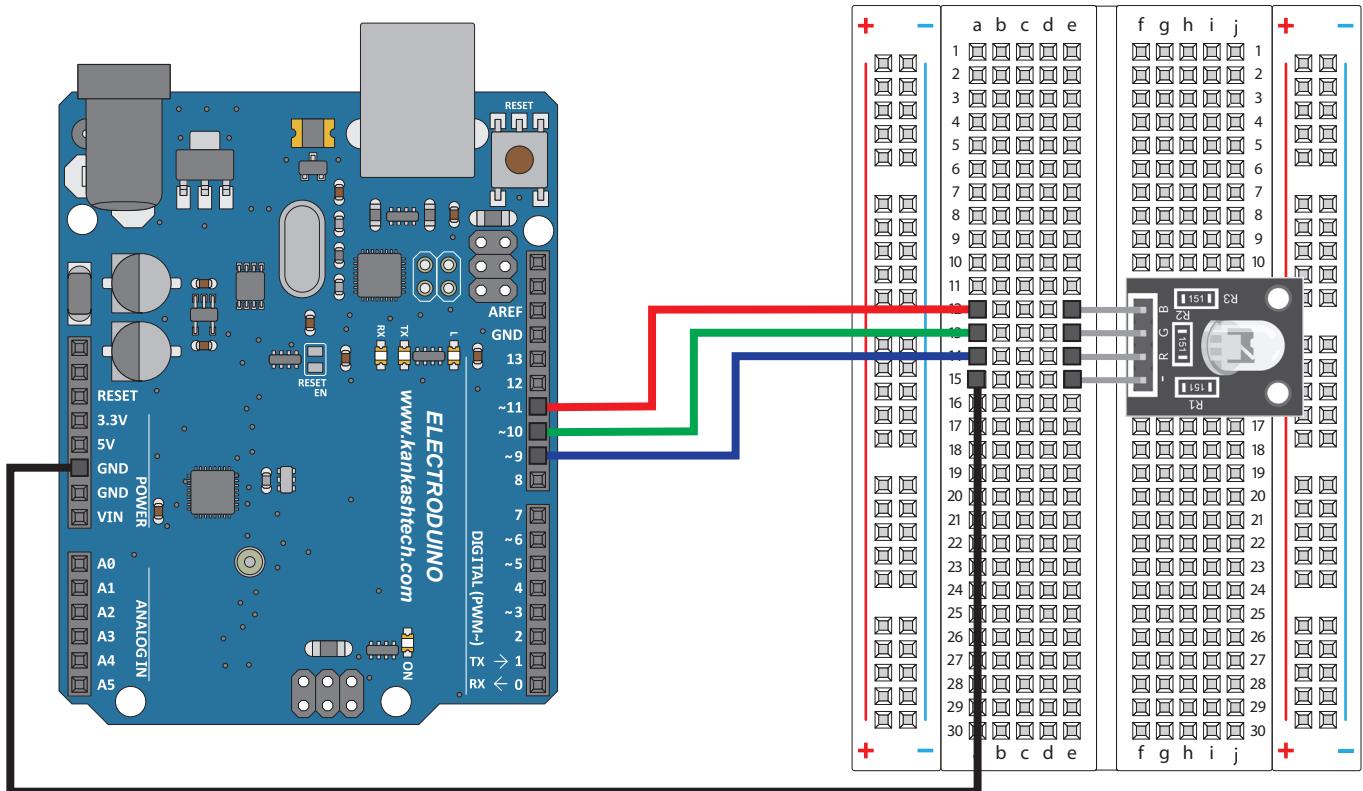


ماژول LED-RGB
(۱ عدد)



ماژول LED سه رنگ (RGB)

U



توضیح مدار:

ماژول LED سه رنگ همانند خود LED سه رنگ، چهار پایه دارد. در این ماژول مقاومت های مورد نیاز برای جلوگیری از آسیب دیدگی LED، در خود بُرد ماژول قرار گرفته و نیازی به استفاده از مقاومت مجزا نیست. کافی است پین های مربوط را به آردوینو وصل کرد.

اگر از نزدیک به ماژول نگاه شود، دیده می شود که پایه های ماژول، با استفاده از حروف و علائم، از هم قابل تشخیص است. با توجه به اینکه ماژول بُرد مختص خود را دارد، چیدمان پایه ها مطابق با پایه های خود LED نیست و برای بستن مدار، حتما باید به علائم نوشته شده در کنار پایه ها توجه کرد.

کد این پروژه کاملا مشابه بخش اول پروژه ۶ است.



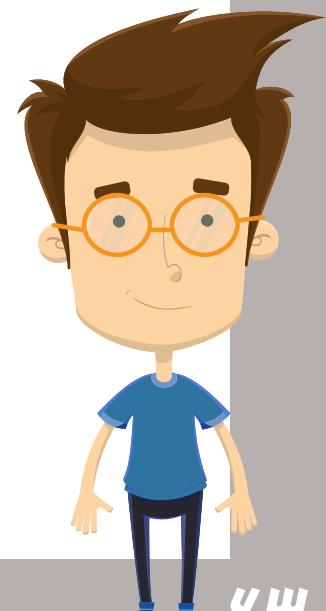
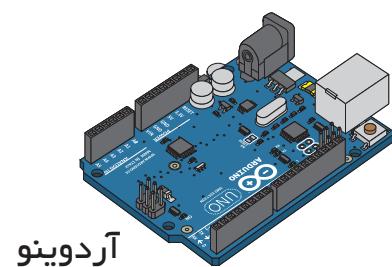
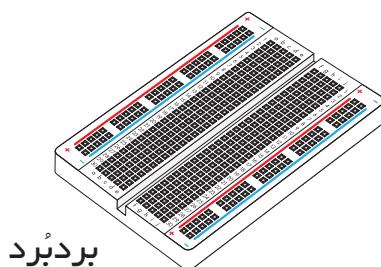
سون سگمنت (7-SEGMENT)

۱

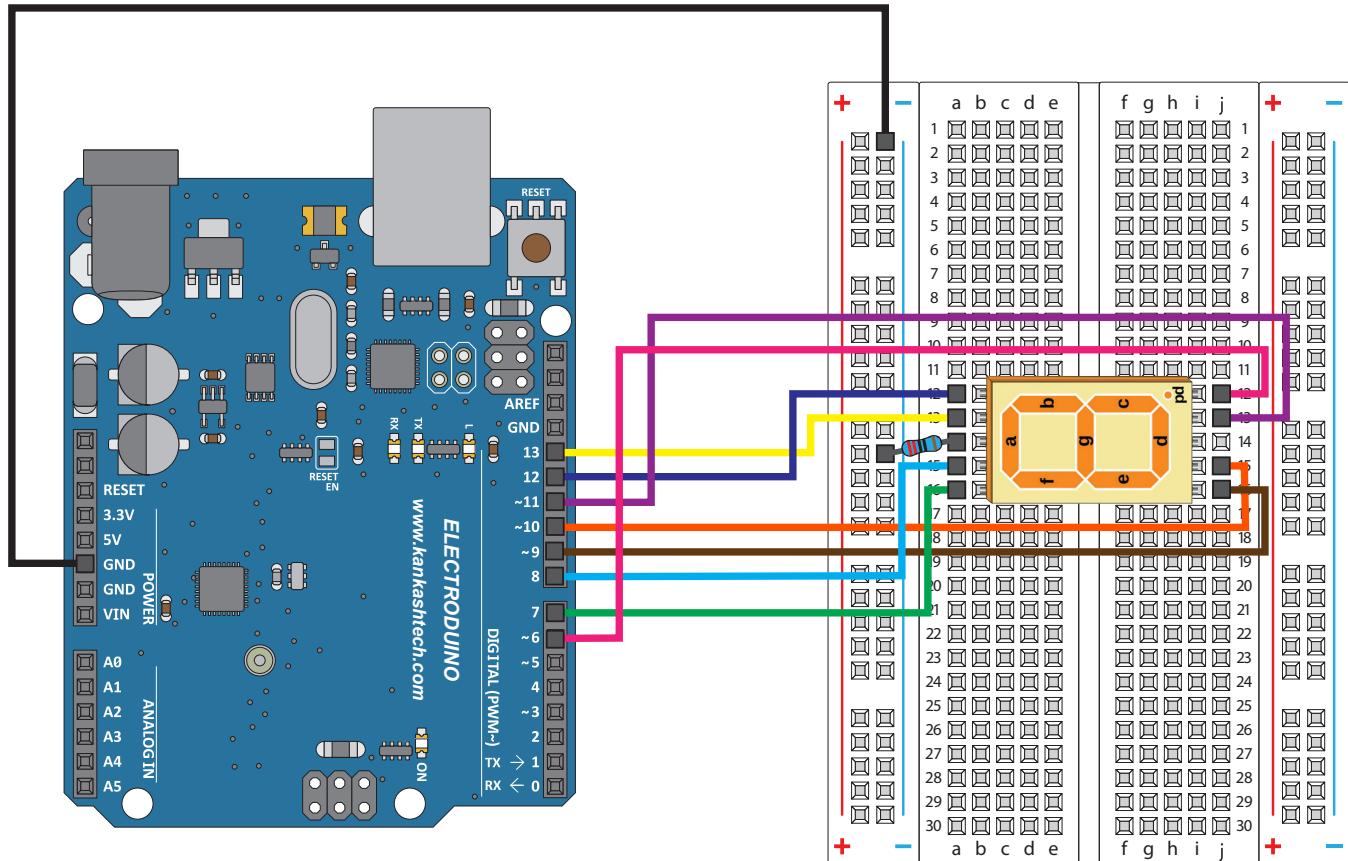
سون سگمنت یک قطعه الکترونیکی مت Shankل از 7 عدد LED است که می تواند برای نمایش اعداد و بخی از حروف استفاده می شود. هر کدام از این 7 LED یک سگمنت تامیله می شود. در برخی از سون سگمنت ها (مانند قطعه موجود در این مجموعه) یک LED هشتم برای نشان دادن نقطه اعشار نیز وجود دارد.

با LOW/HIGH کردن پایه های مختلف می توان اعداد/حروف مختلفی را نمایش داد.
در این پروژه نمایش اعداد و حروف بر روی سون سگمنت انجام می شود.

مطلوبات



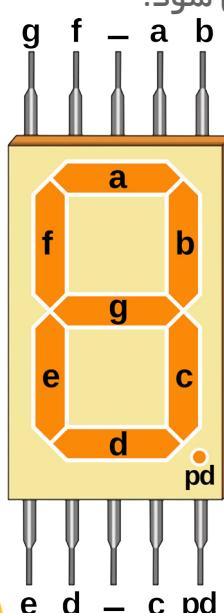
سون سگمنت-کاتد مشترک



توضیح مدار:

مشابه LED های سه رنگ (RGB) سون سگمنت ها نیز انواع کاتد مشترک و آند مشترک دارند. در نوع کاتد مشترک، کاتدهای سگمنت ها (پایه منفی LED) به هم متصل هستند و با اعمال ولتاژ مثبت به سایر پایه ها، سگمنت مورد نظر روشن می شود. هر کدام از سگمنت ها توسط یکی از حروف g-a-d-p به پین های ۶ الی ۱۳ وصل می شود. یکی از پین های منفی به GND آردوینو متصل می شود.

چیدمان پایه های سون سگمنت کاتد مشترک مانند شکل رو برو است. برای جلوگیری از آسیب دیدگی سگمنت ها باید از مقاومت در مسیر استفاده شود. برای سادگی مدار، تنها یک مقاومت به پایه مشترک سون سگمنت متصل می شود. می توان به جای این کار از ۸ مقاومت برای ۸ پایه سگمنت ها استفاده کرد.



سون سگمنت - کاتد مشترک



در سون سگمنت های کاتد مشترک، کاتدها (پایه های منفی) به هم و به GND متصل می شوند. برای روشن کردن هر سگمنت، باید پایه مربوطه HIGH شود تا جریان برقرار شود.

برای نمایش اعداد و حروف، ابتدا پین های متناظر با سگمنت ها تعریف می شود. با توجه به مدار پروژه:

```
int SEG_a=13;  
int SEG_b=12;  
int SEG_c=11;  
int SEG_d=10;  
int SEG_e=9;  
int SEG_f=8;  
int SEG_g=7;  
int SEG_pd=6;
```

سپس وضعیت سگمنت ها در هر کدام از اعداد و حروف در قالب تابع هایی نوشته می شود. به عنوان

مثال برای نمایش عدد ۰ انگلیسی با نقطه اعشار خاموش بر روی سون سگمنت کاتد مشترک:

```
void ZERO () {  
    digitalWrite(SEG_a, HIGH);  
    digitalWrite(SEG_b, HIGH);  
    digitalWrite(SEG_c, HIGH);  
    digitalWrite(SEG_d, HIGH);  
    digitalWrite(SEG_e, HIGH);  
    digitalWrite(SEG_f, HIGH);  
    digitalWrite(SEG_g, LOW);  
    digitalWrite(SEG_pd, LOW);  
}
```

به جز اعداد ۰ تا ۹، حروف A, b, c, C, d, E, F, h, H, J, L, n, o, P, q, u, U عنوان مثال برای تعریف حرف d:

```
void dLetter () {  
    digitalWrite(SEG_a, LOW);  
    digitalWrite(SEG_b, HIGH);  
    digitalWrite(SEG_c, HIGH);  
    digitalWrite(SEG_d, HIGH);  
    digitalWrite(SEG_e, HIGH);  
    digitalWrite(SEG_f, LOW);  
    digitalWrite(SEG_g, HIGH);  
    digitalWrite(SEG_pd, LOW);  
}
```

ساختمان اعداد و حروف به همین صورت تعریف می شوند

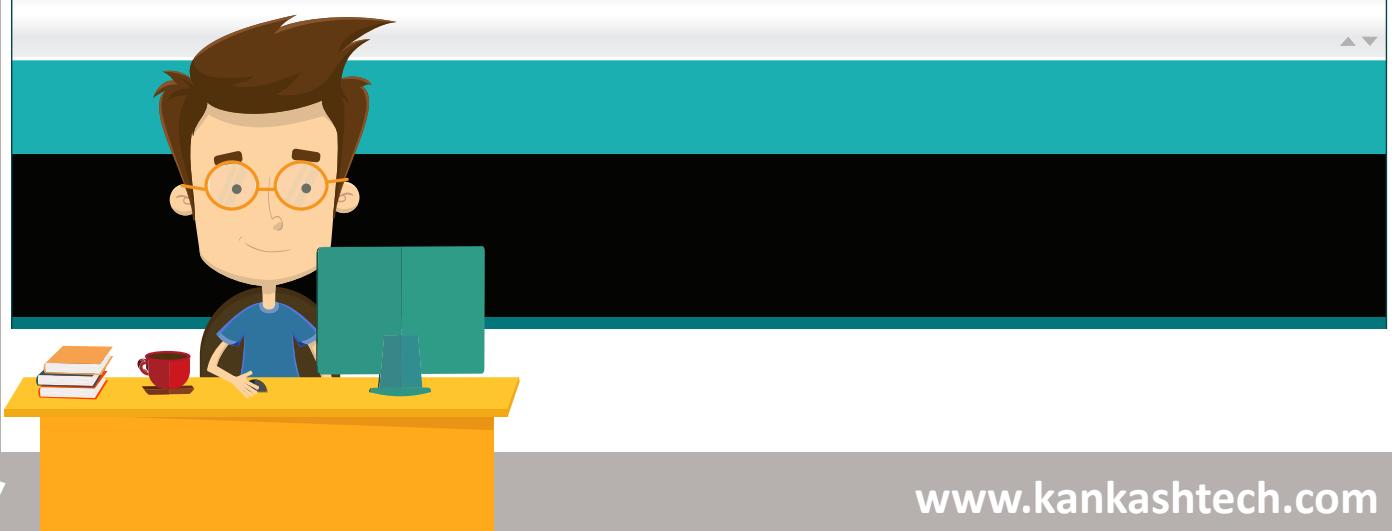




File > Examples > Electroduino > Circuit_8

```
int SEG_a=13;
int SEG_b=12;
int SEG_c=11;
int SEG_d=10;
int SEG_e=9;
int SEG_f=8;
int SEG_g=7;
int SEG_pd=6;

void setup() {
    pinMode (SEG_pd,OUTPUT);
    pinMode (SEG_c,OUTPUT);
    pinMode (SEG_d,OUTPUT);
    pinMode (SEG_e,OUTPUT);
    pinMode (SEG_f,OUTPUT);
    pinMode (SEG_g,OUTPUT);
    pinMode (SEG_a,OUTPUT);
    pinMode (SEG_b,OUTPUT);
}
```



File > Examples > Electroduino > Circuit_8

```
void loop() {
    ZERO();
    delay(1000);
    ONE();
    delay(1000);
    TWO();
    delay(1000);
    THREE();
    delay(1000);
    FOUR();
    delay(1000);
    FIVE();
    delay(1000);
    SIX();
    delay(1000);
    SEVEN();
    delay(1000);
    EIGHT();
    delay(1000);
    NINE();
    delay(1000);
    aLetter();
    delay(1000);
    bLetter();
    delay(1000);
}
```



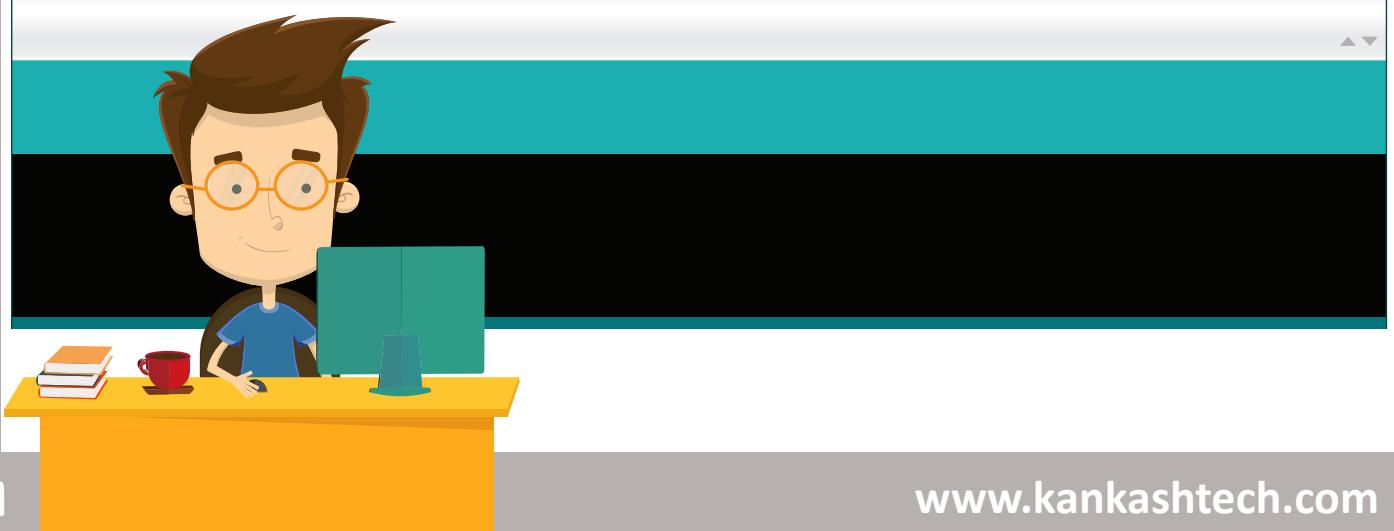


File > Examples > Electroduino > Circuit_8

```
Circuit_8

void ZERO () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, LOW);
}

void ONE () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, LOW);
}
```



سون مگنت - کاتد مشترک



File > Examples > Electroduino > Circuit_8

```
void TWO () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}

void THREE () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}
```





File > Examples > Electroduino > Circuit_8

```
Circuit_8

void FOUR () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}

void FIVE () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}
```



سون مگنت - کاتد مشترک

۱۱

File > Examples > Electroduino > Circuit_8



Circuit_8



```
void SIX () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}

void SEVEN () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, LOW);
}
```



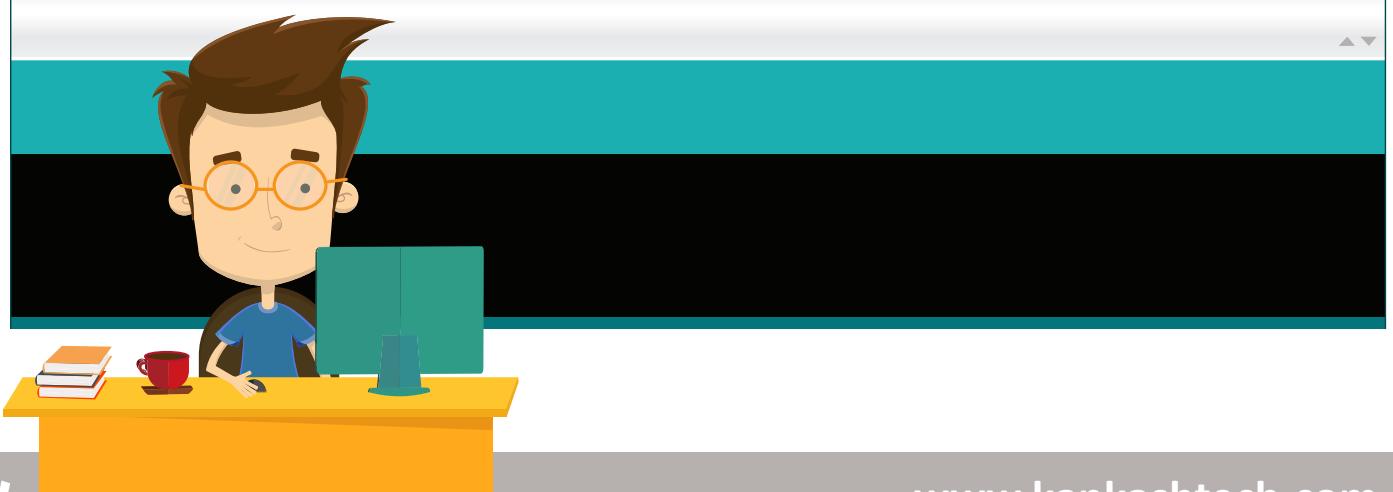


File > Examples > Electroduino > Circuit_8

```
Circuit_8

void EIGHT () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}

void NINE () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}
```



سون مگنت - کاتد مشترک

ا

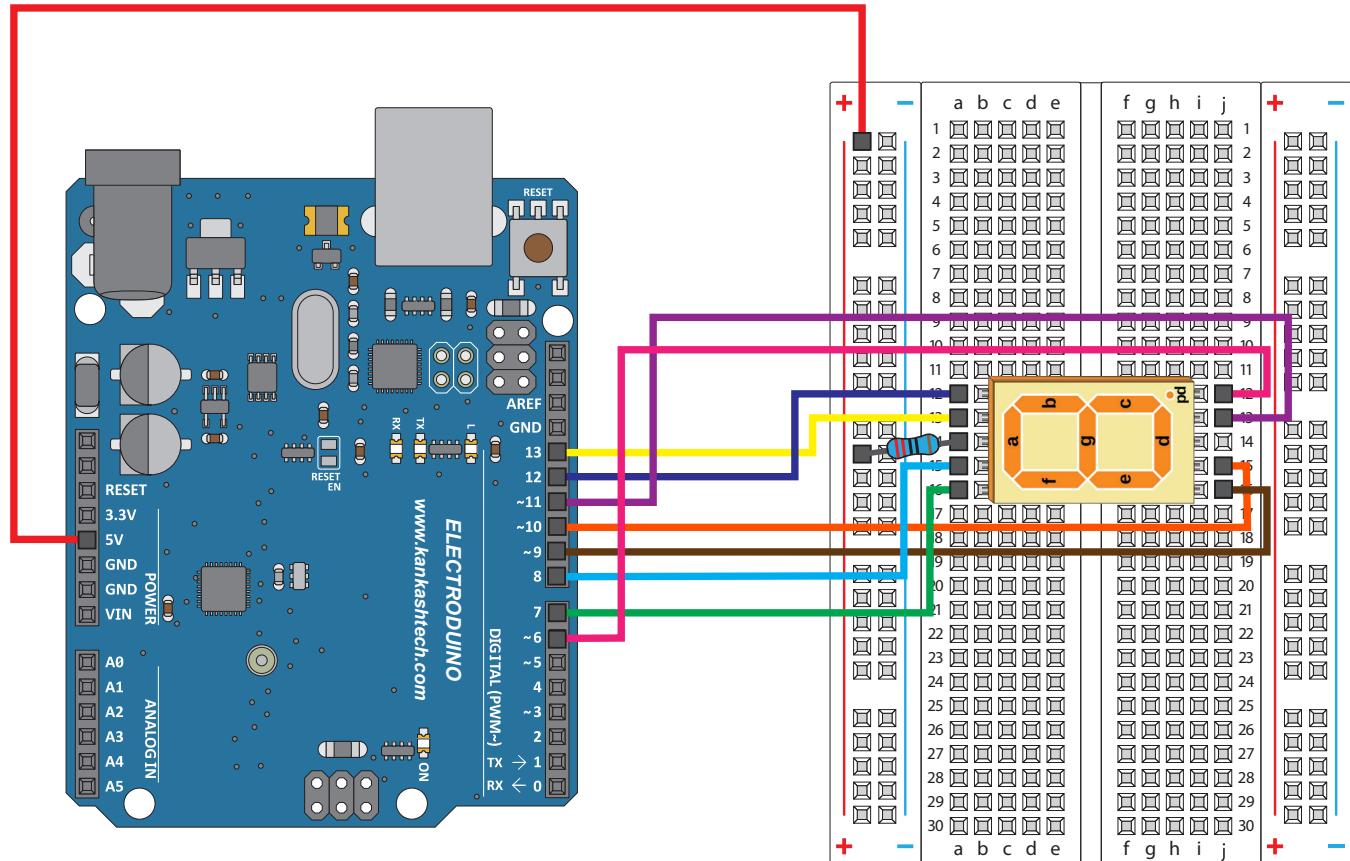
File > Examples > Electroduino > Circuit_8

```
Circuit_8

void aLetter () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}

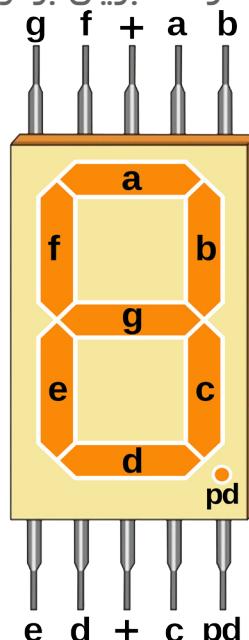
void bLetter () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, LOW);
}
```





توضیح مدار:

مدار سون سگمنت آند مشترک، مشابه گونه کاتد مشترک است با این تفاوت که پایه مشترک به ۵ ولت وصل می شود. چیدمان پایه های سون سگمنت آند مشترک مانند شکل زیر است. در سون سگمنت های کاتد مشترک، آندها (پایه های مثبت) به هم و به ۵ ولت متصل می شوند. برای روشن کردن هر سگمنت، باید پایه مربوطه **LOW** شود تا جریان برقرار شود.



کدنویسی سون سگمنت های آند مشترک، مشابه گونه های کاتد مشترک است، با این تفاوت که به جای **HIGH** از **LOW** و به جای **LOW** از **HIGH** استفاده می شود.



File > Examples > Electroduino > Circuit_8_2



Circuit_8_2

```
int SEG_a=13;
int SEG_b=12;
int SEG_c=11;
int SEG_d=10;
int SEG_e=9;
int SEG_f=8;
int SEG_g=7;
int SEG_pd=6;

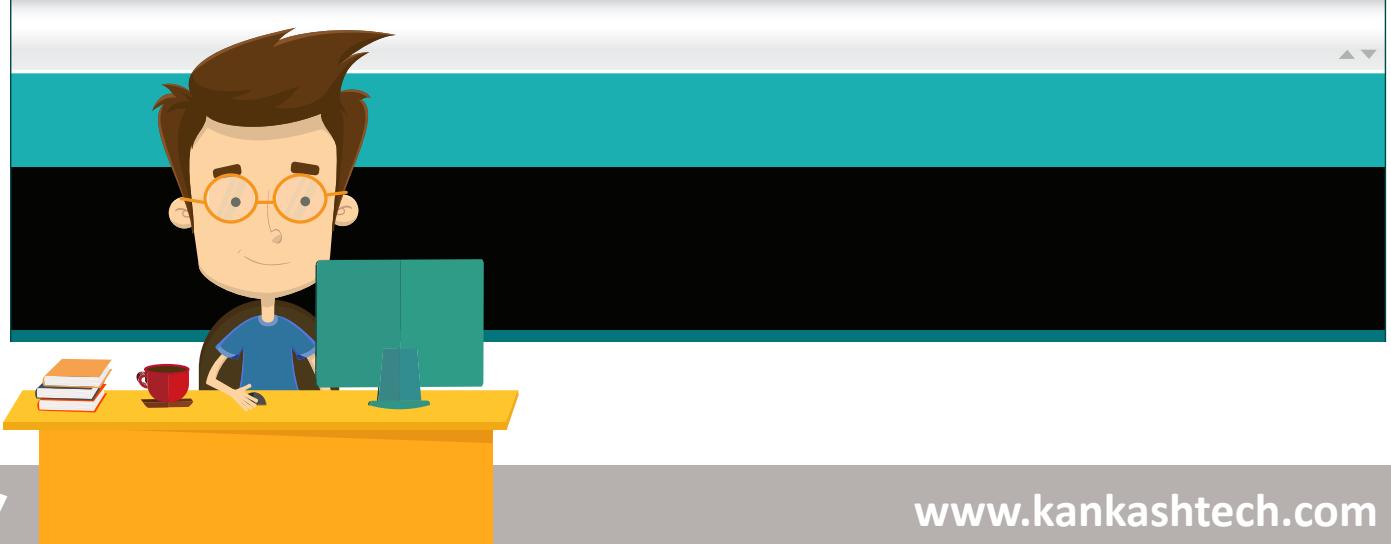
void setup() {
    pinMode (SEG_pd,OUTPUT);
    pinMode (SEG_c,OUTPUT);
    pinMode (SEG_d,OUTPUT);
    pinMode (SEG_e,OUTPUT);
    pinMode (SEG_f,OUTPUT);
    pinMode (SEG_g,OUTPUT);
    pinMode (SEG_a,OUTPUT);
    pinMode (SEG_b,OUTPUT);
}
```



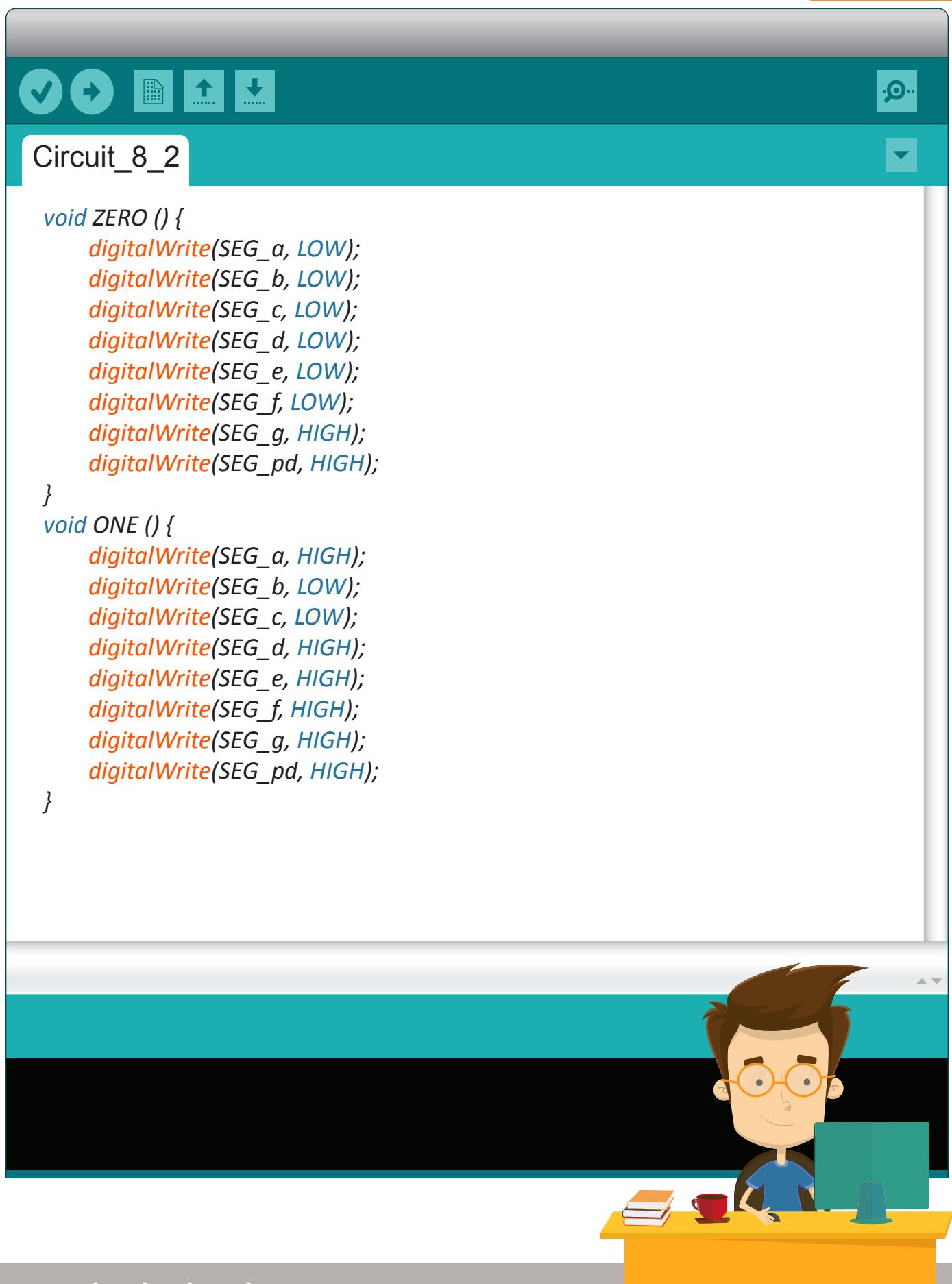


File > Examples > Electroduino > Circuit_8_2

```
void loop() {
    ZERO();
    delay(1000);
    ONE();
    delay(1000);
    TWO();
    delay(1000);
    THREE();
    delay(1000);
    FOUR();
    delay(1000);
    FIVE();
    delay(1000);
    SIX();
    delay(1000);
    SEVEN();
    delay(1000);
    EIGHT();
    delay(1000);
    NINE();
    delay(1000);
    aLetter();
    delay(1000);
    bLetter();
    delay(1000);
}
```



File > Examples > Electroduino > Circuit_8_2



```
Circuit_8_2

void ZERO () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, HIGH);
}

void ONE () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, HIGH);
    digitalWrite(SEG_pd, HIGH);
}
```



File > Examples > Electroduino > Circuit_8_2



Circuit_8_2

```
void TWO () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, HIGH);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}
void THREE () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, HIGH);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}
```



File > Examples > Electroduino > Circuit_8_2

```
void FOUR () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, OLW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, HIGH);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}

void FIVE () {
    digitalWrite(SEG_a, HIGH);
    digitalWrite(SEG_b, HIGH);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}
```



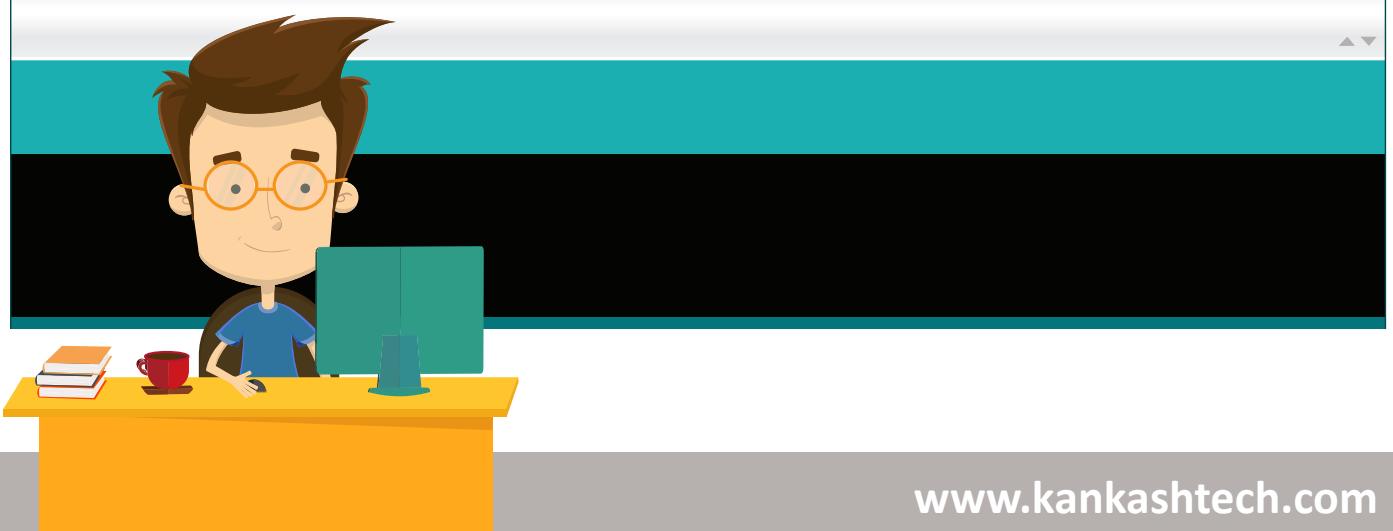


File > Examples > Electroduino > Circuit_8_2



Circuit_8_2

```
void SIX () {  
    digitalWrite(SEG_a, LOW);  
    digitalWrite(SEG_b, HIGH);  
    digitalWrite(SEG_c, LOW);  
    digitalWrite(SEG_d, LOW);  
    digitalWrite(SEG_e, LOW);  
    digitalWrite(SEG_f, LOW);  
    digitalWrite(SEG_g, LOW);  
    digitalWrite(SEG_pd, HIGH);  
}  
void SEVEN () {  
    digitalWrite(SEG_a, LOW);  
    digitalWrite(SEG_b, LOW);  
    digitalWrite(SEG_c, LOW);  
    digitalWrite(SEG_d, HIGH);  
    digitalWrite(SEG_e, HIGH);  
    digitalWrite(SEG_f, HIGH);  
    digitalWrite(SEG_g, HIGH);  
    digitalWrite(SEG_pd, HIGH);  
}
```



File > Examples > Electroduino > Circuit_8_2



Circuit_8_2

```
void EIGHT () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, LOW);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}

void NINE () {
    digitalWrite(SEG_a, LOW);
    digitalWrite(SEG_b, LOW);
    digitalWrite(SEG_c, LOW);
    digitalWrite(SEG_d, LOW);
    digitalWrite(SEG_e, HIGH);
    digitalWrite(SEG_f, LOW);
    digitalWrite(SEG_g, LOW);
    digitalWrite(SEG_pd, HIGH);
}
```





File > Examples > Electroduino > Circuit_8_2



Circuit_8_2

```
void aLetter () {  
    digitalWrite(SEG_a, LOW);  
    digitalWrite(SEG_b, LOW);  
    digitalWrite(SEG_c, LOW);  
    digitalWrite(SEG_d, HIGH);  
    digitalWrite(SEG_e, LOW);  
    digitalWrite(SEG_f, LOW);  
    digitalWrite(SEG_g, LOW);  
    digitalWrite(SEG_pd, HIGH);  
}  
void bLetter () {  
    digitalWrite(SEG_a, HIGH);  
    digitalWrite(SEG_b, HIGH);  
    digitalWrite(SEG_c, LOW);  
    digitalWrite(SEG_d, LOW);  
    digitalWrite(SEG_e, LOW);  
    digitalWrite(SEG_f, LOW);  
    digitalWrite(SEG_g, LOW);  
    digitalWrite(SEG_pd, HIGH);  
}
```



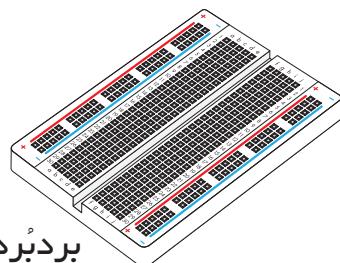
سوئیچ زاویه

سوئیچ زاویه (Tilt Switch) استفاده شده در مجموعه الکترودوینو یک سوئیچ با ساقمه فلزی است که عملکردی مشابه دکمه فشاری دارد. اصل کاری این سوئیچ بسیار ساده است. با جایجا شدن سوئیچ، ساقمه دور آن باعث اتصال دو پایه سوئیچ می شود. در غیر این صورت، ساقمه دور از پایه ها قرار گرفته و جریان قطع می شود. از این سوئیچ برای تشخیص تغییرات زاویه ای کوچک استفاده می شود.

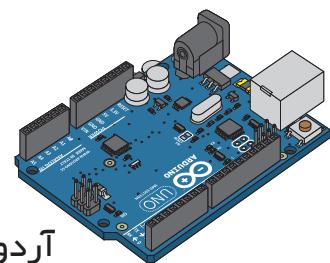
مطلوبات



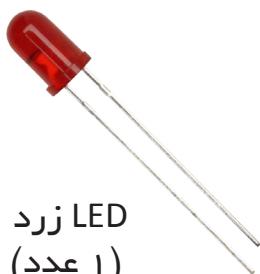
سیم جامپر نری-نری
(۲ عدد)



بردبرد



آردوینو



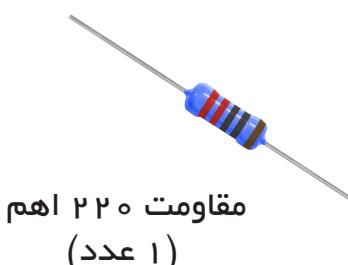
LED زرد
(۱ عدد)



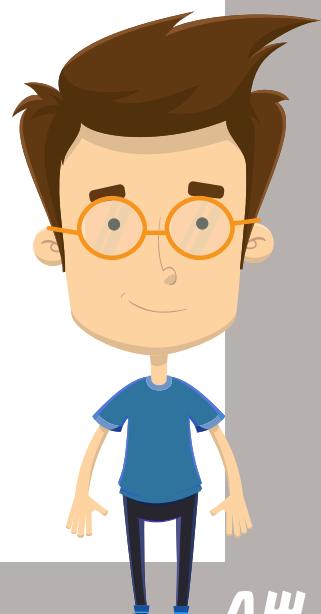
سوئیچ زاویه
(۱ عدد)

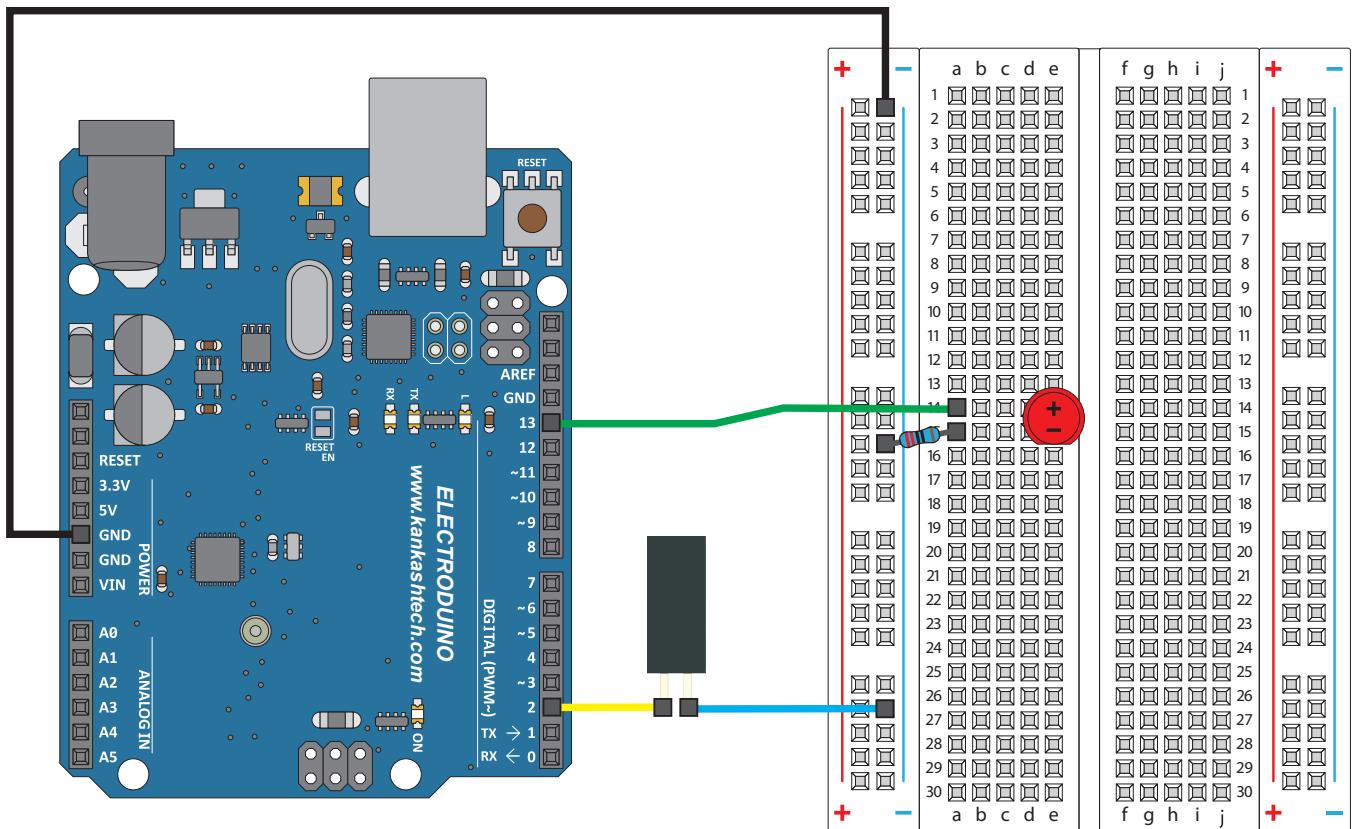


سیم جامپر نری-مادگی
(۲ عدد)



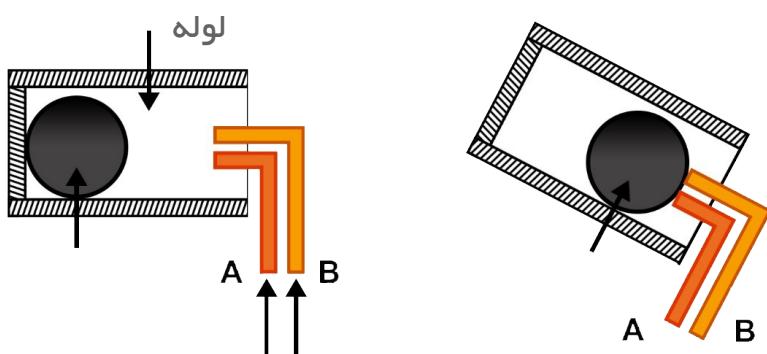
مقاومت ۲۵ اهم
(۱ عدد)





توضیح مدار:

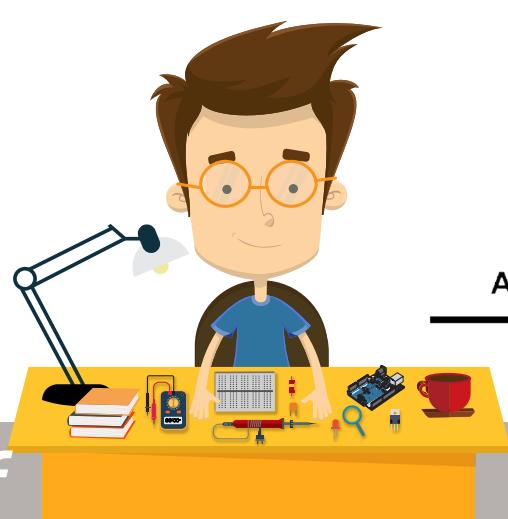
مدار این پروژه بسیار ساده است. یک پایه سوییچ زاویه به پین ۲ و دیگری به GND متصل می‌شود. یک LED و مقاومت به پایه ۱۳ متصل شده که در صورت فعال شدن سوییچ، روشن می‌شود.



با زاویه گرفتن سوییچ، ساقمه به پایین غلتیده و دو پایه را به هم وصل می‌کند



شماتیک عملکرد



سوئیچ زاویه

عملکرد سوئیچ زاویه شبیه عملکرد دکمه های فشاری پروژه ۳ است. در نتیجه به مقاومت PULLUP نیاز است.

آردوینو این قابلیت را دارد که یک پین را به طور داخلی PULLUP کند. بدین صورت که در نبود سیگنال، وضعیت پین را در حالت HIGH قرار می دهد. برای این کار، به جای تعریف پین به صورت ورودی، پین به صورت ورودی PULLUP تعریف می شود.

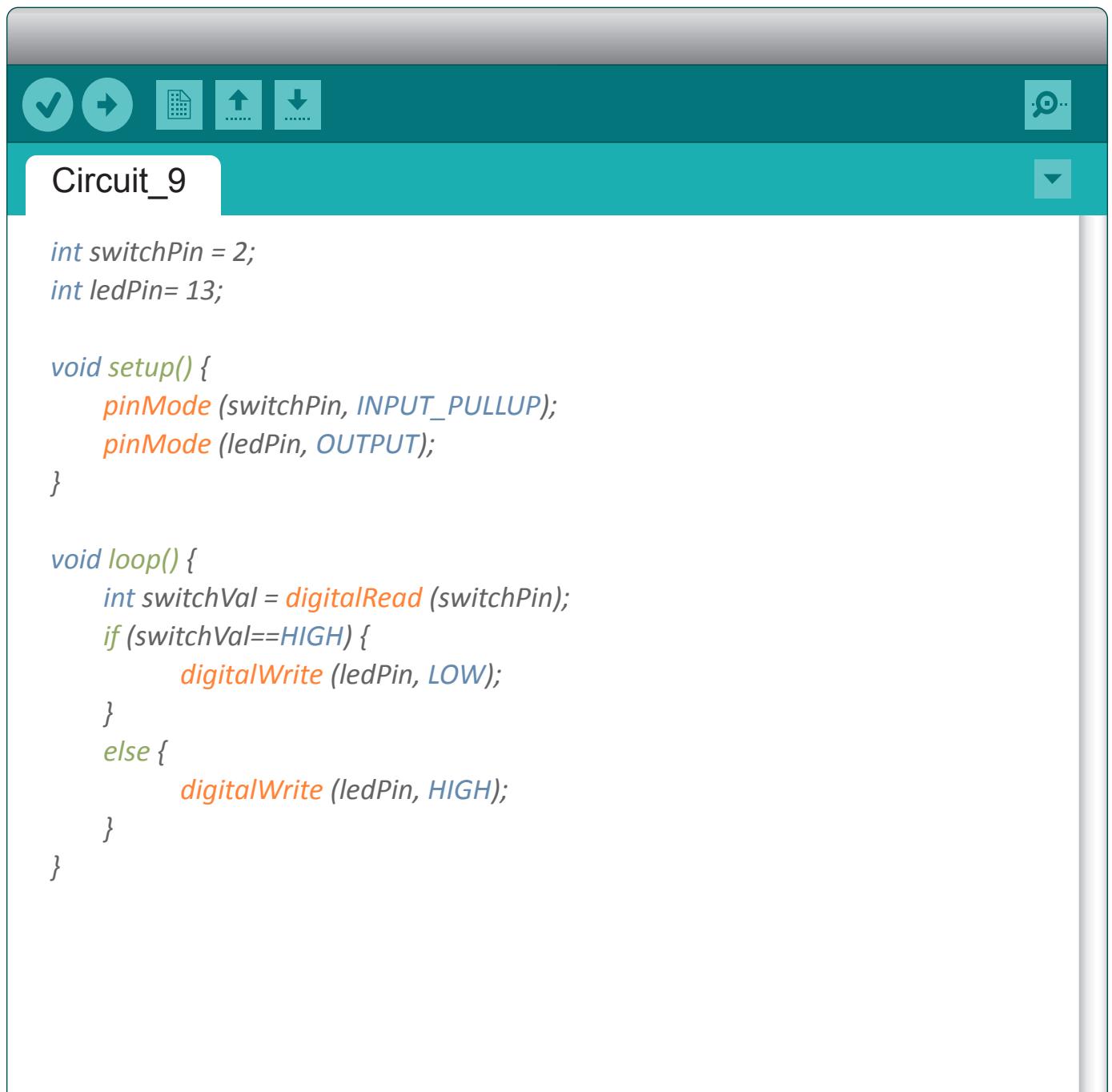
```
int switchPin = 2;
int ledPin= 13;

void setup() {
    pinMode (switchPin, INPUT_PULLUP);
    pinMode (ledPin, OUTPUT);
}
```

اگر با جابجا کردن سوئیچ، ساقمه باعث اتصال پایه های آن شود، پین ۲ به GND متصل شده و وضعیت آن LOW می شود. با استفاده از دستور digitalWrite وضعیت این پین مانیتور می شود. اگر پین ۲ در وضعیت HIGH باشد، ledPin در وضعیت LOW و در نتیجه LED متصل به پین ۱۳ خاموش می ماند. با عمل کردن سوئیچ، پین ۲ در وضعیت LOW قرار گرفته و در نتیجه LED روشن می شود.

```
void loop() {
    int switchVal = digitalRead (switchPin);
    if (switchVal==HIGH) {
        digitalWrite (ledPin, LOW);
    }
    else {
        digitalWrite (ledPin, HIGH);
    }
}
```

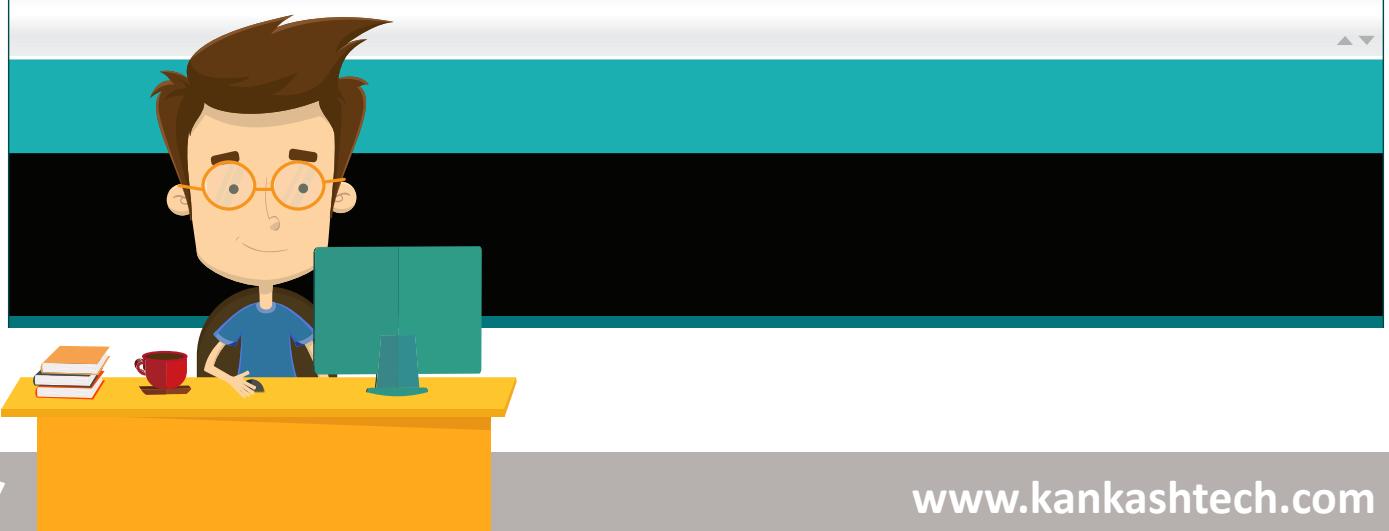




```
int switchPin = 2;
int ledPin= 13;

void setup() {
    pinMode (switchPin, INPUT_PULLUP);
    pinMode (ledPin, OUTPUT);
}

void loop() {
    int switchVal = digitalRead (switchPin);
    if (switchVal==HIGH) {
        digitalWrite (ledPin, LOW);
    }
    else {
        digitalWrite (ledPin, HIGH);
    }
}
```



سنسور دما



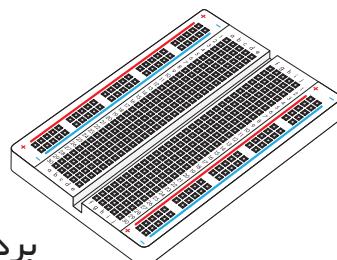
سنسور دما همانگونه که از نامش پیداست، برای اندازه گیری تغییرات دما استفاده می شود. سنسوری که در این پروژه استفاده می شود از نوع LM35 است. این سنسور سه پایه مثبت (VCC) منفی (GND) و سیگنال دارد.

به ازا هر درجه دمایی که این سنسور می خواند، ۱ میلی ولت خروجی می دهد. در این پروژه نحوه استفاده از این سنسور به همراه آردوبینو و نحوه استفاده از مانیتور سریال IDE را توضیح خواهد شد.

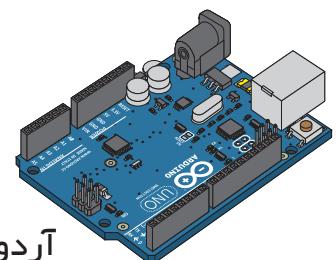
مطلوبات



سیم جامپر نری-نری
(۵ عدد)



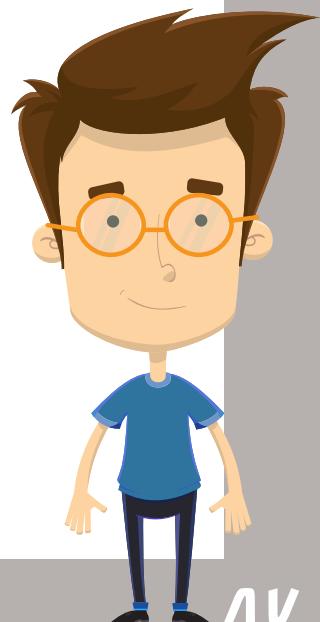
بردبرد

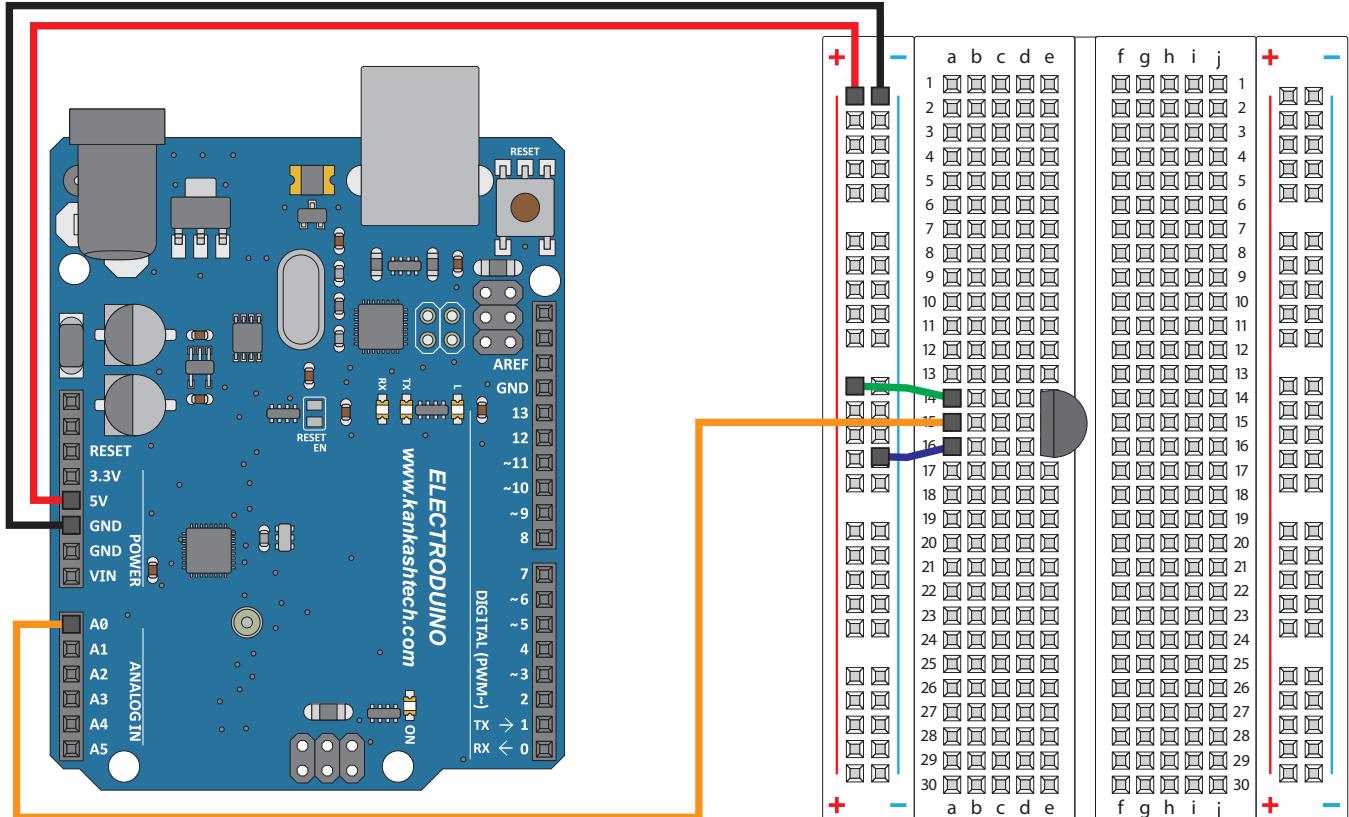


آردوبینو



سنسور دما LM35
(۱ عدد)





توضیح مدار:

همانطور که در مقدمه گفته شد، سنسور دما دارای سه پایه مثبت، منفی و سیگنال است. پایه مثبت (VCC) به پین ۵ ولت آردوینو، پایه منفی (GND) به یکی از پین های GND آردوینو و پایه سیگنال به پین ورودی آنالوگ A0 متصل می شود.





مانیتور سریال

مانیتور سریال (Serial Monitor) یکی از بخش های IDE و رابط میان کامپیوتر و برد آردوینو است که امکان ارسال و دریافت اطلاعات بین کامپیوتر و آردوینو را فراهم می سازد. این ابزار برای عیب یابی برنامه ها و همچنین کنترل آردوینو استفاده می شود. برای استفاده از این ابزار ابتدا باید آنرا راه اندازی کرد:

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

این دستور سرعت مبادله اطلاعات را بر حسب بیت بر ثانیه مشخص می کند. ۹۶۰۰ یکی از سرعت های رایج برای مبادله اطلاعات میان آردوینو و کامپیوتر است. دستورات رایج سریال مانیتور عبارتند از:

Serial.print(val, format)

داده ها به صورت متن ASCII قابل خواندن نمایش می دهد. اعداد به صورت کاراکتر ASCII هر رقم نمایش داده می شوند. اعداد float به صورت رقم های ASCII با دو رقم اعشار نمایش داده می شوند. بایت ها به صورت یک کاراکتر نشان داده می شوند. کاراکتر ها و رشته ها به همان صورتی که هستند نمایش داده می شوند. به عنوان مثال:

```
Serial.print(78) : "78"
```

```
Serial.print(1.23456) : "1.23"
```

```
Serial.print('N') : "N"
```

```
Serial.print("Hello world.") : "Hello world."
```

پارامتر format که پارامتری اختیاری است می تواند مقادیر BIN برای مبنای ۲، OCT برای مبنای ۸، DEC برای مبنای ۱۰ و HEX برای مبنای ۱۶ داشته باشد (در پروژه های بعد با ویژگی ها و کاربردهای مبناهای مختلف آشنا خواهید شد). برای اعداد **float**، این پارامتر به معنای تعداد رقم اعشار است:

```
Serial.print(78, BIN) : "1001110"
```

```
Serial.print(78, OCT) : "116"
```

```
Serial.print(78, DEC) : "78"
```

```
Serial.print(78, HEX) : "4E"
```

```
Serial.print(1.23456, 0) : "1"
```

```
Serial.print(1.23456, 2) : "1.23"
```

```
Serial.print(1.23456, 4) : "1.2346"
```



Serial.println(val, format)

عملکردی مشابه دستور `Serial.print` دارد، با این تفاوت که در انتهای مقدار نمایش داده شده، یک کاراکتر "Enter" قرار می‌دهد و مقدار بعدی در ابتدای خط بعد نمایش داده می‌شود.

برای مشاهده مانیتور سریال، از آیکون ذره بین در گوشه بالا سمت راست IDE استفاده می‌شود. پس از آپلود کد به آردوینو، مانیتور سریال، وضعیت لحظه‌ای عملکرد آردوینو را نشان می‌دهد

محاسبه دما

همانگونه که گفته شد، سنسور LM35 به ازا هر درجه سانتیگراد تغییر دما، ۱۰ میلی ولت خروجی می‌دهد. با توجه به اینکه مقادیر خوانده شده توسط پین آنالوگ برای ۵ ولت ۱۰۲۳ و برای ۰ ولت، ۰ است، می‌توان با استفاده از تناسب، مقدار خوانده شده توسط پین آنالوگ را به دما تبدیل کرد: ۵ ولت برابر با ۵۰۰۰ میلی ولت و به معنای ۵۰۰ درجه سانتیگراد است. اگر `sensorValue` مقدار خوانده شده از پین آنالوگ و `tem` بیانگر دما باشد:

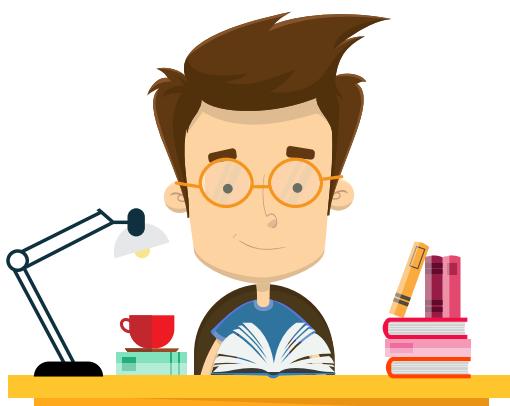
$$\text{sensorValue}/1023 = \text{tem}/500$$

در نتیجه:

$$\text{tem} = \text{sensorValue} * 500 / 1023 = 0.488 * \text{sensorValue}$$

برای افزایش دقیق محاسبات، `sensorValue` به عنوان متغیر `float` تعریف می‌شود که می‌تواند اعشار نیز داشته باشد. ولی با توجه به اینکه دقیق سنسور طبق دیتاشیت سازنده برابر ۰.۵ درجه سانتیگراد است، متغیر `tem` به عنوان عدد صحیح یا `int` تعریف می‌شود.

پس از هر بار محاسبه دما، مقدار اندازه گیری شده در یک متغیر به نام `lastValue` ذخیره می‌شود. تنها در صورتی که دمای اندازه گیری شده با دمای پیشین ذخیره شده در `lastValue` متفاوت بود، دمای جدید در مانیتور سریال نمایش داده می‌شود. این کار باعث جلوگیری چاپ سریع و مکرر مقادیر در مانیتور سریال می‌شود.



File > Examples > Electroduino > Circuit_10

```
int lastValue=0;
int tem=0;
int lmPin = A0;
void setup() {
    Serial.begin(9600);
}

void loop() {
    float sensorValue = analogRead(lmPin);
    tem = 0.488 * sensorValue;
    if (tem != lastValue) {
        Serial.print("Temperature: ");
        Serial.print (tem);
        Serial.print(char(176));
        Serial.println("C");
        lastValue=tem;
    }
    delay(100);
}
```



سنسور دما و رطوبت



ماژول DHT11 یک سنسور دما و رطوبت دیجیتال است که خروجی دیجیتال برای دما و رطوبت ایجاد می‌کند.

این کار توسط یک پردازنده کوچک درون پوسته سنسور انجام می‌شود که تبدیل آنالوگ به دیجیتال را انجام می‌دهد.

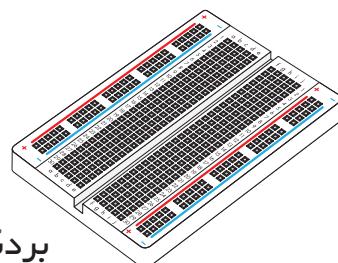
این سنسور دقیق و سرعت پایین تر از LM35 دارد و نمی‌تواند بیش از یکبار در ثانیه خروجی دهد. بازه عملکرد مناسب این سنسور دمای $0\text{--}50^{\circ}\text{C}$ درجه سانتیگراد و رطوبت $20\text{--}80\%$ درصد است.

در این پروژه از کتابخانه مربوط به این سنسور و دستورهای پیش فرض آن استفاده می‌شود.

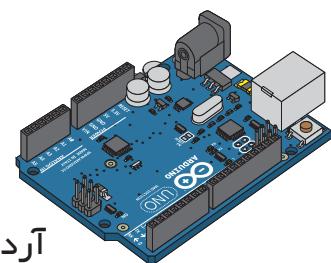
مطلوبات



سیم جامپر نری-نری
(۵ عدد)



بردبرد



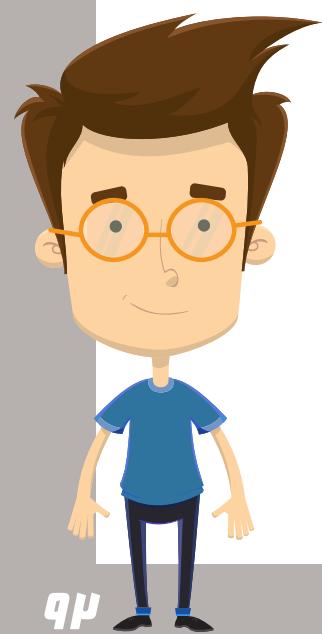
آردوینو

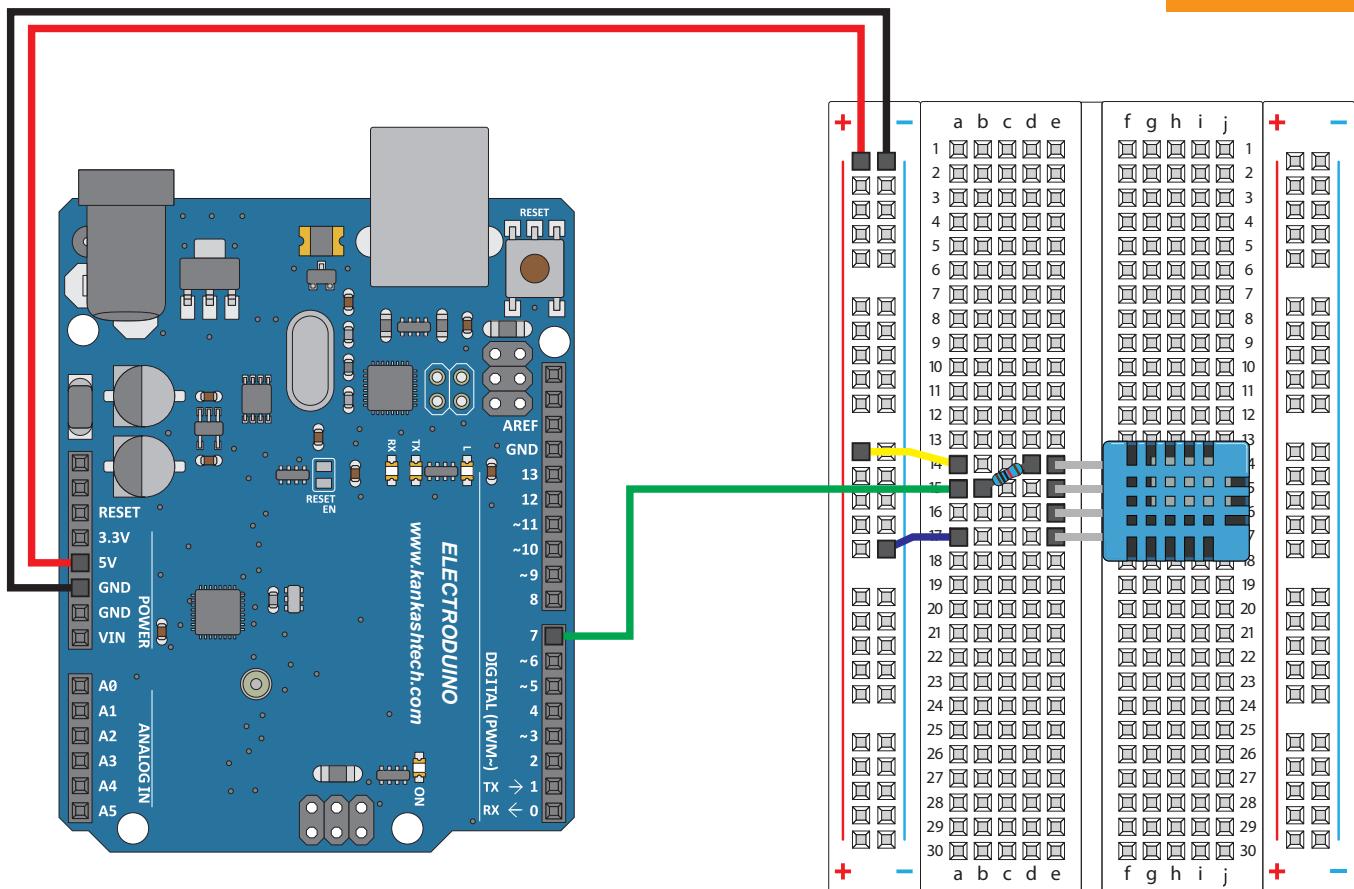


سنسور دما و رطوبت
DHT11
(۱ عدد)



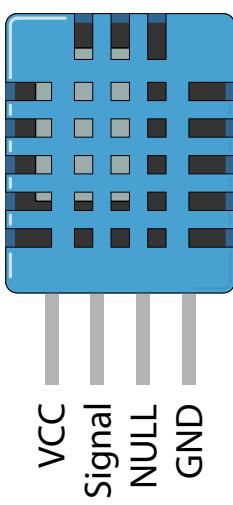
مقاومت ۱۰ کیلو اهم
(۱ عدد)



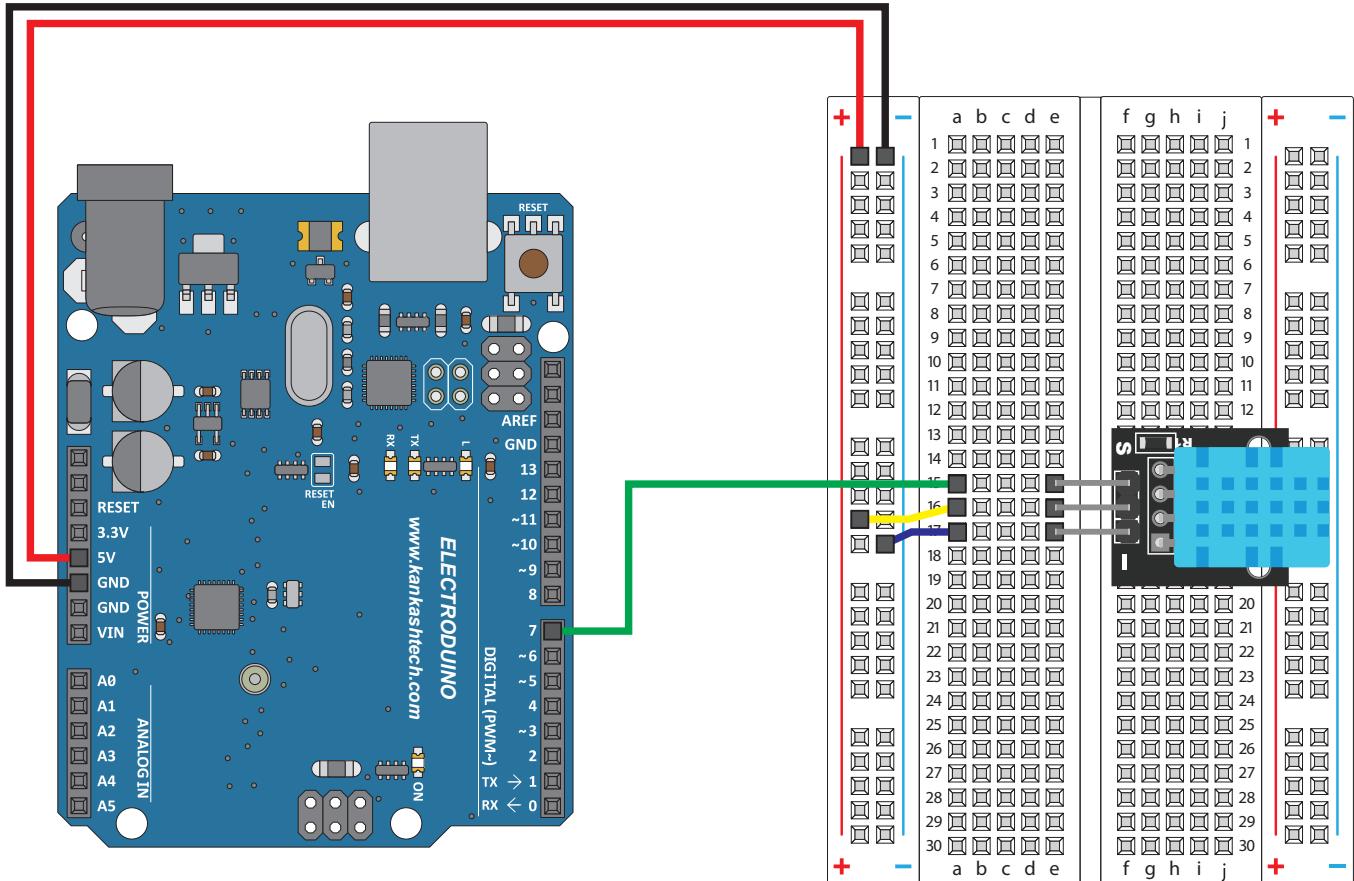


توضیح مدار:

مدار این پروژه مشابه پروژه 10 است، با این تفاوت که به جای اتصال پایه سیگنال به پین آنالوگ A0، این پایه به پین دیجیتال 7 آردوینو متصل می شود. همچنین یک مقاومت ۱۰ کیلواهرمی به عنوان مقاومت pullup میان پایه VCC و DATA و سنسور قرار داده می شود. در برخی منابع، پایه VCC شماره ۱، DATA شماره ۲، NULL شماره ۳ و GND شماره ۴ خوانده می شوند. پایه های سنسور در شکل زیر نشان داده شده است.



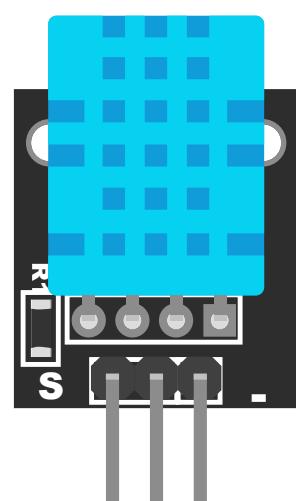
ماژول سنسور دما و رطوبت



در صورت استفاده از ماژول DHT11 نیازی به استفاده از مقاومت نیست و مقاومت مورد نیاز بر روی بُرد ماژول تعییه شده است.

در هنگام استفاده از ماژول ها باید به علائم نوشته شده در کنار پایه ها دقت کرد. در ماژول ارائه شده، پایه ای که با علامت S مشخص شده است به پین ۷، پایه وسط به ستون مثبت و پایه مشخص شده با علامت - به ستون منفی متصل می شود.

کد مورد استفاده برای سنسور و ماژول یکسان است



سنسور دما و رطوبت



برای استفاده از سنسور DHT11 ابتدا باید کتابخانه مربوط به آن فراخوانی شود. کتابخانه مربوط به LCD نیز باید فراخوانی شود:

```
#include <dht.h>
```

سپس باید متغیر سنسور تعریف شوند. در اینجا یک سنسور DHT با نام MYDHT تعریف می شود:
`dht MYDHT;`

سپس پین 7 به عنوان پین سیگنال DHT با نام dhtPin تعریف می شود:
`int dhtPin= 7;`

```
Serial.begin(9600);  
pinMode(dhtPin, INPUT);
```

پس از تعریف ماژول و پین ها و راه اندازی مانیتور سریال، وضعیت سنسور خوانده می شود. برای خواندن DHT11 از دستور `read11` استفاده می شود. عبارت قبل از نقطه، نامی است که در هنگام تعریف، برای سنسور انتخاب شده است:

```
MYDHT.read11(dhtPin);  
Serial.print("Tem:");
```

برای خواندن دما، از دستور `temperature` استفاده می شود. به صورت پیش فرض، دما (و رطوبت) با دو رقم اعشار خوانده می شود. برای محدود کردن تعداد رقم اعشار به صورت زیر عمل می شود. به عنوان مثال برای نمایش یک رقم اعشار:

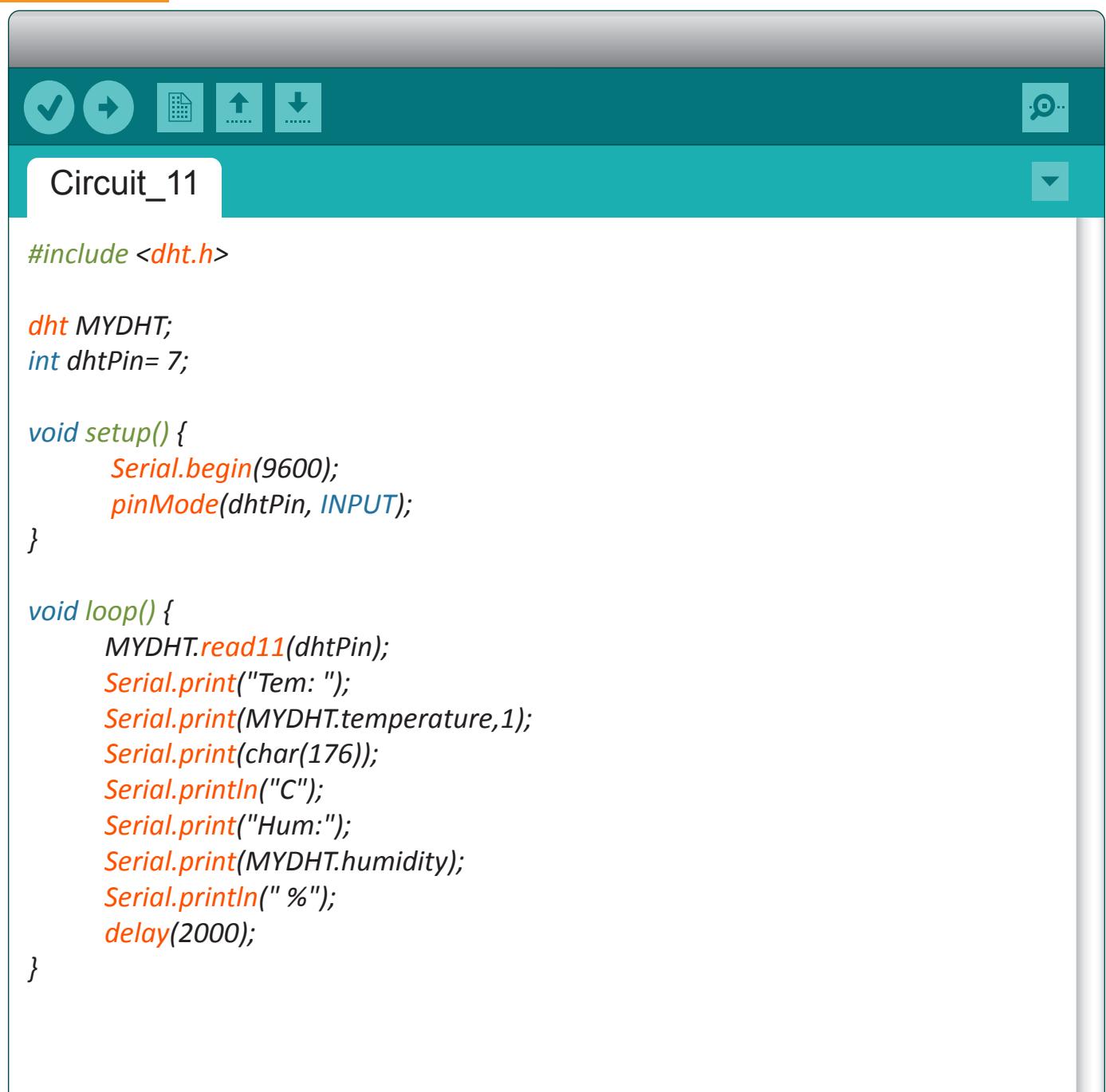
```
Serial.print(MYDHT.temperature, 1);
```

```
Serial.print(char(176));  
Serial.println("C");  
Serial.print("Hum: ");  
Serial.print(MYDHT.humidity, 0);  
Serial.print("%");  
delay(2000);
```

با توجه به دقیقیت این سنسور، نمایش بدون اعشار پیشنهاد می شود.



File > Examples > Electroduino > Circuit_11

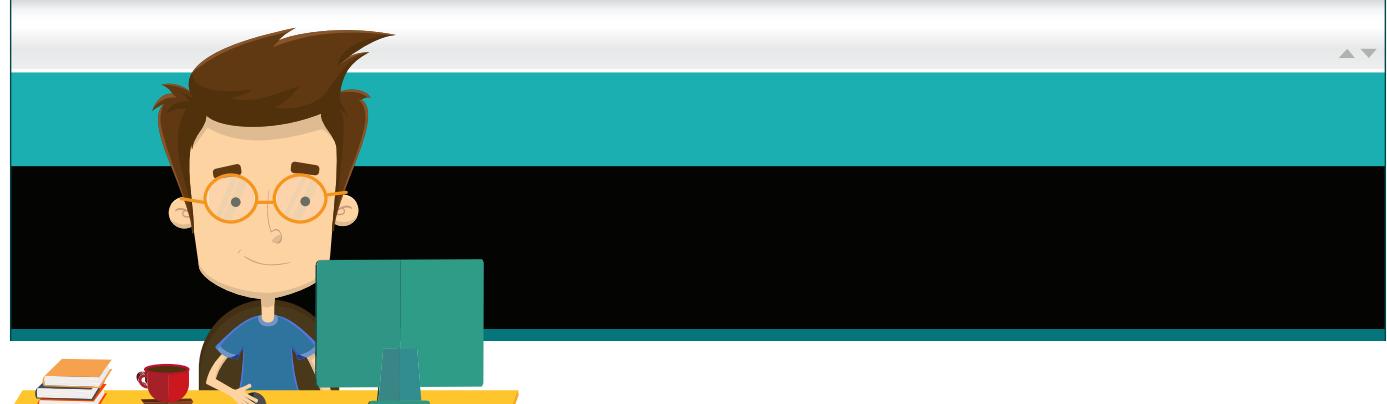


```
#include <dht.h>

dht MYDHT;
int dhtPin= 7;

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

void loop() {
    MYDHT.read11(dhtPin);
    Serial.print("Tem: ");
    Serial.print(MYDHT.temperature,1);
    Serial.print(char(176));
    Serial.println("C");
    Serial.print("Hum:");
    Serial.print(MYDHT.humidity);
    Serial.println(" %");
    delay(2000);
}
```



LCD 1602

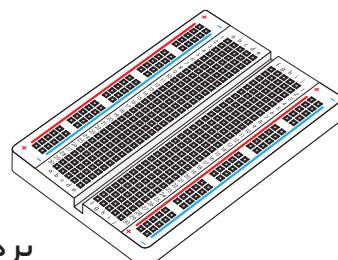
LCD 1602، یک صفحه نمایش دو رده ای با قابلیت نمایش 16 کاراکتر در هر ردیف است. با توجه به اینکه در این پروژه نیز از کتابخانه مربوط به این قطعه استفاده می شود، عمدۀ کاراکترهای مورد نیاز جهت نمایش، به صورت آماده نوشته شده و قابل فراخوانی است.

در این پروژه، دما و رطوبت خوانده شده توسط سنسور DHT11 بر روی LCD نمایش داده می شود. دستورات مختلف کار با LCD1602 در توضیحات پروژه ارائه شده که می توان عملکردهای مختلف LCD را توسط آن مشاهده کرد. (از این دستورها در این پروژه استفاده نشده است).

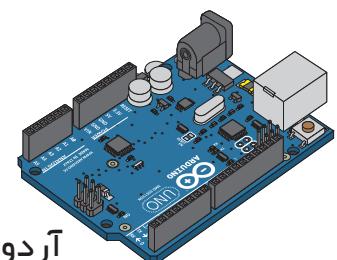
مطلوبات



سیم جامپر نزی-نری
(۵ عدد)



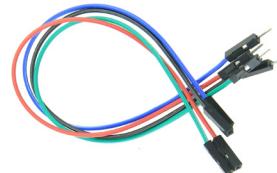
بردبرد



آردوینو



مقاومت ۱۰ کیلو اهم
(۱ عدد)



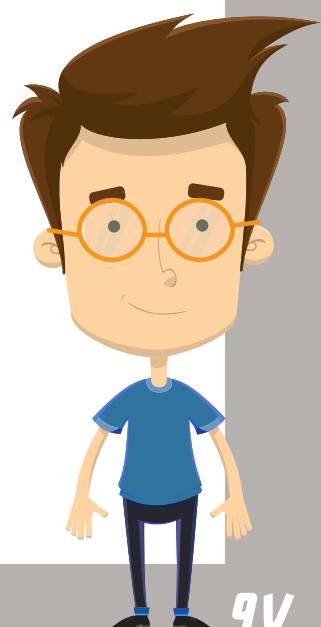
سیم جامپر نزی - مادگی
(۴ عدد)

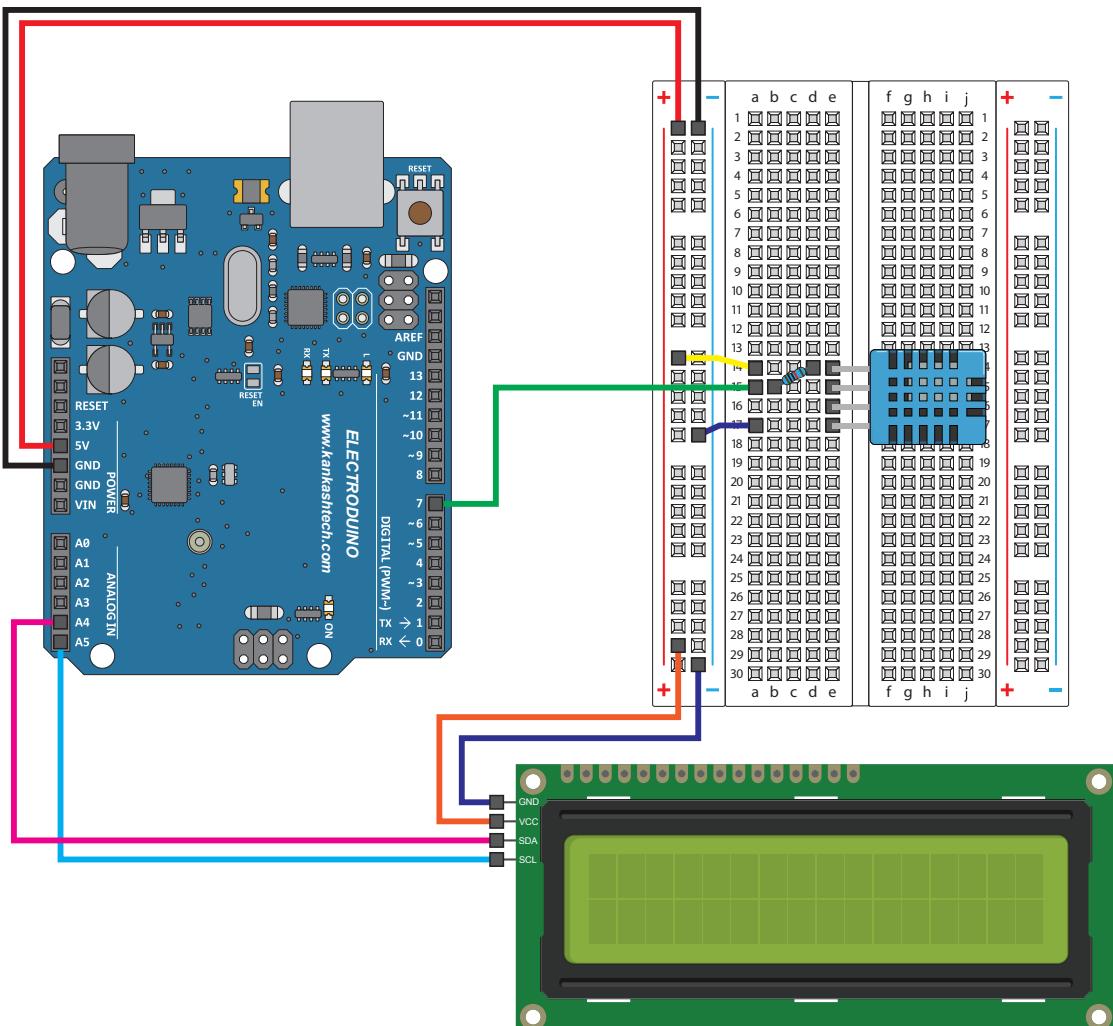


LCD1602-I2C
(۱ عدد)



سنسور دما و رطوبت DHT11
(۱ عدد)





توضیح مدار:

LCD موجود در این مجموعه از یک رابط I2C استفاده کرده و در نتیجه برای کنترل آن، به جز VCC و GND تنها از دو پین آردوینو استفاده می شود. پایه های VCC و GND رابط I2C به پین های ۵ ولت و GND آردوینو متصل می شوند. پایه SDA به پین آنالوگ شماره ۴ (A4) و پایه SCL به پین آنالوگ شماره ۵ (A5) متصل می شوند.

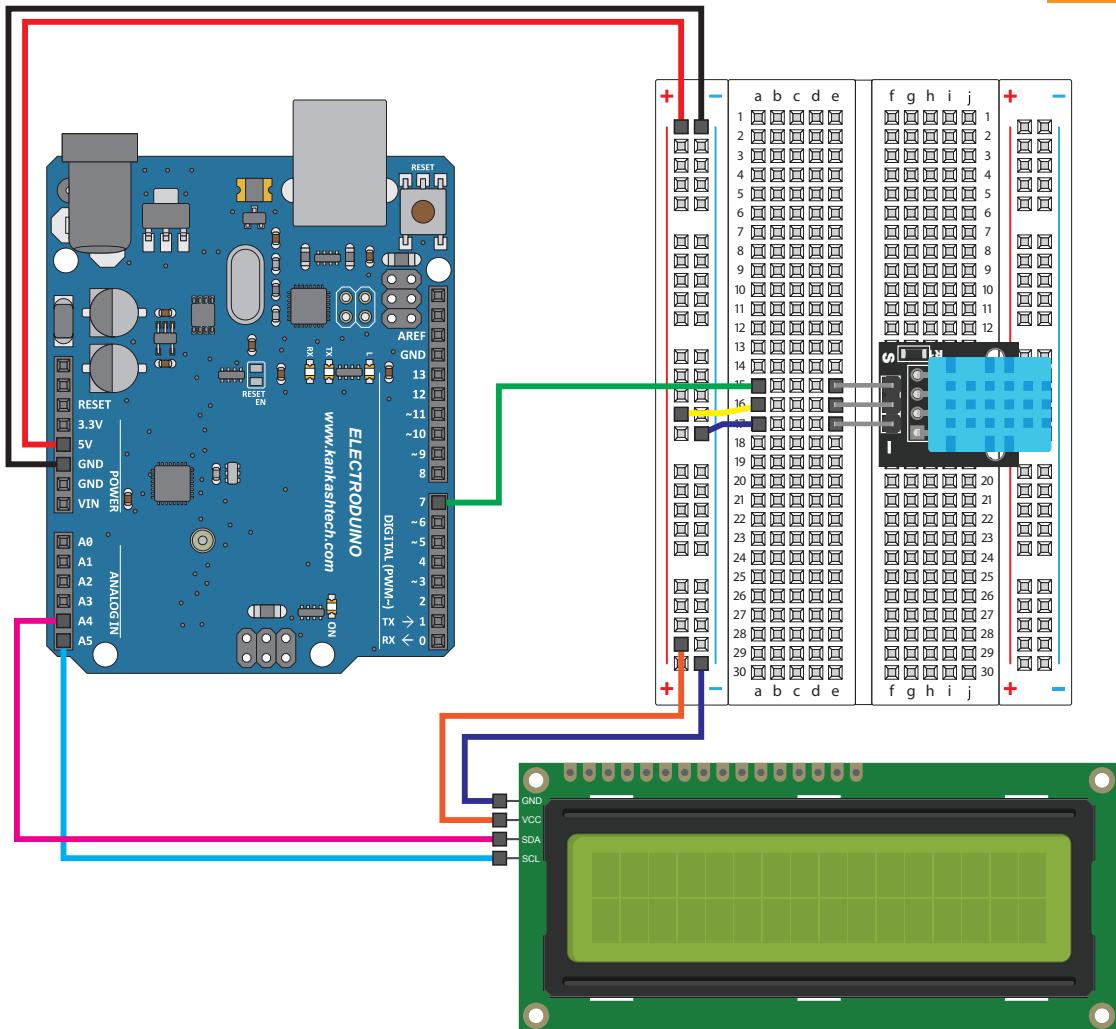


بخش مربوط به خواندن دما و رطوبت، مانند پروژه ۱۱ است.

یک پتاستنیومتر برای تنظیم وضعیت کاراکتر در پشت ماژول قرار دارد. با استفاده از پیچ گوشتی چهارسوی موجود در مجموعه می توانید وضعیت کاراکتر ها را تنظیم کنید.

(DHT11 مدار مازول) LCD 1602

IP



توضیح مدار:

در صورت استفاده از مازول باید به علائم کنار پایه ها دقت کرد. پایه مشخص شده با علامت ۵ به پین ۷، پایه وسط به ستون مثبت و پایه مشخص شده با علامت - به ستون منفی متصل می شود. با توجه به اینکه مقاومت مورد نیاز برروی مازول تعییه شده است، در هنگام استفاده از مازول نیاز به استفاده از مقاومت جداگانه نیست.



برای استفاده از LCD 1602 ابتدا باید کتابخانه مربوط به آن فراخوانی شود:

```
#include <LiquidCrystal_I2C.h>
```

سپس باید مازول LCD تعریف شود:

```
LiquidCrystal_I2C lcd(0x3F,16,2);
```

در کد بالا، lcd به عنوان یک نمایشگر از نوع LiquidCrystal_I2C تعریف شده که دو سطر و ۱۶ کاراکتر را می‌تواند نمایش دهد.

پارامتر ۰x3F آدرس این مازول برای شناسایی توسط پردازنده آردوبینو است که تشریح آن از چارچوب این مجموعه خارج است. در برخی انواع LCD ها این آدرس ۰x27 است.

پس از تعریف lcd، باید این مازول راه اندازی شود. برای مشاهده بهتر کاراکترها، در این مازول یک نور پس زمینه نیز قرار داده شده است که مربوط به آن، در زمان راه اندازی باید فراخوانده شود. می‌توان با استفاده از پتانسیومتر آبی رنگ موجود بر روی رابط I2C (پشت LCD)، میزان وضوح کاراکترها را تنظیم کرد:

```
lcd.init(); //initialize the lcd
lcd.backlight(); //open the backlight
```

در برخی از کتابخانه ها برای راه اندازی، از دستور lcd.begin() استفاده می‌شود. عبارت قبل از نقطه، همان نام LCD تعریف شده در بالاست (به عنوان مثال اگر به جای lcd نام display استفاده می‌شد، دستور راه اندازی باید به صورت display.init() نوشته می‌شد) دستورات اصلی برای استفاده از این نمایشگر عبارتند از:

setCursor

```
lcd.setCursor(15, 1);
```

پارامتر اول مربوط به موقعیت کاراکتر (۱۵-۰) و پارامتر دوم مربوط به سطر (۱-۰) است. باید دقت شود که این پارامترها از صفر شروع می‌شوند.

print

برای نمایش از دستور print استفاده می‌شود. این دستور، کاراکتر یا رشته مشخص شده را در موقعت تعیین شده نمایش می‌دهد.

```
String text= "Electroduino";
lcd.print(text);
lcd.print("Electroduino");
```

می‌توان مستقیماً یک رشته را نمایش داد. در این حالت رشته مورد نظر باید میان " " قرار گیرد:

همچنین می‌توان از آدرس کاراکتر برای نمایش استفاده کرد. به عنوان مثال، کاراکتر علامت درجه، در آدرس ۱۷۶ ذخیره شده است. برای نمایش این کاراکتر:

```
lcd.print(char(176));
```



scrollDisplayLeft

```
lcd.scrollDisplayLeft();
```

دستور فوق رشته نمایش داده شده بر روی نمایشگر را یک کاراکتر به سمت چپ جابجا می کند. کاراکتری که در موقعیت ۰ قرار دارد از نمایشگر خارج و کاراکتری که بعد از کاراکتر موقعیت ۱۵ قرار دارد در موقعیت ۱۵ قرار داده می شود.

scrollDisplayRight

```
lcd.scrollDisplayRight();
```

دستور فوق رشته نمایش داده شده بر روی نمایشگر را یک کاراکتر به سمت راست جابجا می کند.

clear

```
lcd.clear();
```

دستور فوق تمام داده های در حال نمایش بر LCD را پاک می کند.

تعریف کاراکتر

جزییات مربوط به چگونگی ایجاد کاراکتر، در درس ۱۳ (ماتریس 8×8) به طور کامل تشریح شده است. اگر از نزدیک به صفحه LCD دقیق تر کنید، مشاهده می کنید که هر یک از ۳۲ جایگاه کاراکتر، از ۸ ردیف و ۵ ستون پیکسل های کوچک تشکیل شده است.

برای تعریف کاراکتر، ابتدا باید یک آرایه حداقل ۸ تایی برای آن تعریف شود. در این آرایه از نوع بایت بوده و توسط byte تعریف می شوند. (در دستور زیر، در این آرایه ها اعدادی در مبنای ۱۶ (HEX) هستند. توضیحات بیشتر در پروژه ۱۳ ارائه شده است). هر کدام از ۸ در این آرایه ها مربوط به یک سطر هستند و بیانگر پیکسل های روشن در آن سطر هستند.

```
byte bell[8] = {4, 14, 14, 14, 31, 0, 4, 0};
```

```
lcd.createChar(0, bell);
```

در دستور فوق، یک کاراکتر به شکل زنگوله برای آدرس ۰ تعریف می شود که می توان به صورت زیر آن را نمایش داد:

```
lcd.print(char(0));
```





File > Examples > Electroduino > Circuit _12



Circuit_12



```
#include <LiquidCrystal_I2C.h>
#include <dht.h>

LiquidCrystal_I2C lcd(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up
dht MYDHT;
int dhtPin= 7;

void setup() {
    pinMode(dhtPin, INPUT);
    lcd.init();
    lcd.backlight();
    lcd.setCursor (1,0);
    lcd.print("KankashTech.com");
    delay (3000);
    for (int cur=0; cur<16; cur++){
        lcd.scrollDisplayLeft();
        delay(500);
    }
}
```



LCD 1602



File > Examples > Electroduino > Circuit_12

```
void loop() {
    MYDHT.read11(dhtPin);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Temperature: ");
    lcd.print(MYDHT.temperature, 1);
    lcd.print(char(176));
    lcd.print("C");
    lcd.setCursor(0, 1);
    lcd.print("Humidity: ");
    lcd.print(MYDHT.humidity);
    lcd.print(" %");
    delay(2000);
}
```





سنسور سطح آب

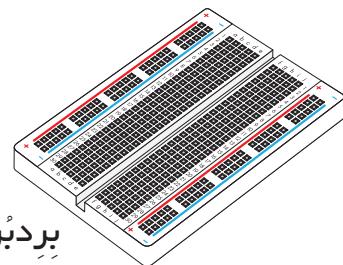
ماژول اندازه گیری سطح آب یک سنسور برای اندازه گیری عمق آب است. بخش اصلی این سنسور یک مدار تقویت کننده شامل یک ترانزیستور و چندین سیم مدار شانه ای شکل است. هنگامی که سیم های شانه ای در آب قرار می گیرند، با تغییر عمق آب، مقاومت آن تغییر می کند و در نتیجه ولتاژ متفاوتی ارسال می کند.

آردوینو با اندازه گیری این ولتاژ، می تواند عمق آب را اندازه گیری کند. در این پروژه نحوه نمایش مقدار سنسور خوانده شده توسط آردوینو و عمق آب متناظر با این مقدار ارائه شده است.

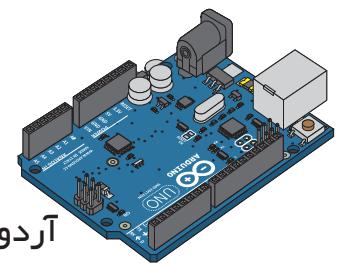
مطلوبات



سیم جامپر نری-نری
(۲ عدد)



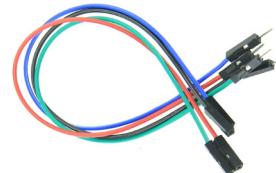
بردبرد



آردوینو



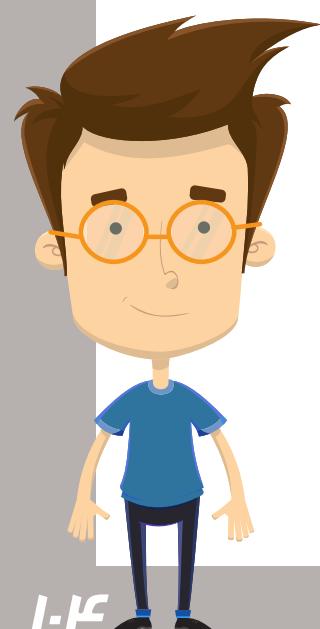
سنسور سنجش سطح آب
(۱ عدد)



سیم جامپر نری - مادگی
(۷ عدد)

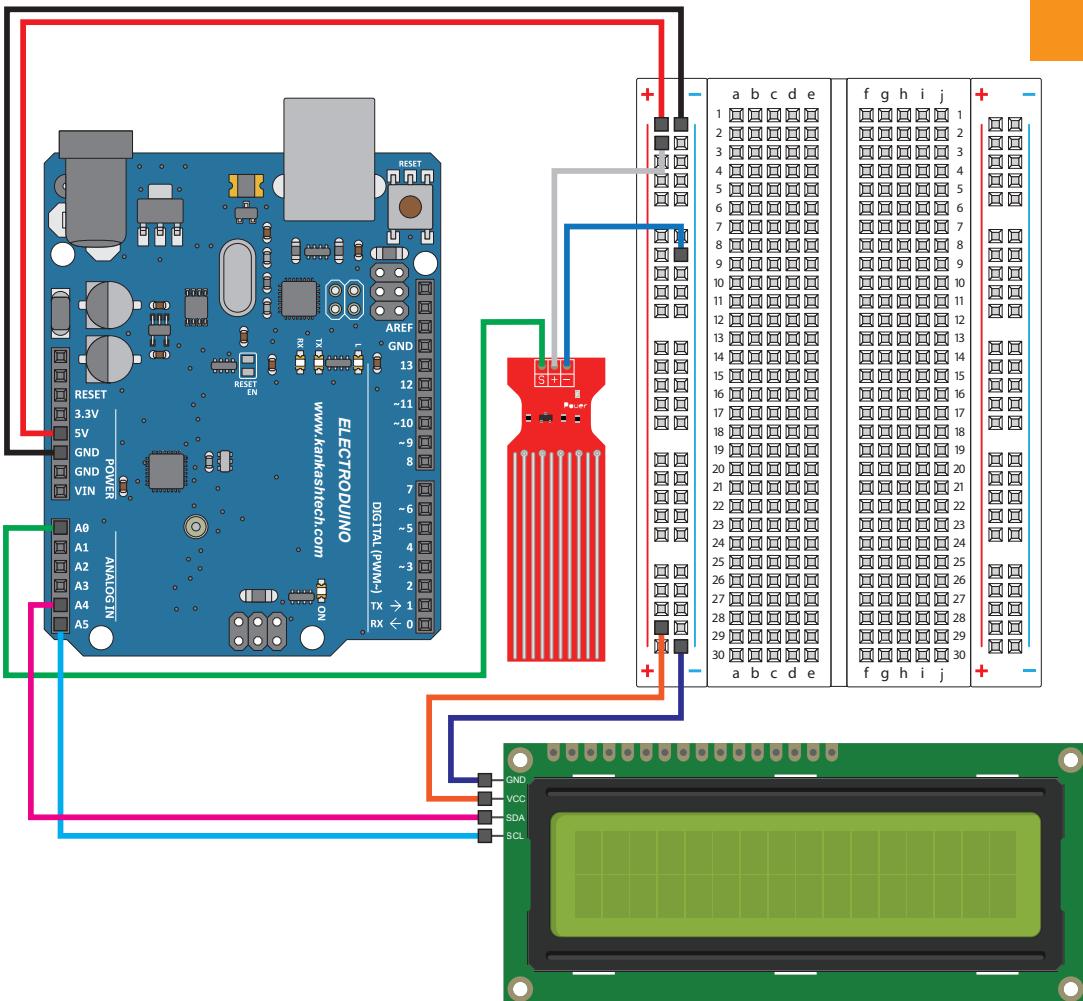


LCD1602-I2C
(۱ عدد)



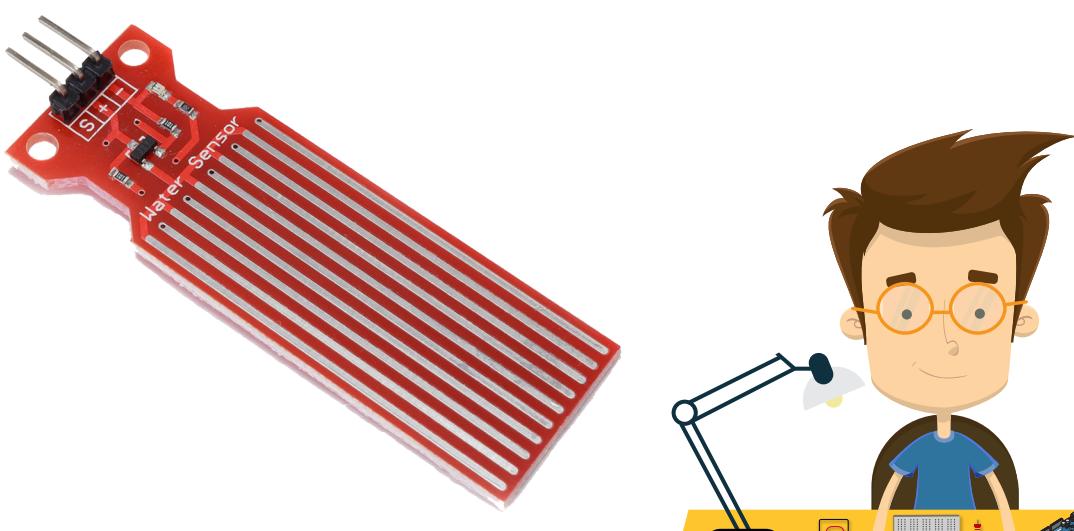
سنسور سطح آب

کنکا



توضیح مدار:

سنسور سطح آب، سه پایه مثبت (VCC)، منفی (GND) و سیگنال دارد. پایه های مثبت و منفی به VCC و GND آردوینو و پایه سیگنال به پین آنالوگ A0 متصل می شود. پین های این سنسور به شکل زیر است:



سنسور سطح آب



نحوه عملکرد سنسور سطح آب مشابه سنسورهای استفاده شده در پروژه های پیشین است. بدین صورت که در اثر تغییر طول بخشی از سنسور که در آب قرار دارد، مقاومت آن تغییر کرده، باعث تغییر ولتاژ خوانده شده در پین آنالوگ می شود.

با مشاهده مقدار سنسور در وضعیت های خالی (empty) و پر (full) می توان میزان پر بون وضعیت مخزن را مشاهده کرد. باید دقیق داشت که در صورتی که سنسور از آب بیرون کشیده شود، به علت خیس بودن، مقدار صحیحی نشان داده نمی شود. لذا باید منتظر ماند تا آب روی سطح سنسور خشک شود.

ابتدا پین سنسور تعریف می شود.

```
int sensorPin= A0;
```

در تابع `setup`، پین مربوط به سنسور به عنوان ورودی تعریف می شود:
`pinMode(sensorPin, INPUT);`

در تابع `loop`، نحوه محاسبه عمق آب با استفاده از مقدار خوانده شده توسط سنسور تعریف می شود:

```
int sensorValue = analogRead(sensorPin);
```

سایر مراحل مربوط تعریف LCD و نمایش مقادیر بر روی آن، مشابه پروژه ۱۲ است.



File > Examples > Electroduino > Circuit _13

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up

void setup() {
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor(1,0);
    myLCD.print("KankashTech.com");
    delay(3000);
    for (int cur=0; cur<16; cur++){
        myLCD.scrollDisplayLeft();
        delay(500);
    }
}

void loop() {
    int sensorValue = analogRead(sensorPin);
    myLCD.clear();
    myLCD.setCursor(2, 0);
    myLCD.print("Water Value: ");
    lcd.setCursor(6, 1);
    lcd.print(sensorValue);
    delay(200);
}
```



سنسور PIR یا Passive IR برای تشخیص حرکت در محیط استفاده می‌شود. این سنسورها، کوچک، ارزان قیمت، دارای مصرف و همچنین استهلاک کم هستند و گزینه بسیار مناسبی برای کاربردهای خانگی و محیط‌های کاری هستند.

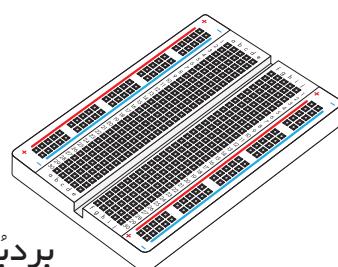
در صورت تشخیص حرکت، پایه سیگنال سنسور در وضعیت HIGH قرار می‌گیرد. حساسیت و مدت زمان HIGH بودن از زمان تشخیص حرکت، به وسیله دو پتانسیومتر تعییه شده بر روی سنسور قابل تنظیم است.

در این پروژه با تشخیص حرکت در محیط، یک LED روشن شده و عبارت "Motion Detected" بر روی LCD نمایش داده می‌شود.

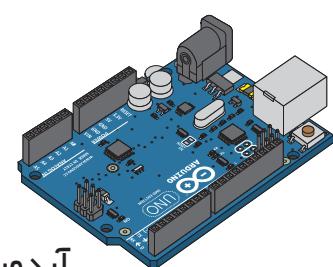
مطلوبات



سیم جامپر نری-نری
(۳ عدد)



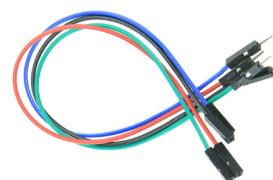
بردبرد



آردوبینو



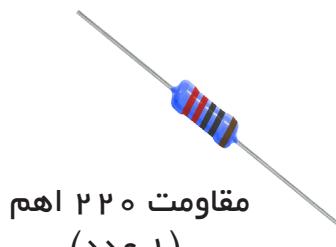
سنسور تشخیص حرکت PIR
(۱ عدد)



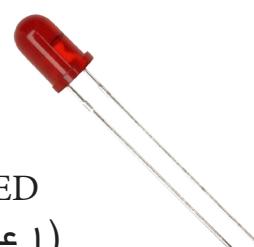
سیم جامپر نری - مادگی
(۷ عدد)



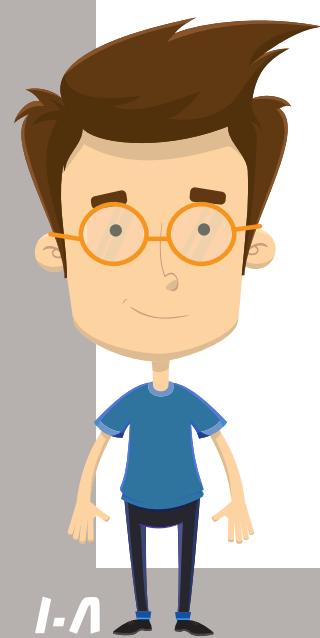
LCD1602-I2C
(۱ عدد)



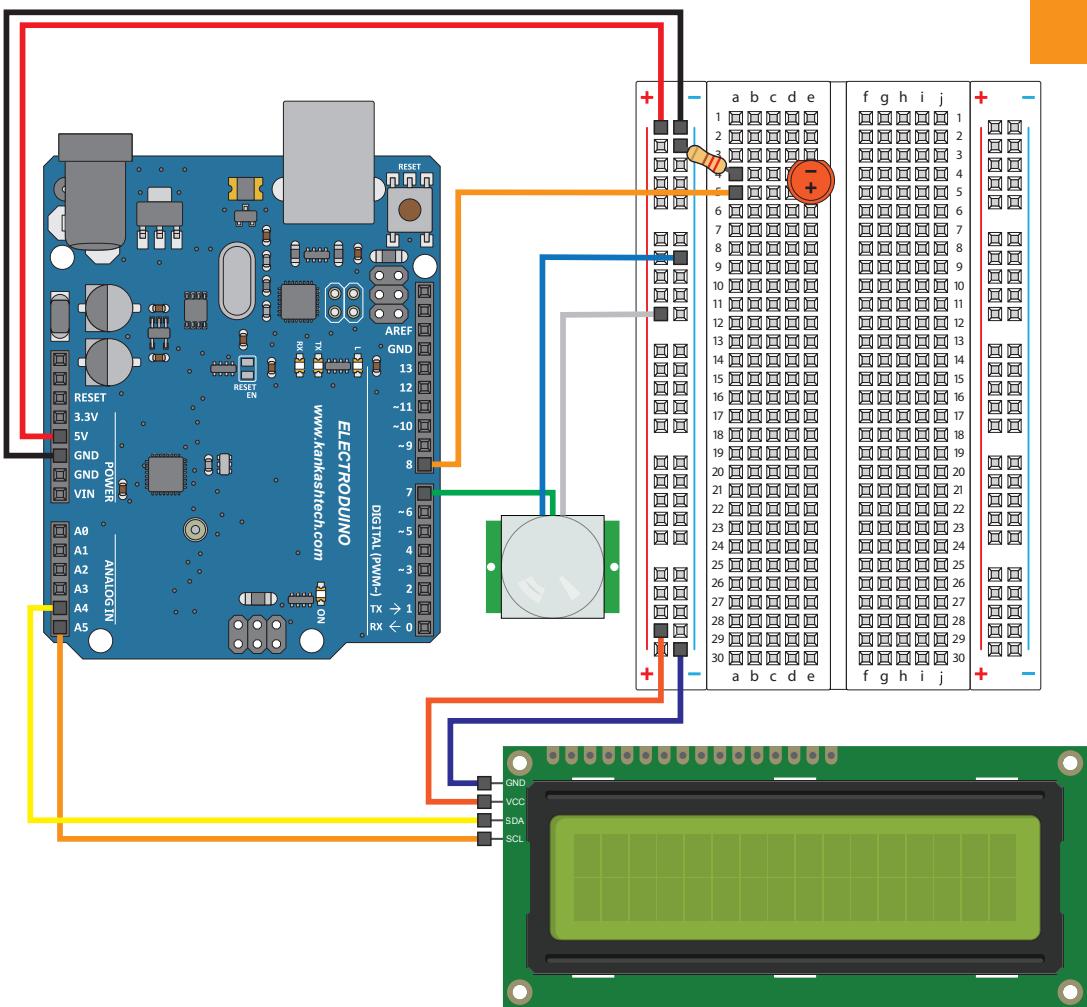
مقاومت ۲۲۰ مهـ (۱ عدد)



LED
(۱ عدد)



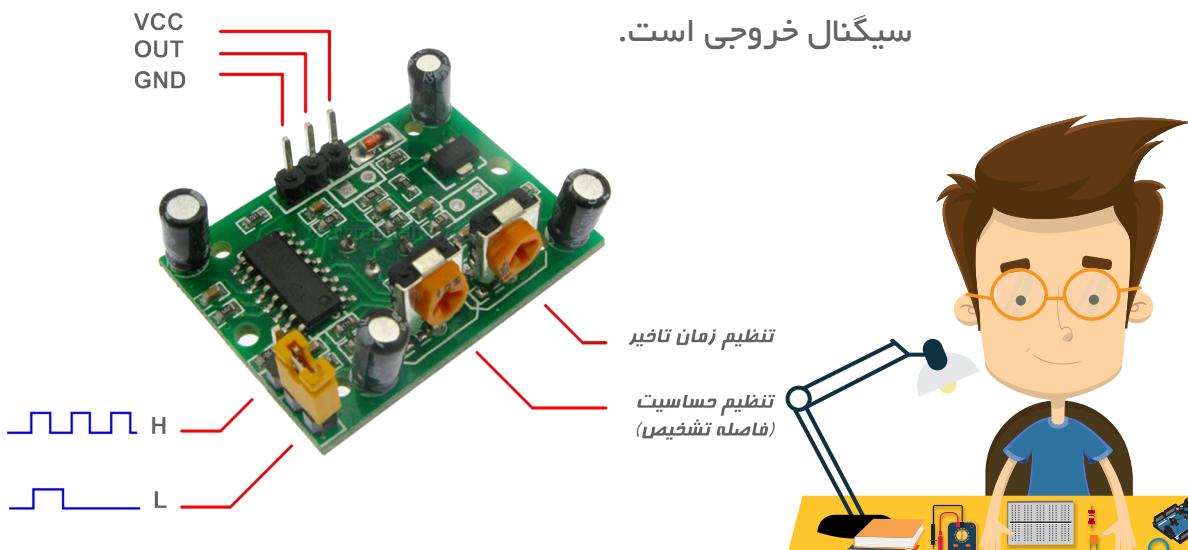
سنسر تشخیص حرکت (PIR)



توضیح مدار:

سنسور PIR سه پایه GND، VCC و سیگنال دارد. پایه های VCC و GND به پین های متناظر بر روی برد آردوینو و پایه سیگنال به پین ۷ متصل می شود. در صورت تشخیص حرکت، پین ۸ در وضعیت HIGH قرار خواهد گرفت. سنسور PIR دو پیچ تنظیم (که در حقیقت دو پتانسیومتر هستند) دارد. یک از این پیچ ها برای تنظیم حساسیت سنسور و پیچ دیگر برای تنظیم مدت زمان HIGH بودن

سیگنال خروجی است.

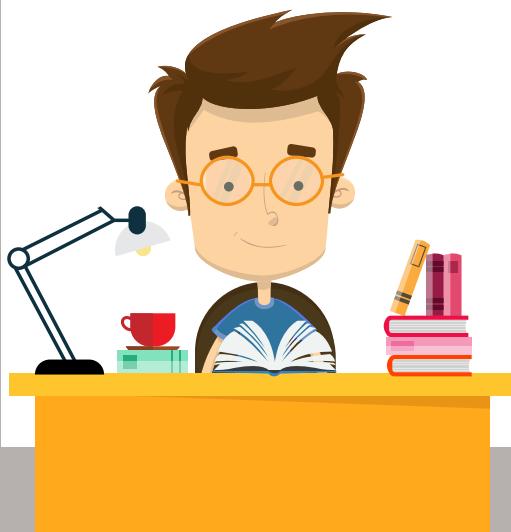
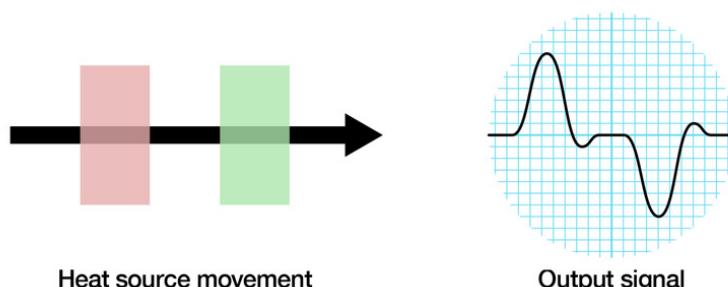
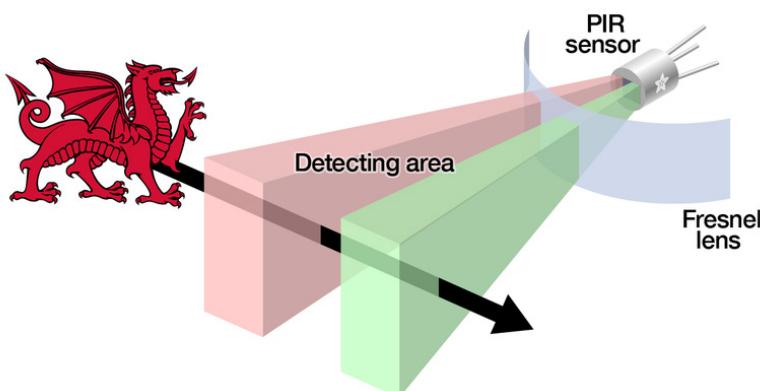


ساختار سنسورهای تشخیص حرکت PIR پیچیده تر از سایر سنسورهای استفاده شده در پروژه های قبل است زیرا متغیرهای متعددی بر ورودی و خروجی سنسور تاثیر می گذارند.

سنسور PIR دو شیار دارد که هر کدام از ماده ای حساس به نور مادون قرمز ساخته شده اند و فضای دید مشخصی دارند. هنگامی که جسمی در محیط حرکت نمی کند، هر دو شیار میزان مادون قرمز یکسانی دریافت می کنند که مقدار ساطع شده از اتاق، دیوارها یا محیط است.

هنگامی که یک جسم گرم نظیر انسان یا حیوان از ناحیه تشخیص سنسور عبور می کند، ابتدا از مقابل نیمی از سنسور (شیار اول) رد شده و باعث ایجاد تغییر جزئی ثابت میان دو نیمه می شود. هنگامی جسم در حال خروج از مقابل سنسور است، برعکس این اتفاق می افتد و یک تغییر جزئی منفی در میزان تشعشع اندازه گیری شده دو شیار ایجاد می شود.

سنسور مادون قرمز در یک پوسته فلزی قرار گرفته که در برابر دما، رطوبت و نویز از آن محافظت کند. این پوسته یک دریچه دارد که از ماده ای با قابلیت عبور مادون قرمز ساخته می شود.



سنسور تشخیص حرکت (PIR)

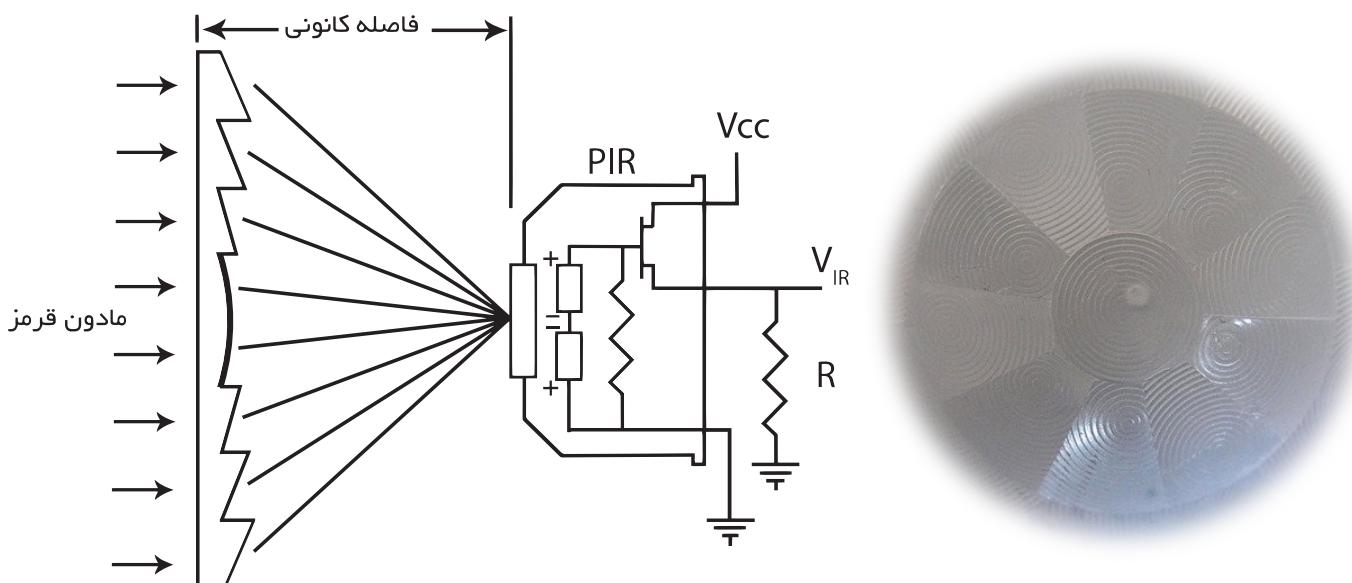


سنسورهای PIR عموماً مشابه بوده و فقط در حساسیت و قیمت متفاوت هستند. تفاوت عملکردی عمدتاً از پوسته اپتیکی ناشی می‌شود.

در شکل بالا، تنها دو ناحیه مستطیلی به عنوان ناحیه تشخیص سنسور دیده می‌شود؛ در حالی که عموماً هدف این است که یک فضای بزرگ توسط سنسور پوشش داده شود. برای این کار، با استفاده از یک لنز، فضای محیط بر روی ناحیه دید سنسور مرکز می‌شود. برای این کار از عدسی فرنل (fresnel) استفاده می‌شود.

عدسی فرنل از تعدادی شیار متحدم مرکز با هندسه خاص تشکیل شده است. این عدسی‌ها به علت ضخامت نازک ولی قدرت تمرکز بالا، استفاده زیادی پیدا کرده‌اند.

برای اینکه فضای بیشتری توسط این سنسور‌ها پوشش داده شود، یک پوسته پلاستیکی کروی بر روی سنسور قرار داده می‌شود که بخش‌های تشکیل دهنده این پوسته، هر کدام یک عدسی فرنل هستند.



عملکرد سنسور

همانگونه که در بخش توضیح مدار گفته شد، در صورت تشخیص حرکت در محیط، پین وسط سنسور (OUT) در وضعیت HIGH قرار می‌گیرد. به دو طریق می‌توان از این خروجی استفاده کرد:

1. خروجی سنسور مستقیماً به یک قطعه مانند LED یا رله یا قطعه مشابه متصل شود. بدین صورت در صورت تشخیص حرکت در محیط، LED روشن شده یا رله فعال شود.

2. خروجی سنسور به یکی از پین‌های آردوینو متصل شده و وضعیت این پین مانیتور شود. در صورت تشخیص حرکت در محیط، پین موردنظر در وضعیت HIGH قرار می‌گیرد. در صورتی که آردوینو تشخیص دهد که پین در وضعیت HIGH قرار گرفته است (مشابه پروژه دکمه فشاری)، دستورات مشخص شده اجرا می‌شود.

در این پروژه روشن شدن LED و نمایش عبارت "Motion Detected" بر روی LCD از این روش انجام می‌شود.

عملکرد جامپر

همانگونه که در بخش توضیح مدار نشان داده شده، در گوشه پایین سنسور، یک جامپر وجود دارد که می‌توان آنرا در دو وضعیت H و L قرار داد. این وضعیت‌ها در کنار جامپر با حروف L و H مشخص شده است.

اگر جامپر در وضعیت L باشد، به مدت زمان مشخص تنظیم شده توسط پتانسیومتر روی سنسور، پین خروجی در وضعیت HIGH قرار می‌گیرد.

اگر جامپر در وضعیت H قرار داشته باشد، به مدت زمان مشخص تنظیم شده توسط پتانسیومتر، پایه خروجی در وضعیت تناوبی HIGH/LOW قرار می‌گیرد و پس از پایان دوره، در وضعیت LOW باقی می‌ماند.



سنسور تشخیص حرکت (PIR)



File > Examples > Electroduino > Circuit_14

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up
int pirPin= 7;
int ledPin= 8;

void setup() {
    pinMode(pirPin, INPUT);
    pinMode(ledPin, OUTPUT);
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor (1, 0);
}
void loop() {
    int pirState = digitalRead(pirPin);
    if (pirState==LOW) {
        myLCD.clear();
        myLCD.setCursor(4 , 0);
        myLCD.print("No Motion");
        digitalWrite(ledPin, LOW);
    }
    else {
        myLCD.clear();
        myLCD.setCursor(0 , 0);
        myLCD.print("Motion Detected");
        digitalWrite(ledPin, HIGH);
    }
    delay(200);
}
```





سنسور شدت نور

برخی از سنسورها تنها دو پایه دارند. این سنسورها در واقع، مقاومت هایی هستند که در اثر قرارگیری در وضعیت های مختلف، مقاومت آنها تغییر می کند.

سنسور شدت نور یکی از این سنسورهاست. در اثر افزایش شدت نور، مقاومت الکتریکی این سنسور کاهش می یابد.

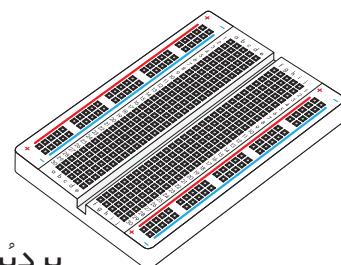
با توجه به اینکه آردوینو نمی تواند مقاومت را مستقیماً اندازه گیری کند، باید به طریقی تغییرات در مقاومت سنسور را به تغییرات ولتاژ تبدیل کرد.

یکی از رایج ترین روش ها برای این کار، استفاده از تقسیم ولتاژ است که در این پروژه از آن استفاده شده است.

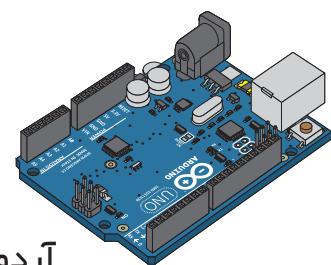
مطلوبات



سیم جامپر نری-نری
(۵ عدد)



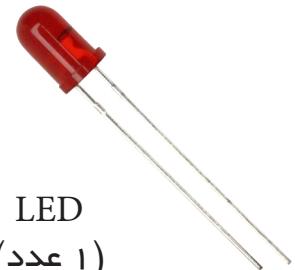
برد برد



آردوینو



سنسور نور
(۱ عدد)



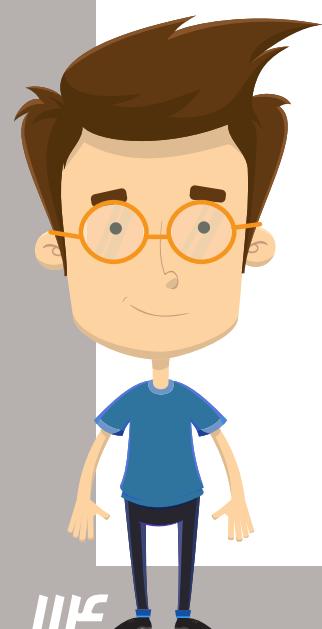
LED
(۱ عدد)

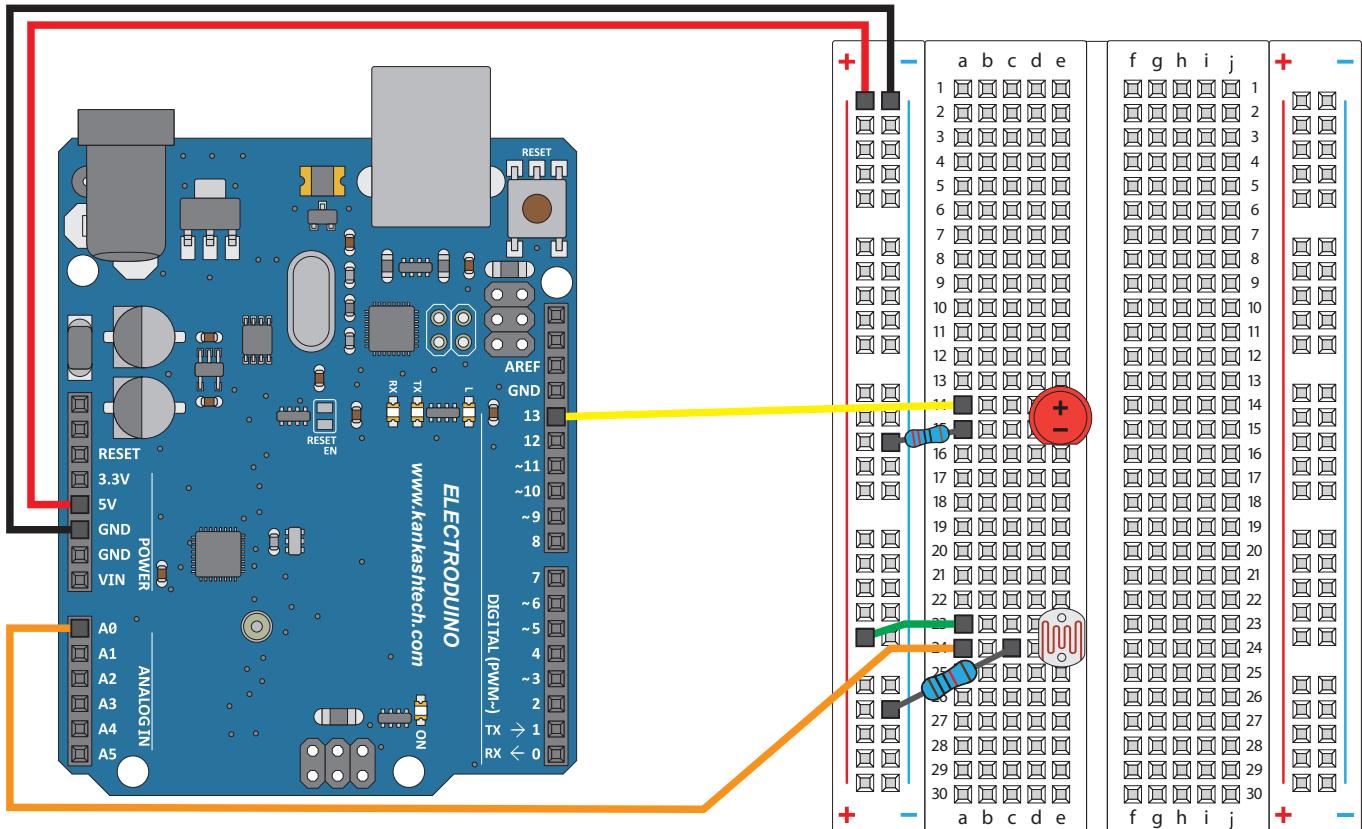


مقاومت ۱۰ کیلو اهم
(۱ عدد)



مقاومت ۲۲۰ اهم
(۱ عدد)





توضیح مدار:

مطابق با شماتیک بخش مقدمه، یک پایه سنسور به پین ۵ ولت آردوینو و پایه دیگر آن به پین آنالوگ A0 متصل می‌شود. یک مقاومت ۱۰ کیلو اهم بین منفی (GND) و پین آنالوگ A0 قرار می‌گیرد تا مدار تقسیم ولتاژ تشکیل شود.

یک LED به همراه یک مقاومت ۲۰۰ اهم به پین ۱۳ متصل شده تا در صورت کاهش نور از حد مشخص،

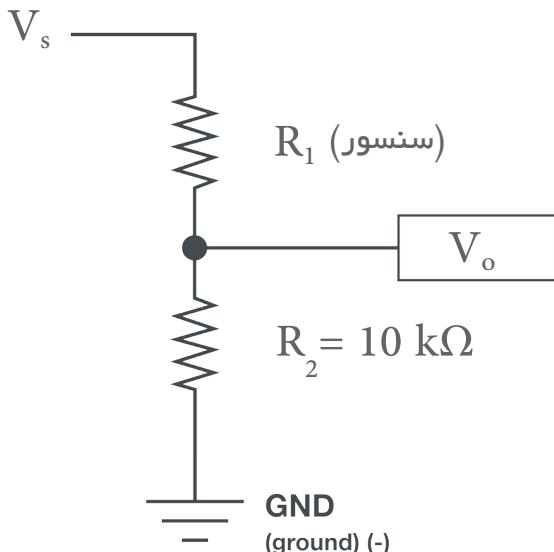
این LED به صورت خودکار روشن شود.



سنسور شدت نور

IQ

در این پروژه از اصل تقسیم ولتاژ، برای اندازه گیری مقدار ولتاژ ناشی از تغییرات مقاومت سنسور استفاده می شود. مقاومت بالا (R_1)، سنسور و مقاومت پایین (R_2) یک مقاومت ثابت (در این پروژه ۱۰ کیلواهم) است. (دقت داشته باشید که در نقطه وسط، ولتاژی اعمال نمی شود و فقط ولتاژ در این نقطه اندازه گیری می شود)



با توجه به اینکه دو مقاومت به صورت سری بسته شده اند، رابطه میان جریان، ولتاژ و مقاومت ها به شرح زیر است:

$$V_s = I \times (R_1 + R_2) \Rightarrow I = \frac{V_s}{R_1 + R_2}$$

با توجه به اینکه ا در کل مسیر ثابت است، اگر مقاومت پایین به صورت تنها در نظر گرفته شود، اختلاف ولتاژ دو سر آن عبارت است از:

$$V_o = I \times R_2$$

با جایگزینی از رابطه اول در رابطه دوم:

$$V_o = V_s \times \frac{R_2}{R_1 + R_2}$$

همانگونه که مشاهده می شود، مقاومت R_1 که سنسور است، در مخرج کسر قرار گرفته است. با این چیدمان، در صورت افزایش مقاومت (تاریک شدن)، ولتاژ نقطه وسط کاهش و در صورت کاهش مقاومت (افزایش شدت نور)، ولتاژ نقطه وسط افزایش می یابد.

در این پروژه در صورت کمتر شدن مقدار ولتاژ از حد مشخص (تاریک شدن هوا)، یک LED روشن می شود.



سنسر شدت نور

IQ

File > Examples > Electroduino > Circuit_15

```
int sensorPin=A0;
int ledPin= 13;

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

void loop() {
    digitalWrite (ledPin, LOW);
    int sensorValue=analogRead(sensorPin);
    if (sensorValue > 400) {
        digitalWrite (ledPin, LOW);
    }
    else {
        digitalWrite(ledPin, HIGH);
    }

    Serial.println(sensorValue);
    delay(200);
}
```



سنسور تشخیص شعله آتش

۱۷

روش های مختلفی برای تشخیص آتش استفاده می شود. سنسوری که در این پروژه برای این مقصود استفاده می شود، از نوع مادون قرمز دور، امواج حرارتی هستند که طول موج هایی میان ۷۰۰ تا ۱۰۰۰ نانومتر دارند.

سنسور YG1006 که در این پروژه استفاده می شود، یک ترانزیستور نوری NPN با سرعت و حساسیت بالاست. به علت پوشش اپوکسی مشکی رنگ خود، در برابر امواج مادون قرمز حساسیت دارد. در این پروژه، از ماژول LM393 برای خواندن سنسور استفاده می شود. این ماژول یک مقایسه گر (Comparator) است که یک خروجی انalog و یک خروجی دیجیتال دارد. نحوه عملکرد این ماژول در ادامه توضیح داده شده است

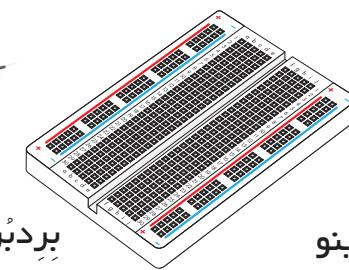
مطلوبات



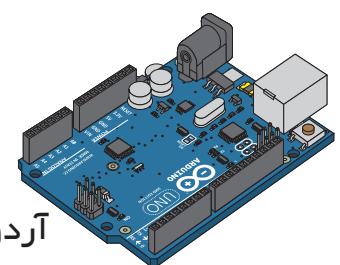
LCD1602
(۱ عدد)



سیم جامپر نری-نری
(۳ عدد)



بردبرد



آردوینو



LM393
(۱ عدد)



سیم جامپر نری-مادگی
(۲ عدد)



سیم جامپر نری-مادگی
(۸ عدد)



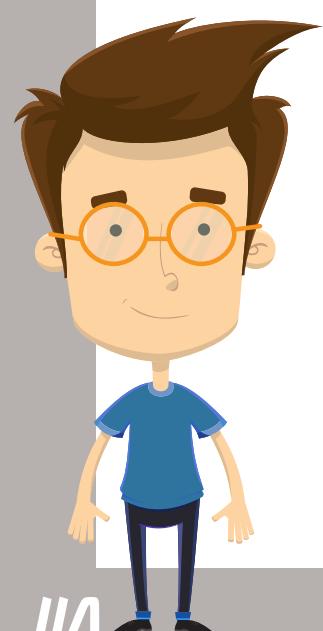
مقاومت ۲۲۰ اهم
(۱ عدد)



LED
(۱ عدد)

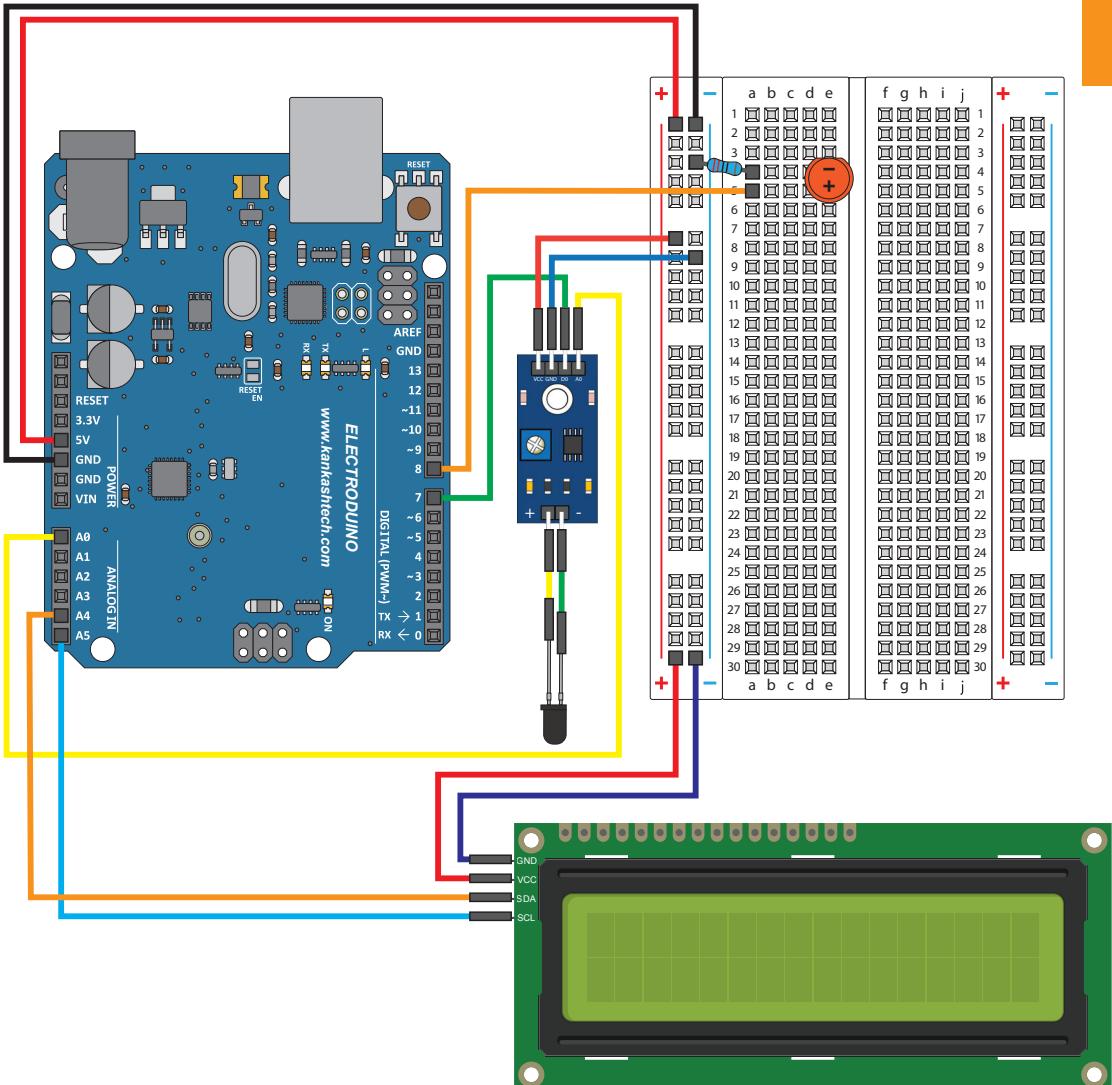


سنسور تشخیص شعله
(۱ عدد)



سنسور تشخیص شعله آتش

۱۷



توضیح مدار:

ماژول LM393، دو پایه برای اتصال به سنسور و ۴ پایه برای اتصال به آردوینو دارد. از این ۴ پایه، دو پایه، GND و VCC هستند که باید به پین های متناظر برد آردوینو متصل شوند. یکی از پایه های دیگر، A0 است که باید به یکی از پین های آنالوگ (در اینجا A0) متصل شود. پایه دیگر، D0 است که به پین ۷ متصل می شود.

از پین ۸ یک خروجی به LED رفته که در صورت تشخیص شعله آن LED روشن شود.

جهت اتصال سنسور تشخیص شعله مهم است. پایه کوتاه سنسور به پایه منفی ماژول LM393 و پایه بلند آن به پایه مثبت ماژول متصل می شود.

برای اتصال سنسور به LM393 یک جفت سیم دو سرمهادگی در کیت قرار داده شده است.



سنسور تشخیص شعله آتش

۱۷

یک آی سی مقایسه گر است. وظیفه این مژول، مقایسه ولتاژ ورودی با یک مقدار از پیش تعیین شده (threshold) و تصمیم گیری است. مژول استفاده شده در این پروژه، در صورت افزایش ولتاژ بیش از حد تعیین شده، پایه D0 را در وضعیت HIGH قرار می دهد. در نتیجه می توان با مانیتور کردن وضعیت پین آنالوگ یا دیجیتال مژول، تصمیم گیری کرد و یا به صورت مستقیم با استفاده از وضعیت HIGH پایه دیجیتال، قطعه یا قطعاتی را راه اندازی کرد.

این قابلیت برای برخی از سنسورها بسیار مفید است. به عنوان مثال، در صورتی که شعله آتش در محیط تشخیص داده شد، یک آذیر به صدا در آید. برای اینکار، تنها نیاز به تامین ولتاژ مورد نیاز عملکرد سنسور و آذیر است و می توان بدون استفاده از آردوینو، این کار را انجام داد.

همانگونه که پیشتر گفته شد، با استفاده از پتانسیومتری که بر روی مژول تعییه شده است می توان وضعیتی که پایه دیجیتال در وضعیت HIGH قرار می گیرد را تنظیم کرد.

در این پروژه، مشابه پروژه سنسور تشخیص حرکت، در صورت مشاهده شعله، عبارت "بر روی LCD نمایش داده می شود. همچنین یک LED روشن می شود.



File > Examples > Electroduino > Circuit_16

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up

int sensorPin= A0;
int digitalPin = 7;
int ledPin = 8;

void setup() {
    pinMode(digitalPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor(1, 0);
    myLCD.print ("kankashtech.com");
    delay(3000);
}

void loop() {
    int sensorValue = analogRead(sensorPin);
    int digitalState = digitalRead(digitalPin);
    myLCD.clear();
    myLCD.setCursor (0, 0);
    myLCD.print ("A: ");
    myLCD.setCursor (3, 0);
    myLCD.print (sensorValue);
```



File > Examples > Electroduino > Circuit_16

```
myLCD.setCursor(8,0);
myLCD.print ("D: ");
myLCD.setCursor (11,0);
myLCD.print (digitalState);

if (digitalState==0) {
    digitalWrite (ledPin , LOW);
} else {
    digitalWrite (ledPin, HIGH);
}

myLCD.setCursor(2 , 1);

if (sensorValue > 500) {
    myLCD.print("No Flame");
}
else {
    myLCD.print("Flame Detected");
}
delay(200);

}
```

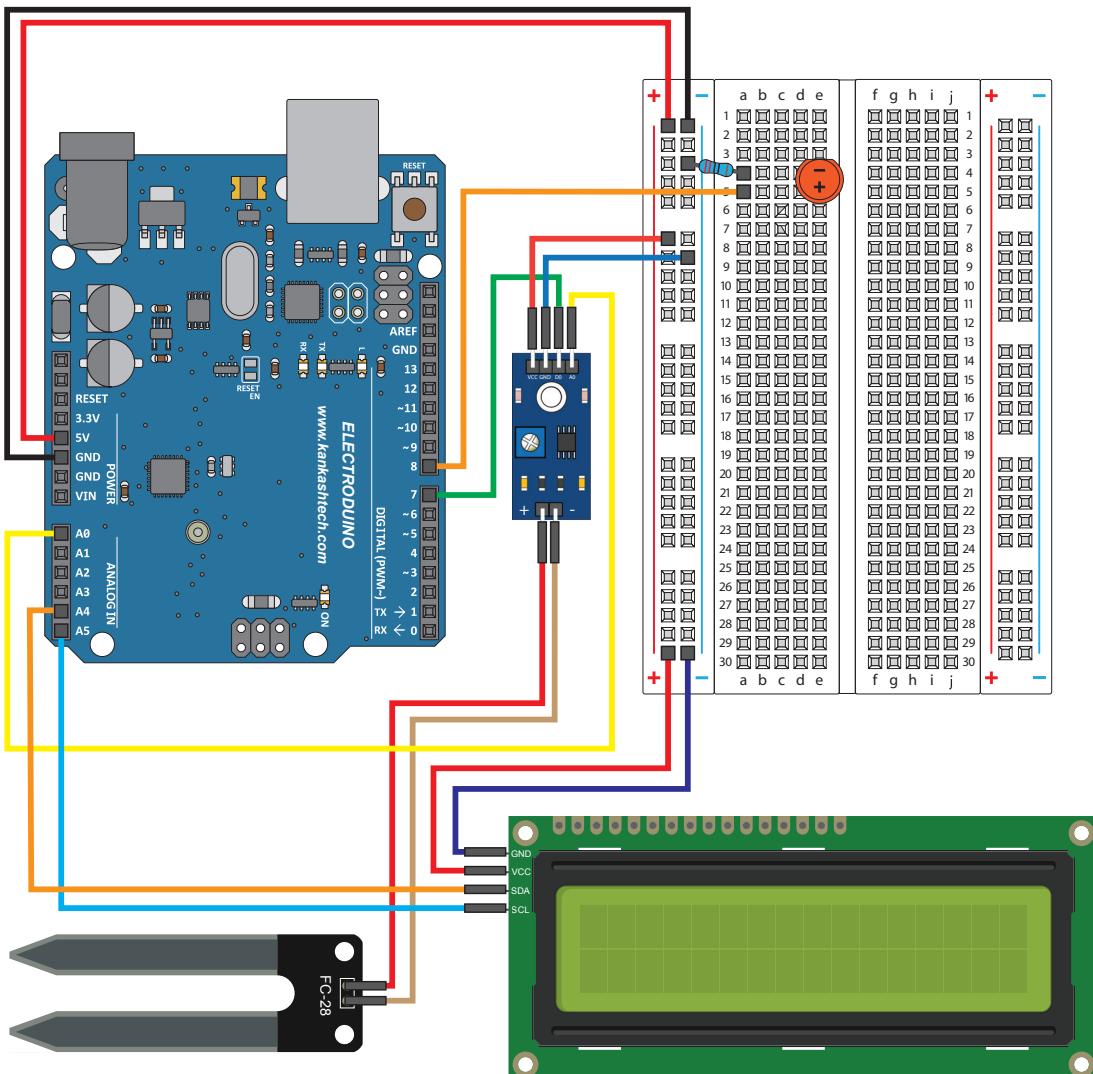


سنسور سنجش رطوبت خاک

سنسور رطوبت خاک از دو الکترود جدا از هم تشکیل شده است که در خاک فرو می‌رود و جریان را از خاک عبور می‌دهد. هرچه رطوبت خاک بیشتر باشد، مقاومت خاک کمتر بوده و جریان بیشتری از خاک عبور می‌کند و مژول عدد کمتری نشان خواهد داد. با استفاده از این سنسور و ترکیب آن با قطعات دیگر، می‌توان سیستم آبیاری هوشمند برای باغچه و گلستان درست کرد. در این پروژه، از مژول LM393 برای خواندن این نوع سنسور استفاده می‌شود.

مطلوبات





توضیح مدار:

مدار و کد این پروژه مشابه پروژه ۱۶ است، با این تفاوت که جهت اتصال سنسور به مازول LM393 مهم نیست و پین های سنسور تفاوت عملکردی با یکدیگر ندارند.



File > Examples > Electroduino > Circuit_17

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up

int sensorPin= A0;
int digitalPin = 7;
int ledPin = 8;

void setup() {
    pinMode(sensorPin, INPUT);
    pinMode(digitalPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor(1, 0);
    myLCD.print ("kankashtech.com");
    delay(3000);
}

void loop() {
    int sensorValue = analogRead(sensorPin);
    int digitalState = digitalRead(digitalPin);
    myLCD.clear();
    myLCD.setCursor (0, 0);
    myLCD.print ("A: ");
    myLCD.print (sensorValue);
```



```
myLCD.setCursor(8,0);
myLCD.print ("D: ");
myLCD.setCursor (11,0);
myLCD.print (digitalState);

if (digitalState==0) {
    digitalWrite (ledPin , LOW);
} else {
    digitalWrite (ledPin, HIGH);
}

myLCD.setCursor(2 , 1);

if (sensorValue < 400) {
    myLCD.print("Soil is Wet");
}
else {
    myLCD.print("I am Thirsty!");
}
delay(200);
```



ماژول سنسور صدا

این ماژول دارای یک میکروفون خازن الکتروولیتی است. امواج صدا باعث ارتعاش لایه نازک الکتریت و در نتیجه تغییر ظرفیت خازن می‌شود. این سنسور مشابه سنسورهای پیشین است، با این تفاوت که سنسور استفاده شده در این مجموعه، به صورت آماده بر روی ماژول LM393 نصب شده است. (با توجه به اینکه در این ماژول از تقویت کننده استفاده نشده است، حساسیت آن پایین بوده و در برابر صدای ضعیف عمل نمی‌کند).

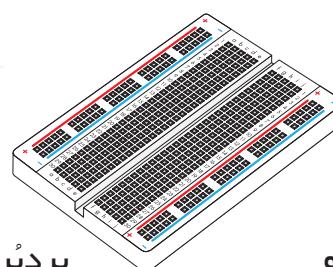
مطلوبات



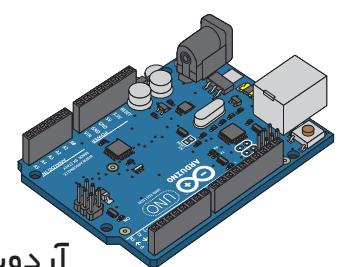
LCD1602
(۱ عدد)



سیم جامپر نری-نری
(۳ عدد)



بردبرد



آردوینو



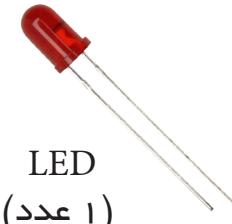
سنسور صدا
(۱ عدد)



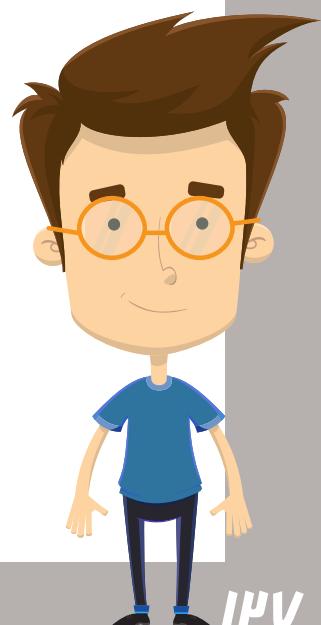
سیم جامپر نری-مادگی
(۸ عدد)



مقاومت ۲۲۰ اهم
(۱ عدد)

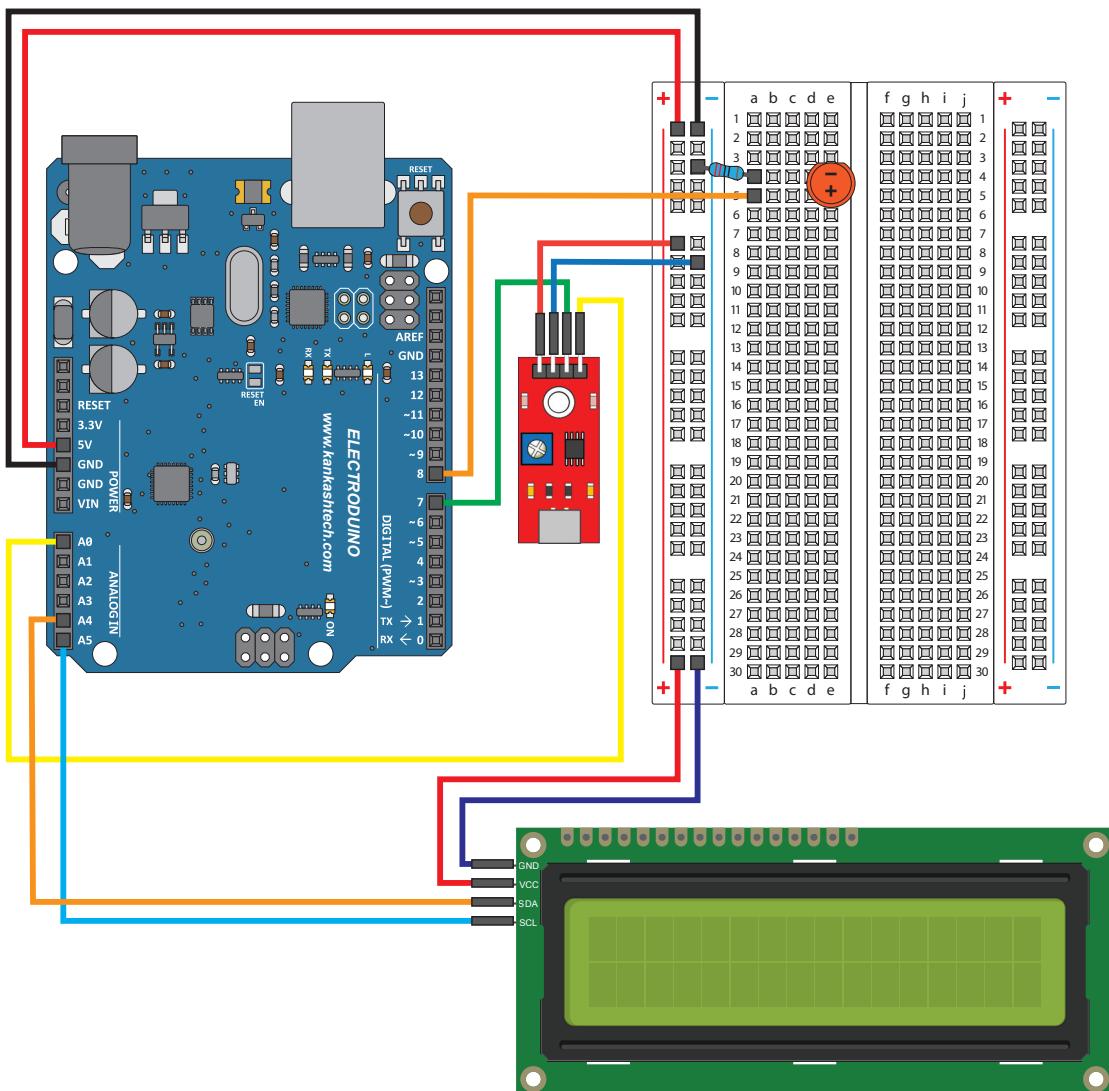


LED
(۱ عدد)



ماژول سنسور صدا

in



توضیح مدار:

مدار و کد این پروژه مشابه پروژه ۱۶ است.



File > Examples > Electroduino > Circuit_18

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up

int sensorPin= A0;
int digitalPin = 7;
int ledPin = 8;

void setup() {
    pinMode(sensorPin, INPUT);
    pinMode(digitalPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor(1, 0);
    myLCD.print ("kankashtech.com");
    delay(3000);
}

void loop() {
    int sensorValue = analogRead(sensorPin);
    int digitalState = digitalRead(digitalPin);
    myLCD.clear();
    myLCD.setCursor (0, 0);
    myLCD.print ("A: ");
    myLCD.print (sensorValue);
```





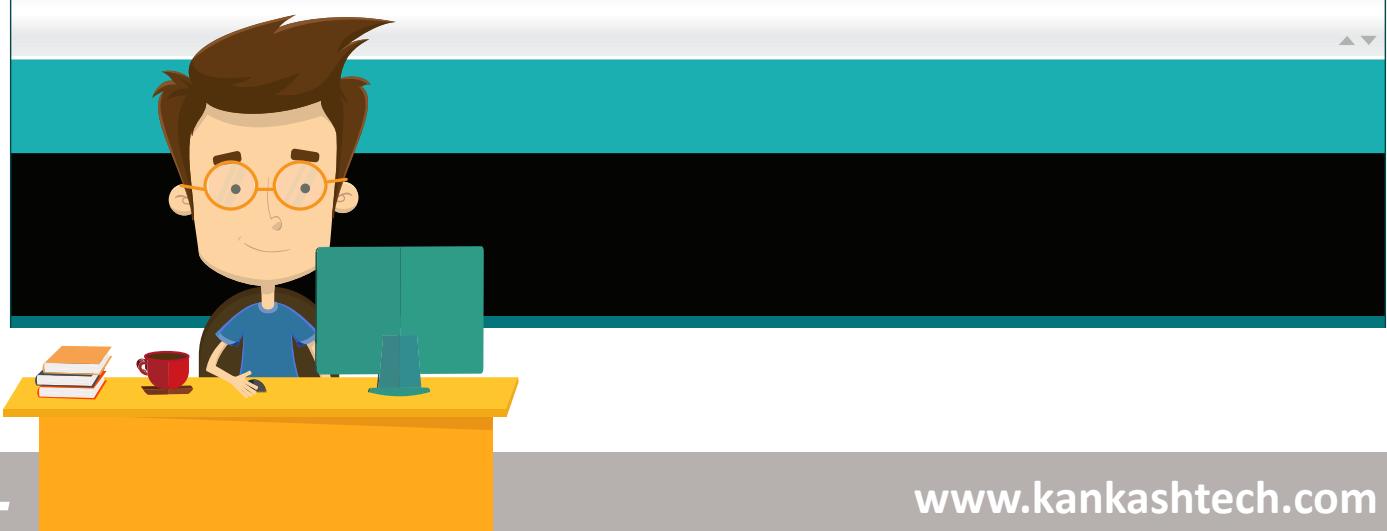
File > Examples > Electroduino > Circuit_18

```
myLCD.setCursor(8,0);
myLCD.print ("D: ");
myLCD.setCursor (11,0);
myLCD.print (digitalState);

if (digitalState==0) {
    digitalWrite (ledPin , LOW);
} else {
    digitalWrite (ledPin, HIGH);
}

myLCD.setCursor(2 , 1);

if (sensorValue < 100) {
    myLCD.print("No Sound");
}
else {
    myLCD.print("Sound Detected");
}
delay(200);
}
```



ماژول تشخیص رنگ اپتیکال

ماژول TCRT5000 یک ماژول تشخیص رنگ اپتیکال است که برای کاربری ربات های تعقیب خط طراحی شده است.

این ماژول از یک فرستنده مادون قرمز و یک ترانزیستور نوری درون یک پوسته مات پلاستیکی تشکیل شده است که بر روی یک ماژول LM393 (مشابه پروژه های قبل) نصب شده است. در صورتی که این ماژول در نزدیکی یک سطح کاملا سفید قرار گیرد عدد ۱۰۲۳ و در صورتی که در نزدیکی یک سطح کاملا سیاه قرار گیرد، عدد ۰ توسط پین آنالوگ خوانده می شود.

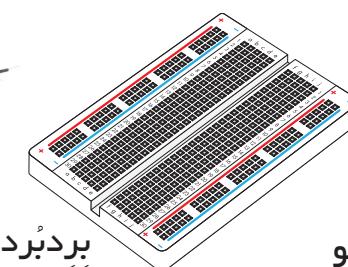
مطلوبات



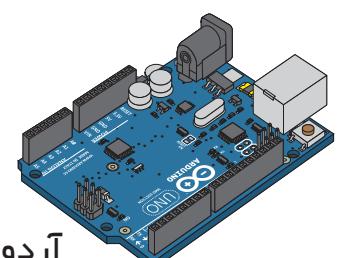
LCD1602
(۱ عدد)



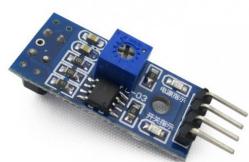
سیم جامپر نری-نری
(۳ عدد)



بردبرد



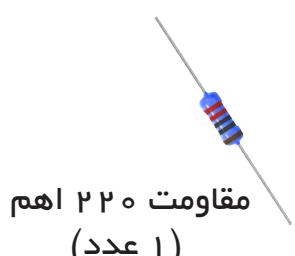
آردوینو



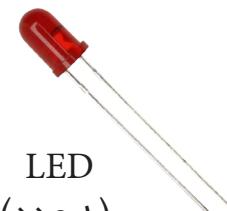
TCRT5000
(۱ عدد)



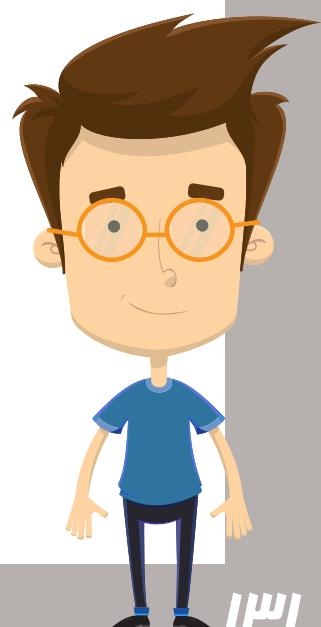
سیم جامپر نری-مادگی
(۸ عدد)

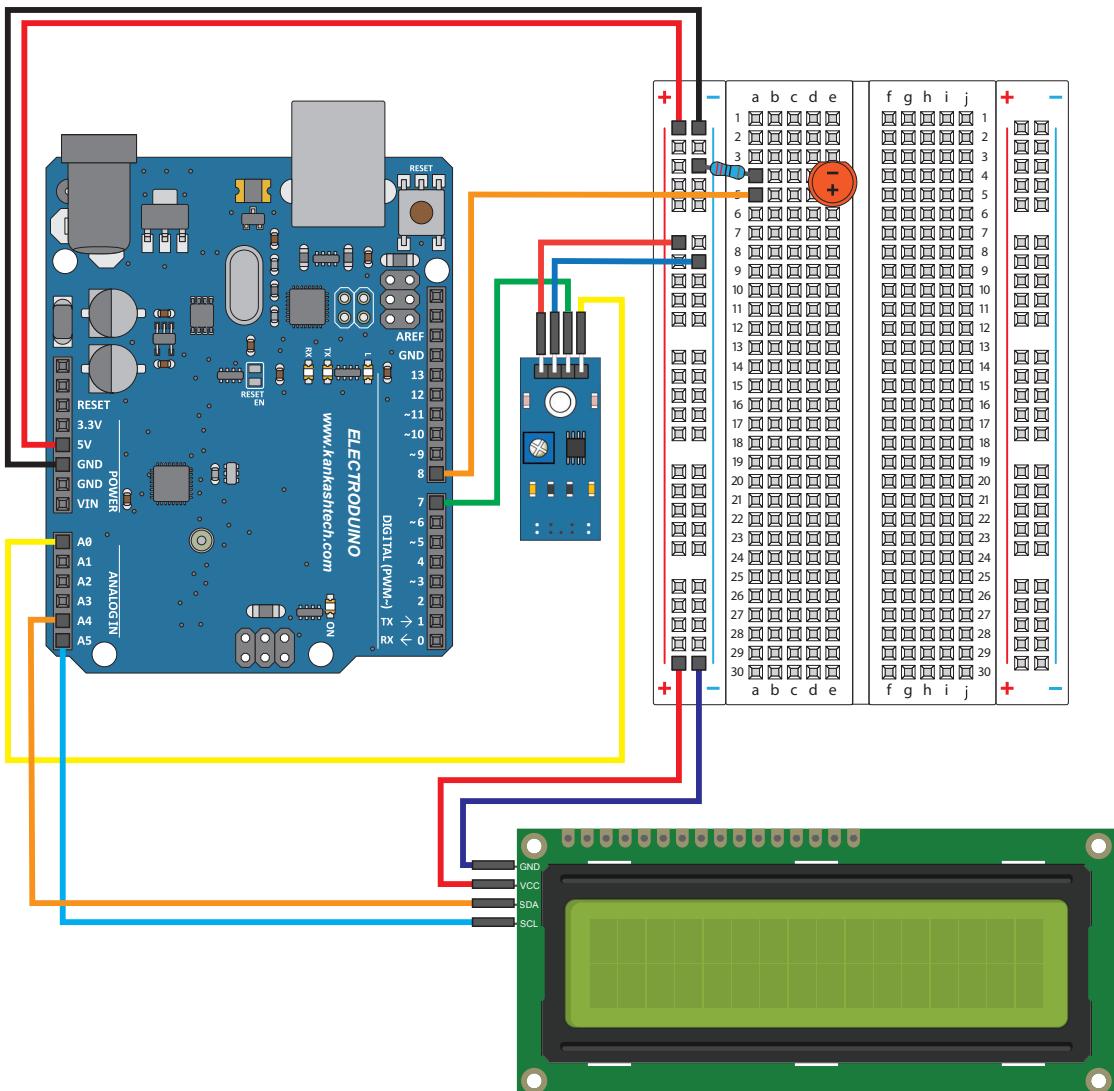


مقاومت ۲۲۰ اهم
(۱ عدد)



LED
(۱ عدد)





توضیح مدار:

کد و مدار این پروژه مشابه پروژه ۱۶ است.



File > Examples > Electroduino > Circuit_19

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C myLCD(0x3F, 16, 2); //replace 0x3F with 0x27 if nothing shows up

int sensorPin= A0;
int digitalPin = 7;
int ledPin = 8;

void setup() {
    pinMode(sensorPin, INPUT);
    pinMode(digitalPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    myLCD.init();
    myLCD.backlight();
    myLCD.setCursor(1, 0);
    myLCD.print ("kankashtech.com");
    delay(3000);
}

void loop() {
    int sensorValue = analogRead(sensorPin);
    int digitalState = digitalRead(digitalPin);
    myLCD.clear();
    myLCD.setCursor (0, 0);
    myLCD.print ("A: ");
    myLCD.print (sensorValue);
}
```



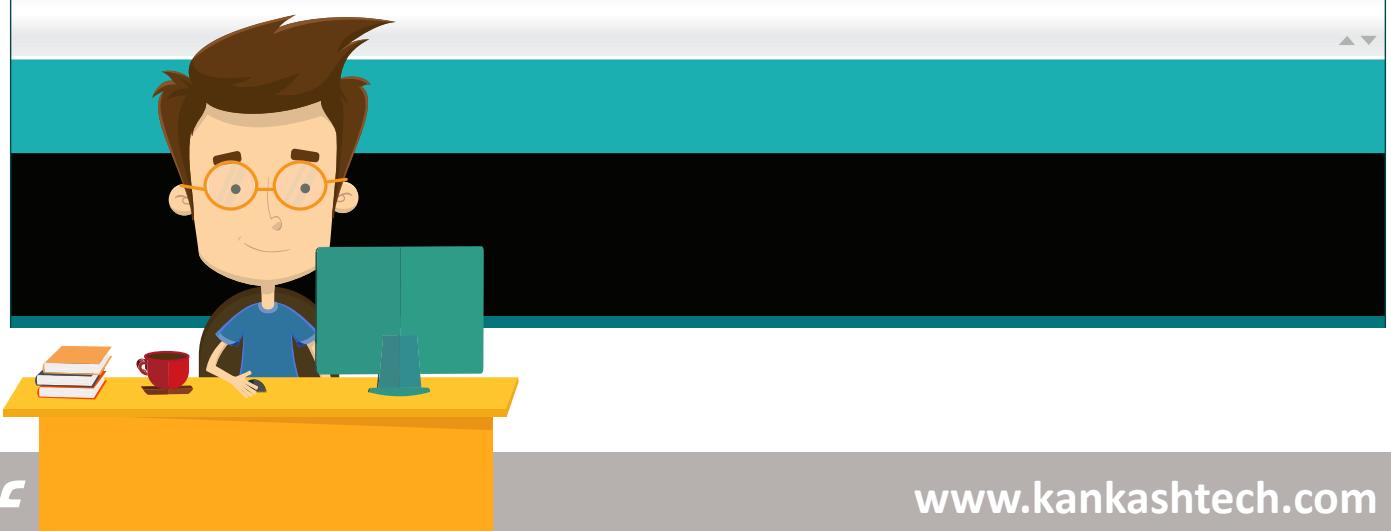
```
myLCD.setCursor(8,0);
myLCD.print ("D: ");
myLCD.setCursor (11,0);
myLCD.print (digitalState);

if (digitalState==0) {
    digitalWrite (ledPin , LOW);
} else {
    digitalWrite (ledPin, HIGH);
}

myLCD.setCursor(2 , 1);

if (sensorValue < 100) {
    myLCD.print("DARK");
}
else {
    myLCD.print("LIGHT");
}
delay(200);

}
```





ماژول Joystick

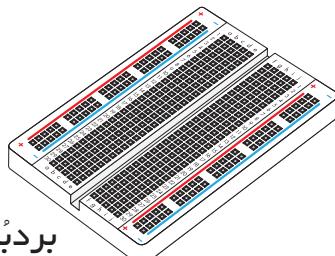
ماژول Joystick تلفیقی از دو پتانسیومتر و یک دکمه فشاری است. بسته به موقعیت قرارگیری اهرم، یک عدد میان ۰-۱۰۲۳ برای راستای افقی (x) و یک عدد میان ۰-۱۰۲۳ نیز برای راستی عمودی (y) گزارش می‌شود. همچنین با فشردن اهرم در راستای Z، پایه مربوط به دکمه فشاری در وضعیت LOW قرار می‌گیرد. در این پروژه از ماژول Joystick برای کنترل وضعیت خاموش و روشن ۴ عدد LED استفاده می‌شود.

مطلوبات

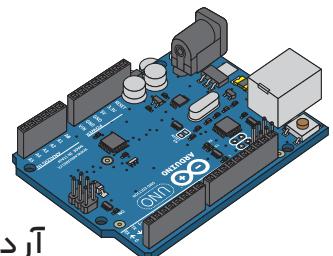
سیم جامپر نری-نری
(۷ عدد)



بردبرد



آردوبینو



Joystick
(۱ عدد)



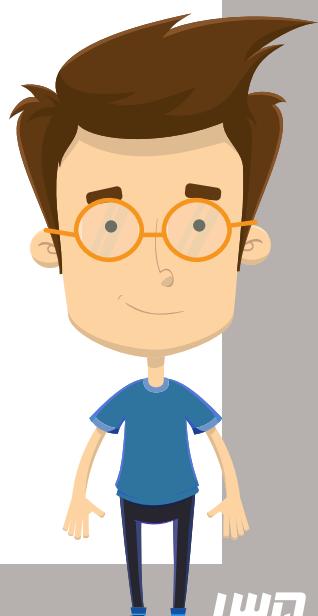
LED سبز
(۴ عدد)

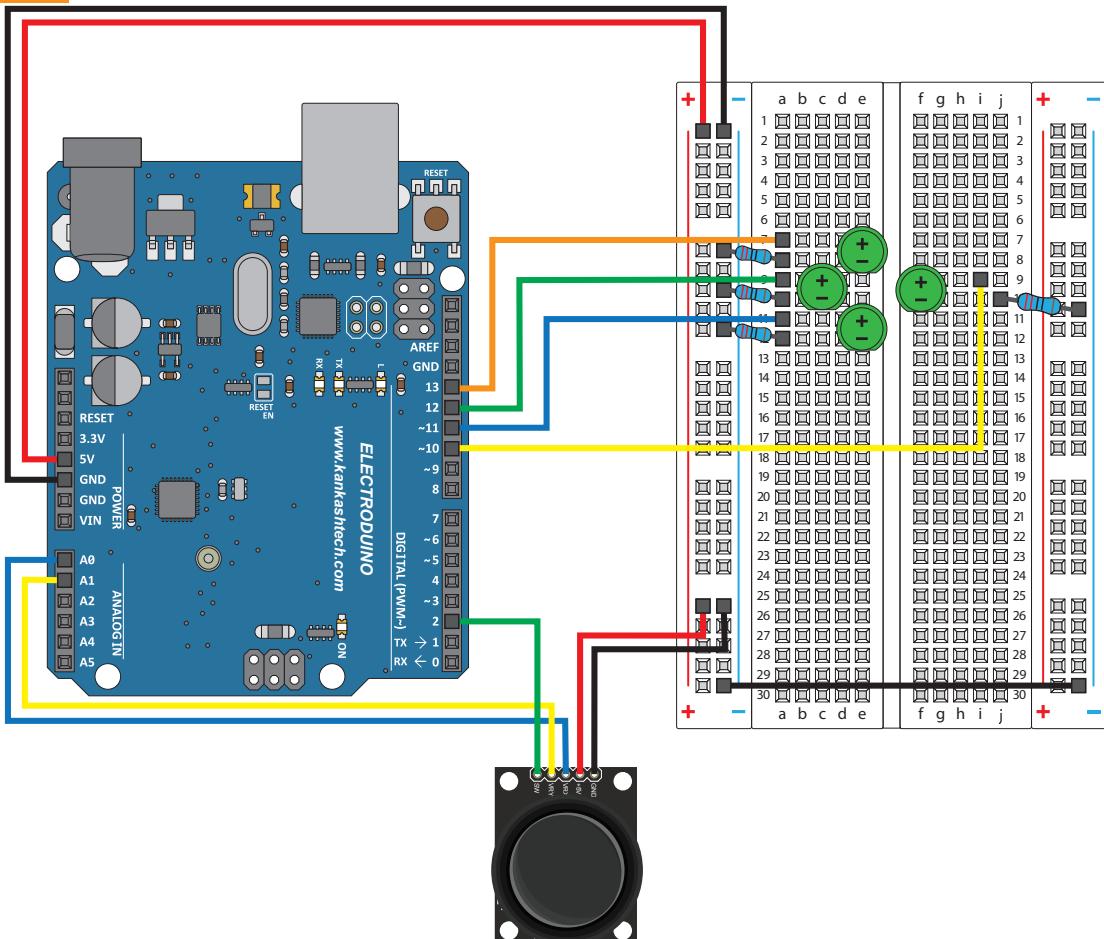


مقاومت ۲۰۰ اهم
(۴ عدد)



سیم جامپر نری- مادگی
(۵ عدد)





توضیح مدار:

ماژول Joystick، پنج پایه دارد. دو VCC و GND به پین های متناظر آردوینو متصل می شود. پایه های دیگر مربوط به خروجی پتانسیومتر راستای X، خروجی پتانسیومتر راستای Y و دکمه فشاری هستند. پایه مربوط به پتانسیومتر Y به پین آنالوگ A0، پایه مربوط به پتانسیومتر X به پایه آنالوگ A1 و پایه مربوط به دکمه فشاری به پین شماره ۲ متصل می شود. در این پروژه LED چهار شکل بالا به پین های ۱۰، ۱۱، ۱۲ و ۱۳ متصل می شوند.

وضعیت صفر اهرم، مقدار وسط بازه ۰-۱۰۲۳ را نشان می دهد. با توجه به اینکه این وضعیت به صورت دقیق نیست و درون یک بازه قرار می گیرد، در این پروژه از ۴۵۰-۵۵۰، به عنوان وضعیت صفر در نظر گرفته شده و در صورت خروج اهرم از این بازه، LED تعیین شده روشن می شود.



Joystick چوچلو



File > Examples > Electroduino > Circuit_20

```
int yPin= A0;
int xPin= A1;
int btnPin= 2;
int topLED= 13;
int leftLED= 12;
int downLED= 11;
int rightLED= 10;

void setup() {
    pinMode(btnPin, INPUT_PULLUP);
    pinMode(leftLED, OUTPUT);
    pinMode(downLED, OUTPUT);
    pinMode(rightLED, OUTPUT);
    pinMode(topLED, OUTPUT);
}

void loop() {
    int btnState = digitalRead(btnPin);
    int yState = analogRead(yPin);
    int xState = analogRead(xPin);
```





File > Examples > Electroduino > Circuit_20



Circuit_20

```
if (btnState == LOW) {
    digitalWrite (leftLED, HIGH);
    digitalWrite (downLED, HIGH);
    digitalWrite (rightLED, HIGH);
    digitalWrite (topLED, HIGH);
}
else {
    digitalWrite (leftLED, LOW);
    digitalWrite (downLED, LOW);
    digitalWrite (rightLED, LOW);
    digitalWrite (topLED, LOW);
}
if (yState > 550) {
    digitalWrite (downLED, LOW);
    digitalWrite (topLED, HIGH);
}
if (yState < 450) {
    digitalWrite (topLED, LOW);
    digitalWrite (downLED, HIGH);
}
```

Joystick چوچلو



File > Examples > Electroduino > Circuit_20

```
if (xState > 550) {  
    digitalWrite (leftLED, HIGH);  
    digitalWrite (rightLED, LOW);  
}  
if (xState < 450) {  
    digitalWrite (leftLED, LOW);  
    digitalWrite (rightLED, HIGH);  
}  
}
```





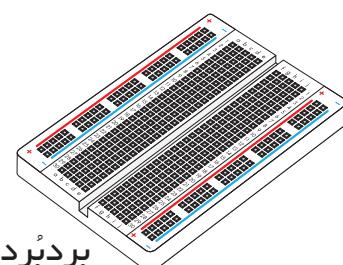
buzzer قطعات الکترونیکی هستند که در اثر اعمال ولتاژ، یک صدای "کلیک" ایجاد می‌کنند. این ویژگی به خودی خود، جذابیتی ندارد، ولی اگر این ولتاژ صدها بار در ثانیه قطع و وصل شود، این یک tone ایجاد می‌کند.

با کنار هم قرار دادن این tone‌ها، می‌توان یک ملودی ایجاد کرد.
در این پروژه نحوه ایجاد آهنگ buzzzer توسط توضیح داده می‌شود.
در این کیت از دو نوع passive و active buzzzer استفاده می‌شود.

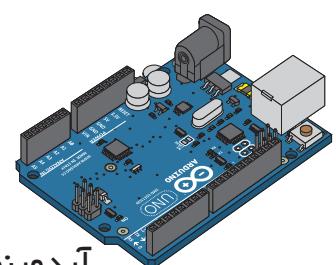
مطلوبات



سیم جامپر نری-نری
(۲ عدد)



بردبرد



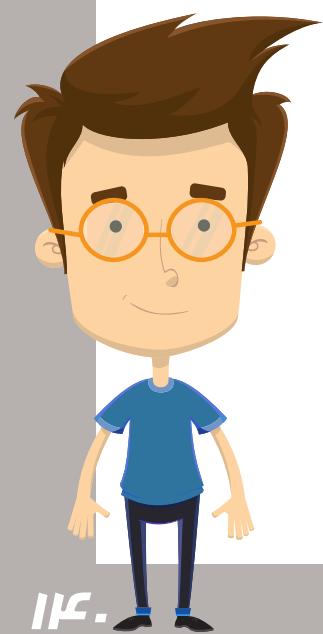
آردوینو

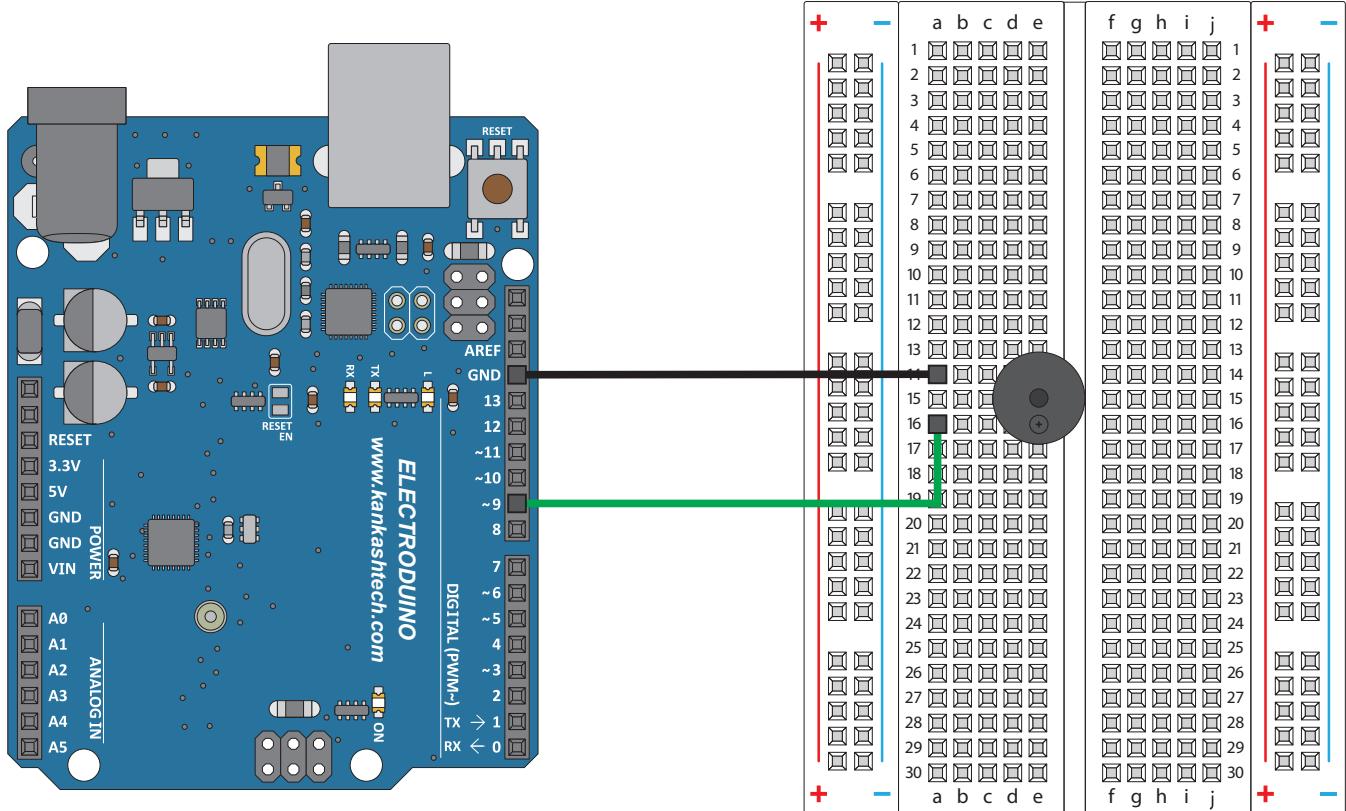


اکتیو Buzzer
(۱ عدد)



پسیو Buzzer
(۱ عدد)





توضیح مدار:

مدار این پروژه بسیار ساده است. پایه مثبت buzzer به پین ۹ آردوینو و پایه منفی آن به GND آردوینو متصل می شود. ممکن است به خاطر ابعاد buzzer، به سادگی در برده برد قرار نگیرد. در این صورت می توان با اندکی چرخاندن، آنرا در خانه های مورد نظر برابرد فروکرد.



بuzzer ها در دونوع active و passive دسته بندی می شوند. با توجه به سطح زیرین قطعه، می توان نوع آنرا مشخص کرد. در نوع passive، مدار چاپی قطعه را می توان از کف آن دید (راست تصویر)، در صورتی که در نوع active، سطح زیرین قطعه توسط یک پوشش سیاه رنگ پوشانده شده است(سمت چپ تصویر).



در نوع active، یک منبع تولید نوسان در قطعه وجود دارد و در صورتی که ولتاژ به آن اعمال شود، با فرکانس مشخص، یک صدا تولید می کند. برای فعال کردن این نوع از buzzer باید توسط دستور آنرا در وضعیت HIGH و برای قطع صدا، در وضعیت LOW قرار داد. (بخش اول کد) در نوع passive، به علت عدم وجود منبع نوسان، با اعمال ولتاژ مستقیم، صدایی تولید نمی شود. برای ایجاد صدا در این نوع buzzer، از دستور tone استفاده می شود. این دستور پین مورد نظر را با توجه به فرکانس مشخص شده، در وضعیت HIGH/LOW قرار می دهد.

به دو صورت می توان از دستور tone استفاده کرد:

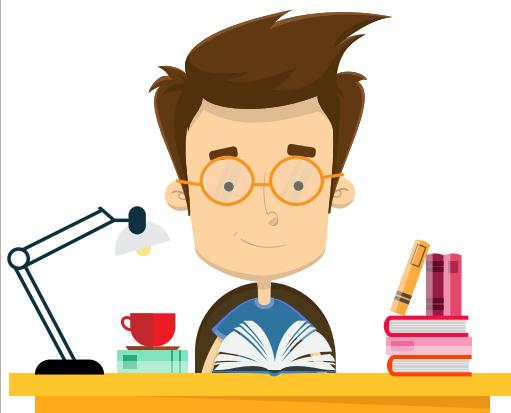
tone (pin, frequency);

tone (pin, frequency, duration);

در این پروژه از روش دوم، این دستور به کار گرفته می شود. با توجه به اینکه می توان buzzer را با فرکانس های مختلف تحریک کرد، از نوع passive برای ایجاد ملودی در این پروژه استفاده می شود. برای این کار، باید فرکانس مربوط به نت های مختلف موسیقی تعریف شود.

فرکانس نت های موسیقی a-g به شرح زیر است:

فرکانس (Hz)	نت
۲۶۲	c
۳۹۲	d
۴۴۰	e
۴۹۴	f
۵۲۳	g
۳۳۰	a
۳۱۱	b
۲۷۷	C



برای فراخوانی فرکانس، از یک تابع به نام frequency استفاده می شود.

تابع frequency

حروف متناظر نت ها و فرکانس مربوطه در دو آرایه ذخیره می شود. زمانی که یک نت باید توسط buzzer نواخته شود، حرف متناظر با آن نت به تابع frequency ارسال می شود. این تابع، حرف دریافتی را با حروف موجود در آرایه مقایسه کرده و در صورت تطیق با یکی از حروف، مقدار فرکانس متناظر با آن حرف را به برنامه اصلی بر می گرداند. در صورت عدم تطابق، عدد ۰ برگردانده می شود.

```
int frequency(char note) {
    int i;
    int numNotes = 8;
    char names[ ] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
    int frequencies[ ] = {262, 294, 330, 349, 392, 440, 494, 523};
    for (i = 0; i < numNotes; i++) {
        if (names[i] == note) {
            return(frequencies[i]);
        }
    }
    return(0);
}
```

تعریف ملودی

برای تعریف ملودی، ابتدا باید آرایه ای از نت هایی که باید پخش شوند نوشته شود. این آرایه باید شامل تمامی نت های تشکیل دهنده ملودی (حتی مکث) باشد.

این آرایه از نوع char تعریف می شود:

```
char notes[ ] = "cdfda ag cdfdg gf";
```

beat

آرایه ای شامل مقادیر مربوط به طول نت ها و همچنین مکث است. عدد ۱ بیانگر یک چهارم ، عدد ۲ بیانگر نیم و عدد ۴ بیانگر یک طول نت کامل است.

نکته ۱ : تعداد آیتم های آرایه beat باید برابر با تعداد آیتم های ملودی باشد.

نکته ۲ : همه نت ها و حتی مکث میان نت ها باید طول داشته باشند

```
int beats[ ] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,4,4,2};
```





tempo

بیانگیر سرعت پخش ملودی است. برای افزایش سرعت، باید عدد مربوط به tempo را کاهش داد.

```
int tempo=150;
```

برای تعیین طول اجرای هر نت به صورت ثانیه، باید beat آن نت در tempo ضرب شود:

```
int duration=beats[i]*tempo;
```

با داشتن آرایه نت‌ها و beat‌های هر ملودی و تنظیم tempo، می‌توان آنرا توسط buzzer پخش کرد.

تولید ملودی

برای تولید ملودی، توسط یک حلقه، نت‌های تعریف شده ملودی به ترتیب بررسی می‌شوند. در صورتی که نت، فاصله باشد، به اندازه beat آن نت، مکث انجام می‌شود. در غیر این صورت، نت مورد نظر به تابع frequency فرستاده شده و فرکانس آن فرآخوانی می‌شود. سپس با استفاده از تابع tone، نت موردنظر نواخته می‌شود.

مدت زمان نواخته شدن نت، توسط beat متناظر با ملودی و tempo تعیین می‌شود. میان اجرای هر نت، یک مکث بسیار کوتاه به اندازه یک دهم tempo ایجاد شده که بتوان تمایز میان نت‌ها را احساس کرد.

```
for (i = 0; i < songLength; i++) {
    if (notes[i] == ' ') {
        delay(duration);
    }
    else {
        tone(buzzerPin, frequency(notes[i]), duration);
        delay(duration);
    }
    delay(tempo/10);
}
```

در ادامه کد مربوط به buzzer از نوع active (کد ۲۱-۱) و سپس کد مربوط به buzzer از نوع passive (کد ۲۱-۲) ارائه شده است. شما می‌توانید با تغییر اعداد، اثرات هر پارامتر را در عملکرد buzzer امتحان کنید.



File > Examples > Electroduino > Circuit_21_1

```
int buzzerPin = 9;

void setup() {
    pinMode(buzzerPin, OUTPUT);
}

void loop() {
    int i=0;
    for(i=0; i<80; i++) {
        digitalWrite(buzzerPin,HIGH);
        delay(1);
        digitalWrite(buzzerPin,LOW);
        delay(1);
    }

    for(i=0; i<100; i++) {
        digitalWrite(buzzerPin,HIGH);
        delay(2);
        digitalWrite(buzzerPin,LOW);
        delay(2);
    }
}
```



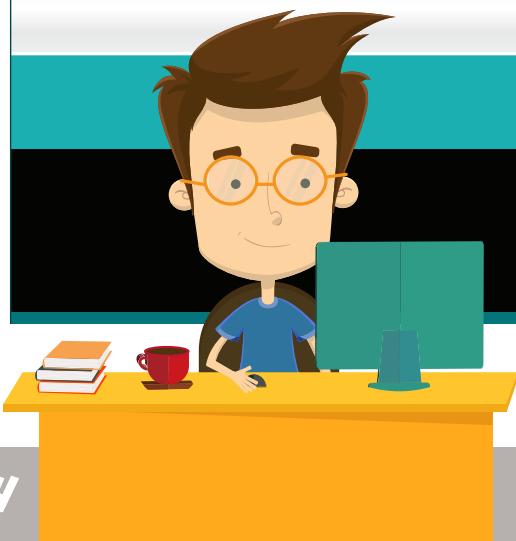


Circuit_21_2

```
int buzzerPin = 9;
int songLength = 18;
char notes[ ] = "cdfda ag cdfdg gf ";
int beats[ ] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,4,4,2};
int tempo = 150;

void setup() {
    pinMode(buzzerPin, OUTPUT);
}

void loop() {
    int i, duration;
    for (i = 0; i < songLength; i++) {
        duration = beats[i] * tempo;
        if (notes[i] == ' ') {
            delay(duration);
        }
        else {
            tone(buzzerPin, frequency(notes[i]), duration);
            delay(duration);
        }
    }
}
```



File > Examples > Electroduino > Circuit_21_2

```
delay(tempo/10);
}

int frequency(char note) {
    char names[ ] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
    int frequencies[ ] = {262, 294, 330, 349, 392, 440, 494, 523};
    for (int i = 0; i < 8; i++) {
        if (names[i] == note) {
            return(frequencies[i]);
        }
    }
    return(0);
}
```





رله یک سوییچ مکانیکی است که توسط جریان الکتریسیته کنترل می‌شود. درون پوسته پلاستیکی رله، یک آهنربای الکتریکی است که در اثر وصل شدن ولتاژ به دو سر آن، باعث جابجا شدن یک دسته آهنی می‌شود که اتصال میان دو نقطه را قطع و دو نقطه دیگر را به یکدیگر وصل می‌کند.

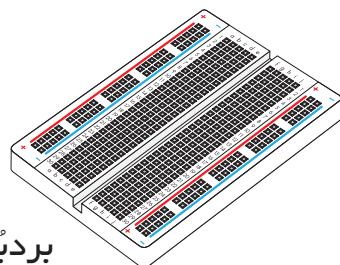
آهنربای الکتریکی رله استفاده شده در الکترودوینو، با اعمال ولتاژ ۵ ولت، فعال شده و دسته آهنی را جابجا می‌کند.

در این پروژه از ماژول رله برای روشن و خاموش کردن یک چراغ با برق شهر (۲۲۰ ولت) استفاده شده است. در صورتی که سنسور تشخیص دهد که نور کم است، به صورت خودکار چراغ روشن می‌شود.

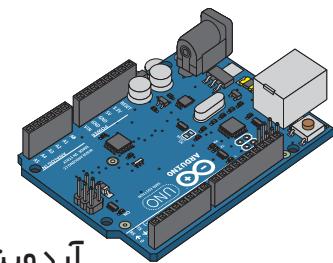
مطلوبات



سیم جامپر نری-نری
(۴۰ عدد)



بردبرد



آردوینو



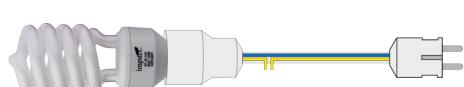
ماژول رله
(۱ عدد)



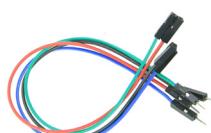
سنسور شدت نور
(۱ عدد)



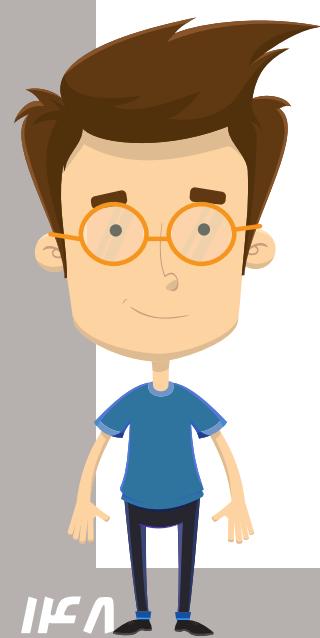
مقاومت ۱ کیلو اهم
(۱ عدد)

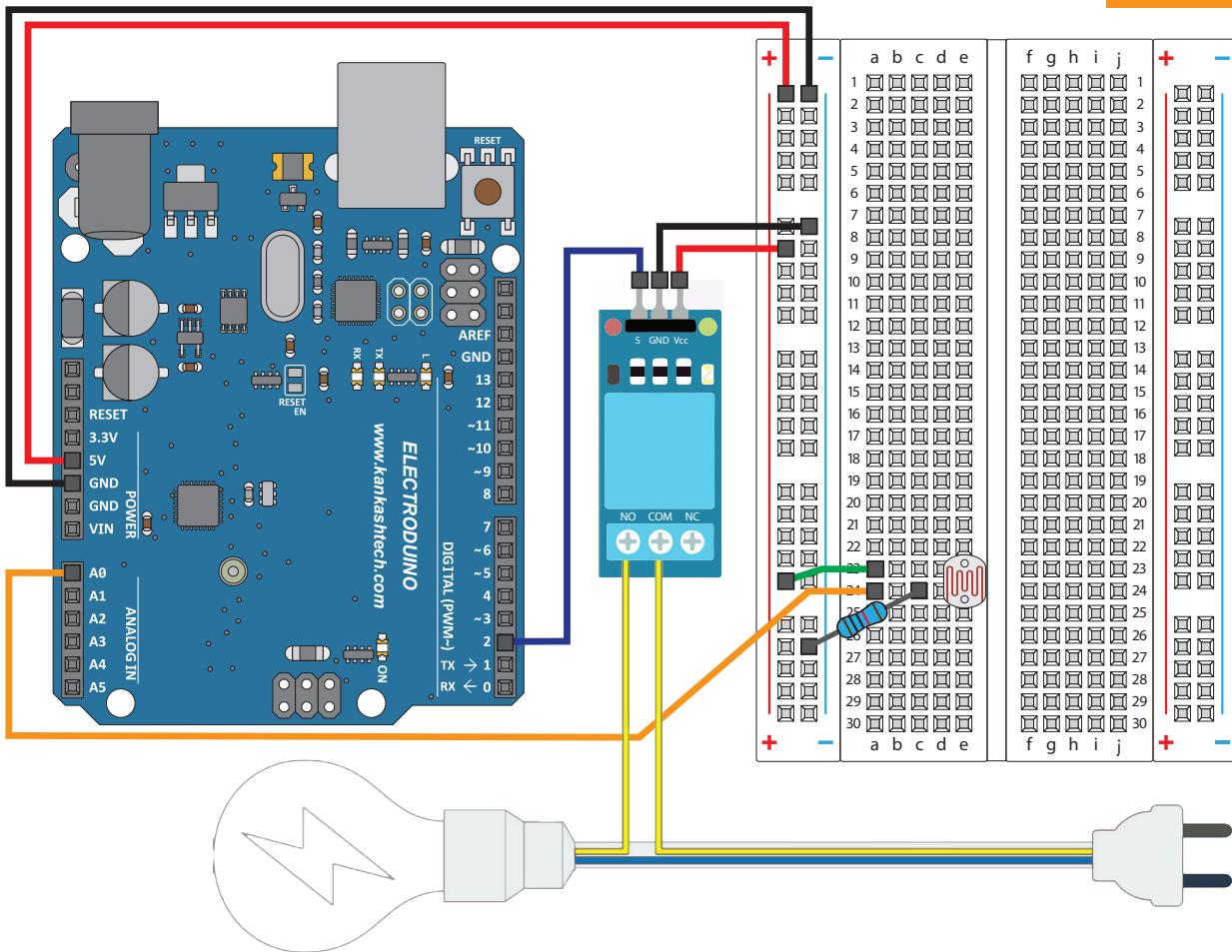


لامپ و دوشاخه
(در کیت موجود نیست)



سیم جامپر نری- مادگی
(۳ عدد)





توضیح مدار:

ماژول رله ۶ نقطه اتصال دارد. سه تا از این نقاط، پایه های ارتباطی تصمیم گیری و سه نقطه دیگر، سوکت های کنترلی هستند.

پایه های تصمیم گیری که در شکل بالا، در پایین ماژول واقع شده اند، شامل VCC، GND و IN هستند که به ترتیب به +5V، GND و پین ۲ آردوینو متصل می شوند.

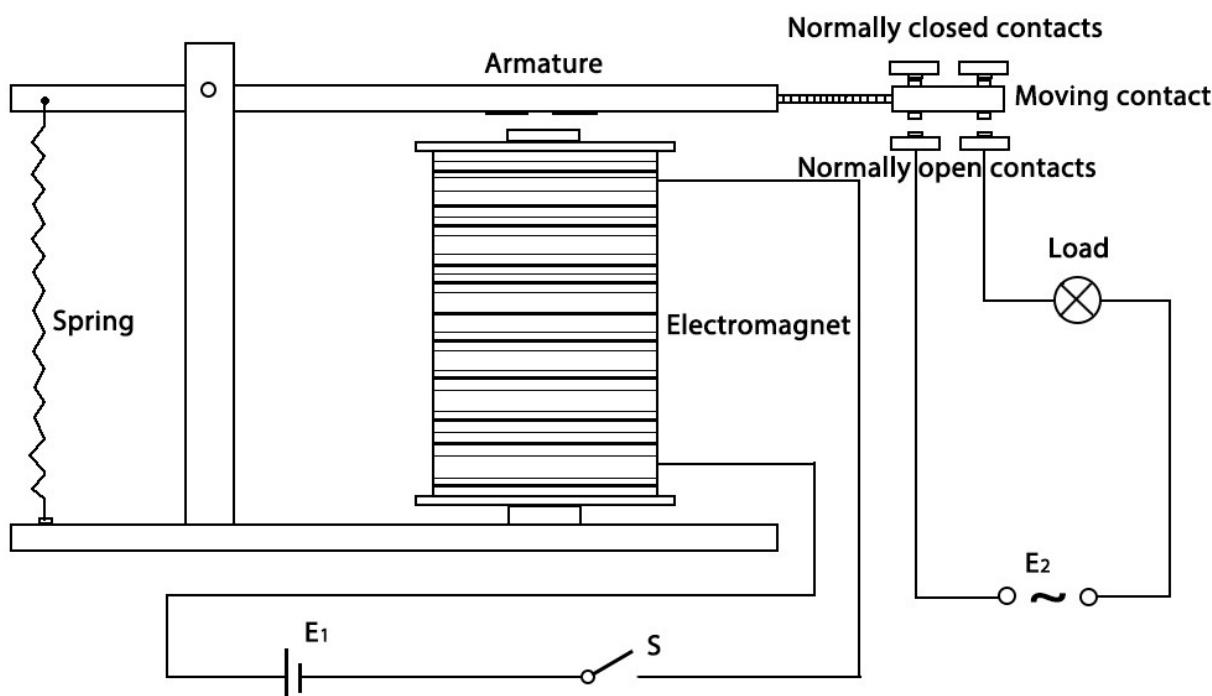
سوکت های کنترلی، سوکت مشترک COM (جريان اصلی ورودی به رله COMMON)، سوکت مدار بسته NC (مداری که به صورت پیش فرض متصل است Normally Close) و سوکت مدار باز NO (مداری که به صورت پیش فرض قطع است Normally Open) هستند. در این پروژه، جريان اصلی ورودی از دوشاخه به COM و خروجی به سمت لامپ، به NO وصل می شود. سیم دیگر لامپ مستقیماً به دوشاخه وصل می شود.

دقت داشته باشید که در هنگام بستن سیم ها به رله، دوشاخه حتماً از پریز خارج شده باشد.



رله ها برای ایجاد اتصال میان دو یا چند نقطه از مدار یا دستگاه در اثر اعمال سیگنال الکتریکی استفاده می شوند. به عبارت دیگر، رله برای جداسازی میان کنترلر و دستگاه استفاده می شود. آنجایی که سیگنال ارسالی به رله از میکروکنترلر به صورت DC و دستگاه ها عموما با ولتاژ AC کار می کنند. در نتیجه از رله برای حل این تفاوت استفاده می شود. مزیت اصلی رله ها ایجاد امکان کنترل جریان یا ولتاژ بزرگ توسط یک سیگنال الکتریکی کوچک است.

در شکل زیر، توسط یک سیگنال AC (مانند برق شهر) کنترل می شود.



هر رله از ۵ بخش تشکیل شده است:

- ۱. آهنربای الکتریکی (electromagnet):** یک هسته آهنی سیم پیچیده شده؛ در هنگام عبور جریان از سیم، هسته آهنی به آهنربا تبدیل می شود.
- ۲. نقاط اتصال (contacts):** دو نقطه اتصال وجود دارد، NC و NO
 - مدار بسته پیش فرض (Normally Close)؛ در هنگام غیرفعال بودن رله این مدار برقرار است.
 - مدار بسته پیش فرض (Normally Open)؛ در هنگام فعال بودن رله این مدار برقرار است.
- ۳. دسته آهنی متحرک (armature):** هنگامی که هسته آهنی در اثر اعمال جریان، آهنربا می شود، دسته آهنی را جذب کرده و باعث قطع شدن مدار NC و اتصال مدار NO می شود.
- ۴. فنر (spring):** هنگامی که جریانی وجود ندارد، فنر دسته آهنی را به سمت نقطه اتصال NC می کشد و مدار NO را قطع می کند.
- ۵. پوسته پلاستیکی:** تمامی بخش های رله درون یک پوسته پلاستیکی قرار گرفته اند



ماژول رله

ماژول رله استفاده شده برای این راهنمای، در اثر اعمال سیگنال **LOW** فعال و در صورت اعمال سیگنال **HIGH** غیرفعال می شود.

با توجه به اینکه در مدار این پروژه، لامپ به خروجی NO رله متصل شده است، برای روشن شدن لامپ، باید رله فعال شود. در نتیجه پایه ۹ باید در وضعیت **LOW** قرار گیرد.

دقت داشته باشید که در رله های ساخته شده توسط شرکت های مختلف، محل پایه ها ممکن است متفاوت باشد. لذا در هنگام بستن مدار و کد نویسی حتما به نوشته های کنار پایه های ماژول دقต شود.





File > Examples > Electroduino > Circuit_22



Circuit_22



```
int sensorPin=A0;
int relayPin= 2;

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

void loop() {
    digitalWrite (relayPin, HIGH);
    int sensorValue=analogRead(sensorPin);
    if (sensorValue<400) {
        digitalWrite (relayPin, LOW);
        Serial.println("Relay Activated");
    }
    Serial.println(sensorValue);
    delay(200);
}
```



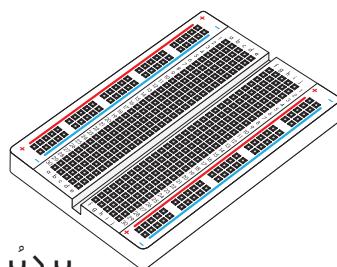
سروو موتور

سروو موتورها گونه ای از موتورهای الکتریکی هستند که می تواند برای موقعیت دهی استفاده شود. مزیت این موتورها نسبت به موتورهای DC ساده، کنترل زاویه شفت موتور است. با تغییر ولتاژ اعمالی به سرووموتور، می توان سروو را به یک موقعیت مشخص جابجا کرد. در این پروژه با استفاده از یک پتانسیومتر، موقعیت قرارگیری سروو موتور کنترل می شود.

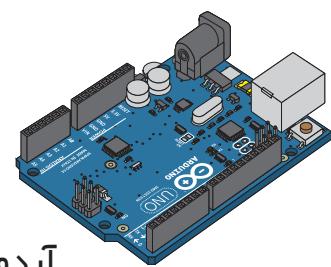
مطلوبات



سیم جامپر نری-نری
(۸ عدد)



بردبرد



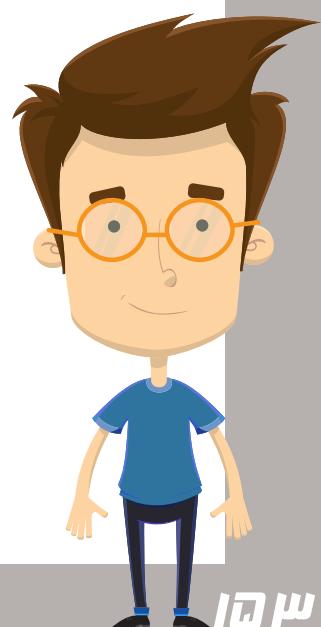
آردوینو

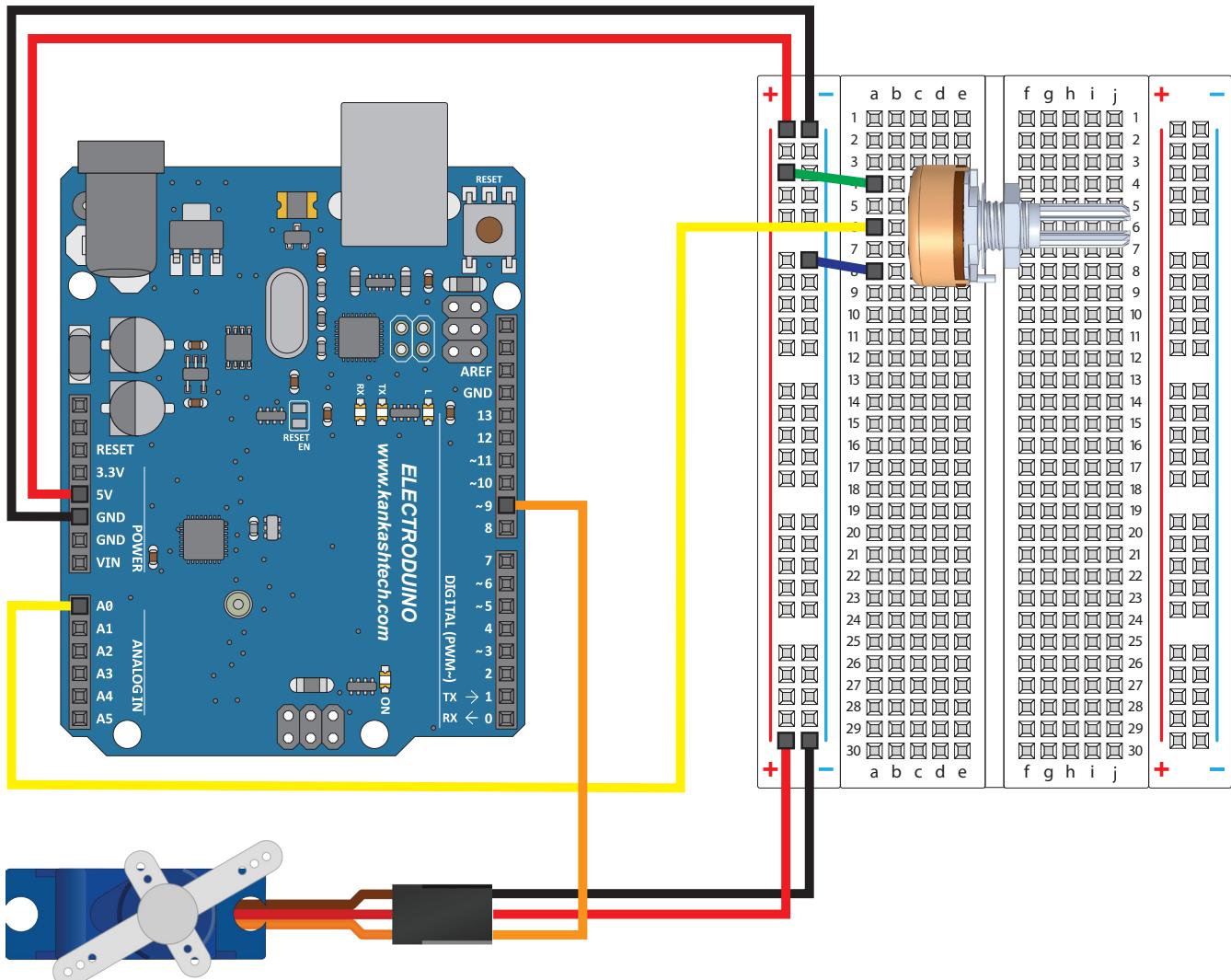


سرووموتور
(۱ عدد)



پتانسیومتر
(۱ عدد)





توضیح مدار:

مدار این پروژه بسیار ساده است. سیم نارنجی رنگ، مربوط به سیگنال بوده و به پین ۹ آردوینو متصل می شود. سیم های قرمز و مشکی سرو موتور به ترتیب مربوط به VCC و GND بوده که به ترتیب به پین های +5V و GND آردوینو متصل می شود.



سرو و موتور



برای استفاده از سرو و موتور، ابتدا باید کتابخانه مربوط به آن، فراخوانی شود:

```
#include <Servo.h>
```

مشابه قطعات دیگری نظیر DHT11 و LCD1602، برای به کارگیری سنسور، باید یک قطعه از نوع سرو و موتور تعریف شود:

```
Servo myServo;
```

در اینجا myServo قطعه ای از نوع Servo تعریف شده است و می‌توان دستورات مورد نیاز استفاده از آن را به کار گرفت. ابتدا باید مشخص شود سرو و موتور به چه پینی از آردوینو متصل شده است:
`myServo.attach(9);`

برای موقعیت دهی از دستور زیر استفاده می‌شود:

```
myServo.write(100);
```

در دستور بالا، سرو و موتور در زاویه ۱۰۰ درجه قرار می‌گیرد. سرو و موتور استفاده شده برای این راهنمای از نوع SG90 است که بازه حرکتی ۱۸۰ - ۰ درجه دارد.

در این پروژه از یک پتانسیومتر برای کنترل موقعیت سرو و موتور استفاده می‌شود. با توجه به اینکه خروجی پتانسیومتر بین ۰ - ۱۰۲۳ است، با استفاده از دستور map برای تبدیل بازه ۰ - ۱۰۲۳ به بازه ۰ - ۱۸۰ استفاده می‌شود:

```
int sensorPin = A0;  
int sensorValue = analogRead(sensorPin);  
int position = map(sensorValue, 0, 1023, 0, 180);
```

برای استفاده از پتانسیومتر در موقعیت دهی به سرو و موتور، از دستور زیر استفاده می‌شود:

```
myServo.write(position);
```





Circuit_23



```
#include <Servo.h>

int sensorPin =A0;
Servo myServo;

void setup () {
    myServo.attach (9);
    myServo.Write (0);
    delay (1000);
    myServo.Write (90);
    delay (1000);
    myServo.Write (180);
    delay (1000);
}

void loop () {
    int sensorValue = analogRead (sensorPin);
    int position = map (sensorValue, 0, 1023, 0, 180);
    myServo.Write (position);
}
```



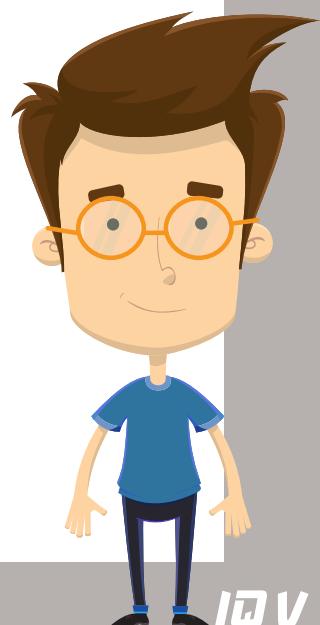
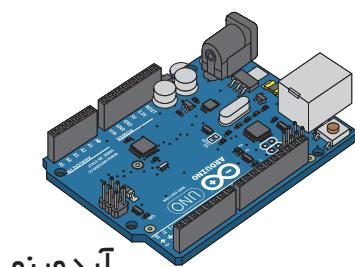
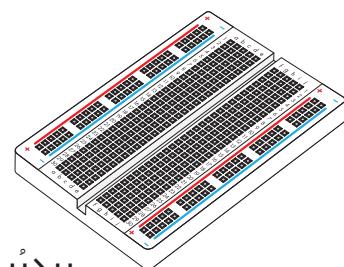
موتور DC

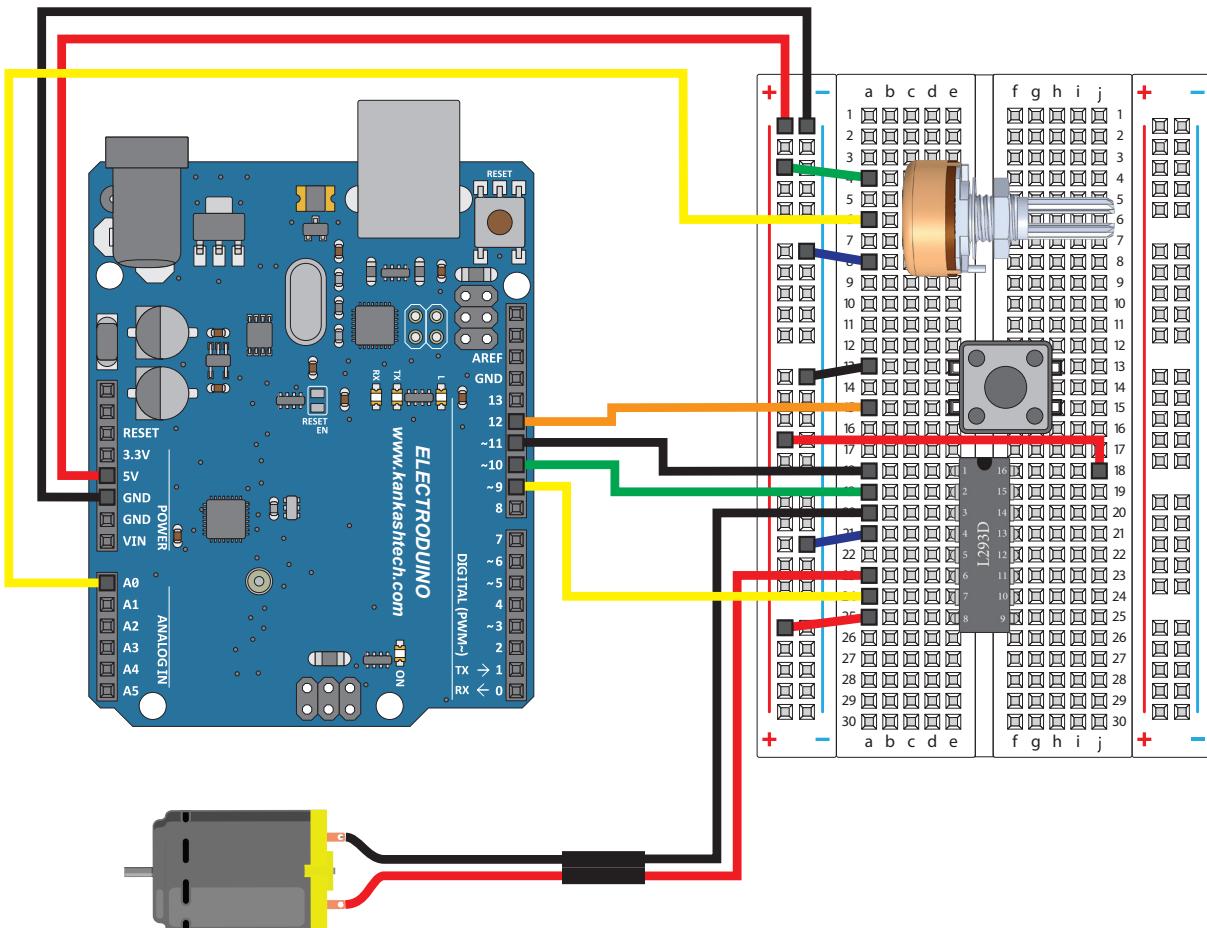
در بسیاری از وسایل نظیر اسباب بازی ها، تنها نیاز است که سرعت و جهت حرکت موتور تنظیم شود و نیازی به کنترل موقعیت نیست.

برای این کاربردها، عموماً از موتورهای DC ساده استفاده می شود.

در این پروژه، سرعت و جهت حرکت یک موتور DC توسط یک آی سی L293D کنترل می شود.

مطلوبات





توضیح مدار:

آی سی L293D، توانایی کنترل جهت و سرعت دو موتور DC را دارد. برای کنترل سرعت، پایه فعال سال (شماره ۱) به پین ۱۱ آردوینو متصل می شود. برای هر موتور، دو ورودی و دو خروجی وجود دارد. دو خروجی موتور شماره ۱ (پایه شماره ۳ و شماره ۶) مستقیماً به موتور DC متصل می شود. ورودی شماره ۱ موتور ۱ (پایه شماره ۲) به پین شماره ۱۰ آردوینو و ورودی شماره ۲ موتور ۱ (پایه شماره ۷) به پین شماره ۹ آردوینو متصل می شود.

یک کلید فشاری به پین شماره ۱۲ و یک پتانسیومتر به پین آنالوگ A0 متصل می شود (پایه های کناری پتانسیومتر به +5V و GND و پایه وسط به A0 متصل شود)



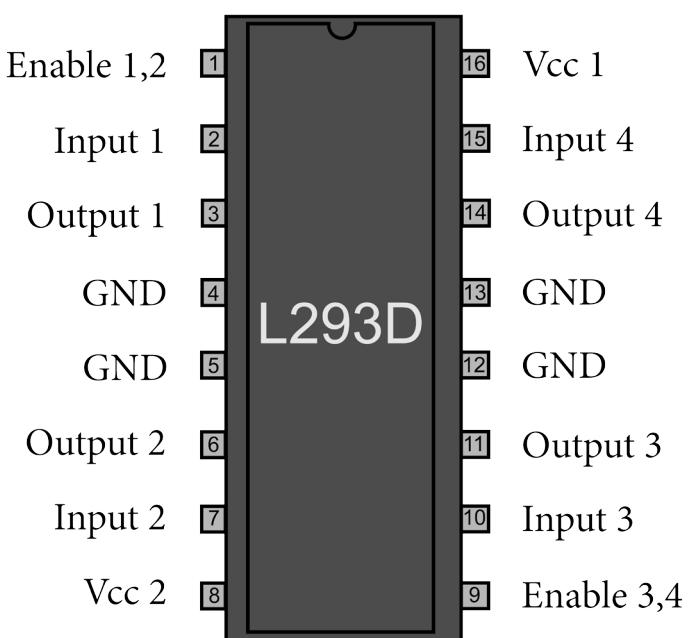
موتورهای DC عملکرد ساده‌ای دارند. در صورتی که جهت اتصال سیم‌های موتور برعکس شود (GND و +5V) جهت چرخش آن تغییر می‌کند.

با توجه به اینکه تعویض سیم‌های موتور کار دشواری است، به یک واسطه نیاز است که در موقع موردنیاز، با ارسال دستور از طرف آردوینو، GND و +5V متصل به موتور را جابجا کند.

در این پروژه از آی‌سی L293D برای کنترل سرعت و همچنین جهت دوران موتور DC استفاده می‌شود. این آی‌سی قابلیت کنترل همزمان دو موتور DC را دارد.

ساختار L293D به صورت زیر است:

عملکرد	نام پین	شماره پین	عملکرد	نام پین	شماره پین
پایه فعال کننده موتور ۱	Enable 1,2	۱	ولتاژ تغذیه	Vcc 1	۱۶
ورودی ۱ موتور ۱	Input 1	۲	ورودی ۲ موتور ۲	Input 4	۱۵
خروجی ۱ موتور ۱	Output 1	۳	خروجی ۲ موتور ۲	Output 4	۱۴
زمین (۰ ولت)	GND	۴	زمین (۰ ولت)	GND	۱۳
زمین (۰ ولت)	GND	۵	زمین (۰ ولت)	GND	۱۲
خروجی ۱ موتور ۲	Output 2	۶	خروجی ۱ موتور ۲	Output 3	۱۱
ورودی ۲ موتور ۱	Input 2	۷	ورودی ۱ موتور ۲	Input 3	۱۰
ولتاژ تغذیه	Vcc 2	۸	پایه فعال کننده موتور ۲	Enable 3,4	۹



پایه فعال کننده

برای اینکه یک موتور توسط L293D کنترل شود، ابتدا باید پین فعال کننده آن موتور در وضعیت HIGH قرار بگیرد. این کار می‌تواند با اتصال این پایه فعال کننده به صورت مستقیم به $+5V$ انجام شده و یا توسط یکی از پین‌های آردوینو و دستور `digitalWrite` این پایه در وضعیت HIGH قرار داده شود. در این حالت موتور در سرعت ماکزیمم حرکت می‌کند.

در صورتی که پایه فعال کننده موتور، به یکی از پین‌های با قابلیت PWM متصل شود، با استفاده از دستور `analogWrite` می‌توان سرعت چرخش را کنترل کرد.

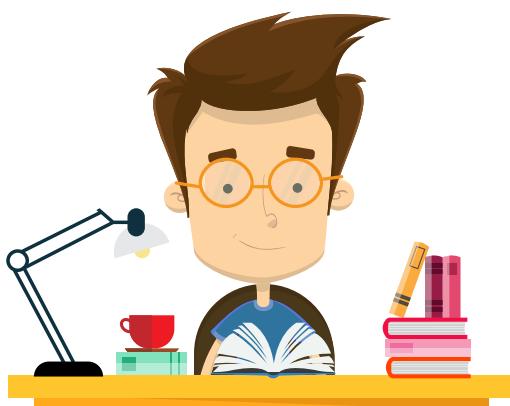
پایه‌های ورودی

هر موتور دو ورودی دارد که می‌تواند در وضعیت‌های HIGH و LOW قرار گیرد. چهار وضعیت برای هر موتور امکان‌پذیر است:

عملکرد موتور	پین ورودی ۲	پین ورودی ۱
موتور متوقف است.	GND	GND
موتور در جهت A حرکت می‌کند.	GND	$+5V$
موتور در جهت معکوس A حرکت می‌کند.	$+5V$	GND
موتور متوقف است.	$+5V$	$+5V$

برای کنترل سرعت از پتانسیومتر و برای کنترل جهت از یک دکمه فشاری استفاده می‌شود. در پروژه ۳، نحوه استفاده از دکمه‌های فشاری و مقاومت Pull-up ذکر شد. پین‌های آردوینو یک مقاومت داخلی دارند که می‌تواند به عنوان مقاومت pull-up استفاده شود. برای استفاده از این ویژگی باید در هنگام تعریف پین، این ویژگی فراخوانی شود: `pinMode(switchPin, INPUT_PULLUP);`

در این صورت، دیگر نیازی به استفاده از مقاومت مجازی PULLUP نیست.



File > Examples > Electroduino > Circuit_24

```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 12;
int potPin = A0;
void setup() {
    pinMode (in1Pin, OUTPUT);
    pinMode (in2Pin, OUTPUT);
    pinMode (enablePin, OUTPUT);
    pinMode (switchPin, INPUT_PULLUP);
}
void loop() {
    int speed = analogRead(potPin) / 4;
    int state = digitalRead (switchPin);
    if (state == HIGH) {
        analogWrite (enablePin, speed);
        digitalWrite (in1Pin, HIGH);
        digitalWrite (in2Pin, LOW);
    }
    else {
        analogWrite (enablePin, speed);
        digitalWrite (in1Pin, LOW);
        digitalWrite (in2Pin, HIGH);
    }
}
```



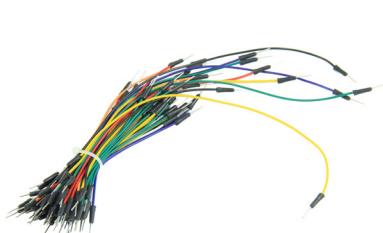
استپ موتور

۲۰

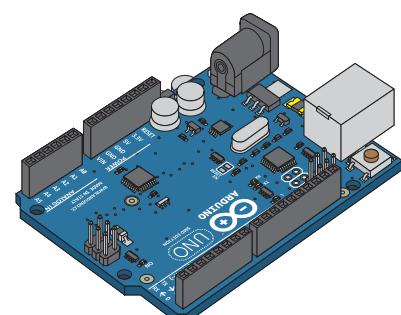
استپ موتورها چیزی میان سرو و موتور و موتورهای DC هستند. استپ موتورها این مزیت را دارند که می توانند به صورت دقیق موقعیت دهی شوند بدون آنکه به سنسور یا انکوادر موقعیت نیاز داشته باشند. این موتورها می توانند یک پله (step) به جلو یا عقب حرکت کنند ولی می توانند به طور پیوسته هم حرکت کنند.

در این پروژه کنترل یک استپ موتور از نوع BYJ48 و ماژول درایور ULN2003A توضیح داده می شود

مطلوبات



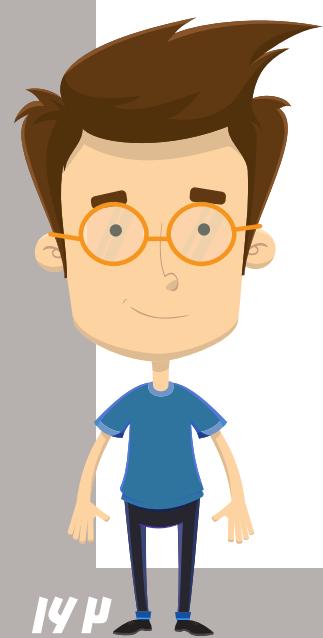
سیم جامپر نری-نری
(۶ عدد)

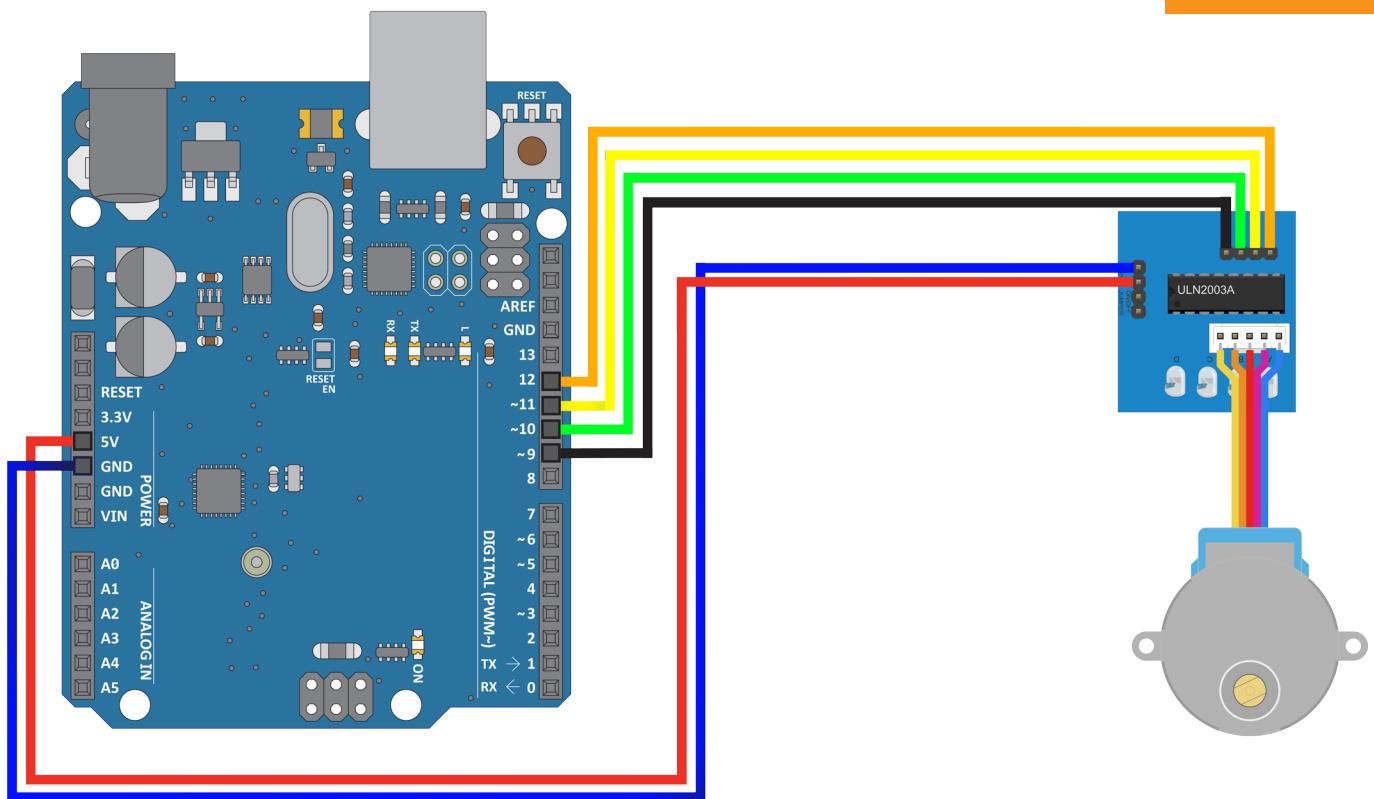


آردوبینو



استپ موتور و درایور
(۱ عدد)





توضیح مدار:

با توجه به شکل سوکت استپ موتور، تنها به یک صورت می‌توان موتور را به درایور ULN2003A متصل کرد.

درایور ULN2003، شش عدد پین دارد. دو پین مربوط به VCC و GND است که مطابق با علامت های + و - درج شده بر روی آن، به پین های متناظر GND و 5V آردوینو متصل می شود.

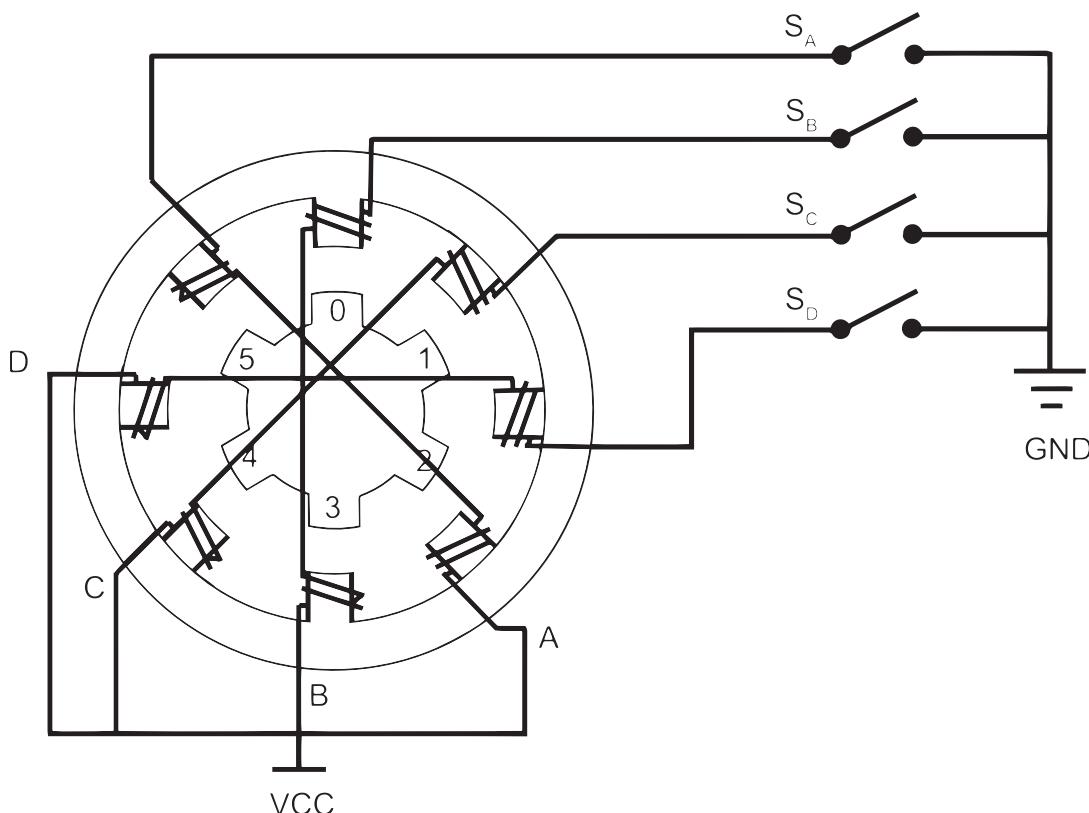
۴ پین دیگر درایور با نام های IN1-IN4 نامگذاری شده که به ترتیب به پین های ۱۲-۹ آردوینو متصل می شود. این پایه ها، دوران استپ موتور را کنترل می کنند.



استپ موتور

کانکاس

استپ موتور موجود در این مجموعه، چهار فاز است. با اعمال جریان به سیم پیچ هر فاز در توالی مشخص، می توان باعث چرخش موتور به صورت پله به پله شد. شماتیک این استپ موتور به صورت زیر است:



در شکل فوق، در وسط موتور، یک روتور چرخدنده ای شکل از جنس آهنربای دائم است. روتور ۶ دندانه در اطراف خود دارد. در اطراف روتور، ۸ قطب آهنربای الکتریکی وجود دارد که سیم پیچ قطب های رو برو به یکدیگر متصل است. در نتیجه ۴ جفت قطب آهنربایی وجود دارد که فاز نامیده می شود. ۴ فاز A-D توسط سیم هایی به ۴ سوییچ S_A-S_D متصل شده اند. در واقع فاز ها با یکدیگر موازی و دو قطب آهنربایی یک فاز، به صورت سری به همدیگر ارتباط دارند.

در شکل فوق، هنگامی که S_B وصل و سایر سوییچ ها قطع باشند، قطب های آهنربایی ۰ و ۳ در راستای قطب های فاز B قرار می گیرند. ولی سایر دندانه ها با اختلاف زاویه نسبت به قطب های مقابل خود قرار می گیرند.

هنگامی که S_C وصل و سایر سوییچ ها قطع شوند، در اثر ایجاد میدان مغناطیسی فاز C، روتور چرخیده تا دندانه های ۱ و ۴ در برابر قطب های فاز C قرار گیرند.

با باز و بسته شدن متوالی سوییچ ها، یک حرکت پله ای (استپ) ایجاد می شود.

با افزایش تعداد دندانه روتور، دقیق استپ موتور افزایش می یابد.



برای استفاده از استپ موتور، از کتابخانه مربوط به آن استفاده می شود:

```
#include <Stepper.h>
```

یک متغیر نماینده تعداد استپ در یک دور شفت، تعریف می شود:

```
int stepsPerRevolution = 2048;
```

در اینجا، یک دور شفت موتور به ۲۰۴۸ پله تقسیم می شود. در نتیجه در صورتی که موتور ۲۰۴۸ پله حرکت کند، یک دور کامل زده می شود.

تعریف استپ موتور

پس از این باید استپ موتور تعریف شود. برای تعریف استپ موتور از دستور زیر استفاده می شود:
Stepper myStepper(stepsPerRevolution, 12, 10, 11, 9);

ترتیب پین ها به صورت IN1, IN3, IN2, IN4 است.

تنظیم سرعت چرخش

پیش از راه اندازی موتور، ابتدا باید سرعت آن تنظیم شود:

```
myStepper.setSpeed (16);
```

در اینجا، سرعت چرخش، ۱۶ دور در دقیقه تنظیم شده است. محاسبات مربوط به سرعت، در کتابخانه نوشته شده و فقط نیاز است که دستور مربوط به آن فراخوانی شود.

دستور چرخش

پس از تنظیم سرعت چرخش، تعداد استپ چرخش باید به موتور ارسال شود. با توجه به اینکه در ابتدای برنامه، یک دور کامل، ۲۰۴۸ استپ تعریف شد، در صورتی که فرمان چرخش به میزان ۲۰۴۸ استپ ارسال شود، موتور یک دور کامل می چرخد:

```
myStepper.step (2048);
```

در صورتی که نیاز به حرکت در جهت معکوس باشد، باید تعداد استپ به صورت عدد منفی وارد شود:

```
myStepper.step (-2048);
```





File > Examples > Electroduino > Circuit_25



Circuit_25



```
#include <Stepper.h>

int stepsPerRevolution = 2048;

Stepper myStepper(stepsPerRevolution, 12, 10, 11, 9);

void setup() {
    myStepper.setSpeed(16);
    Serial.begin(9600);
}

void loop() {
    Serial.println("clockwise");
    myStepper.step(stepsPerRevolution);
    delay(500);

    Serial.println("counterclockwise");
    myStepper.step(-stepsPerRevolution);
    delay(500);
}
```



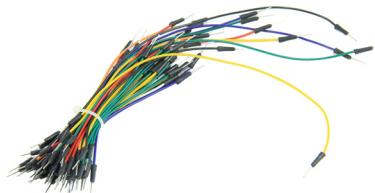
سنسور فاصله سنج اولتراسونیک

ماژول فاصله سنج اولتراسونیک HC-SR04 با ارسال یک موج صوتی و محاسبه زمان بازگشت موج به سنسور پس از برخورد با مانع، فاصله سنسور تا مانع را محاسبه می کند. این سنسور برای ساخت ربات های MAZE و ربات های اجتناب از مانع (obstacle avoidance) و کاربردهایی از این قبیل استفاده می شود. در این پروژه نحوه استفاده از این سنسور و نمایش فاصله سنسور تا موانع در مانیتور سریال ارائه شده است.

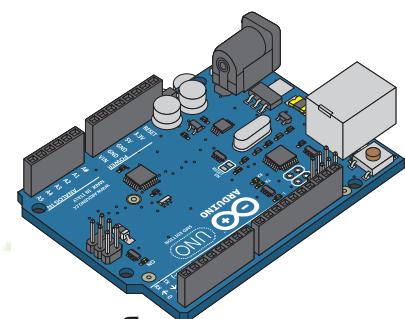
مطلوبات



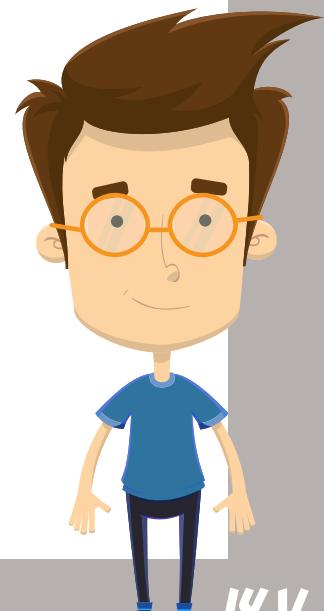
سنسور فاصله سنج اولتراسونیک
(۱ عدد)

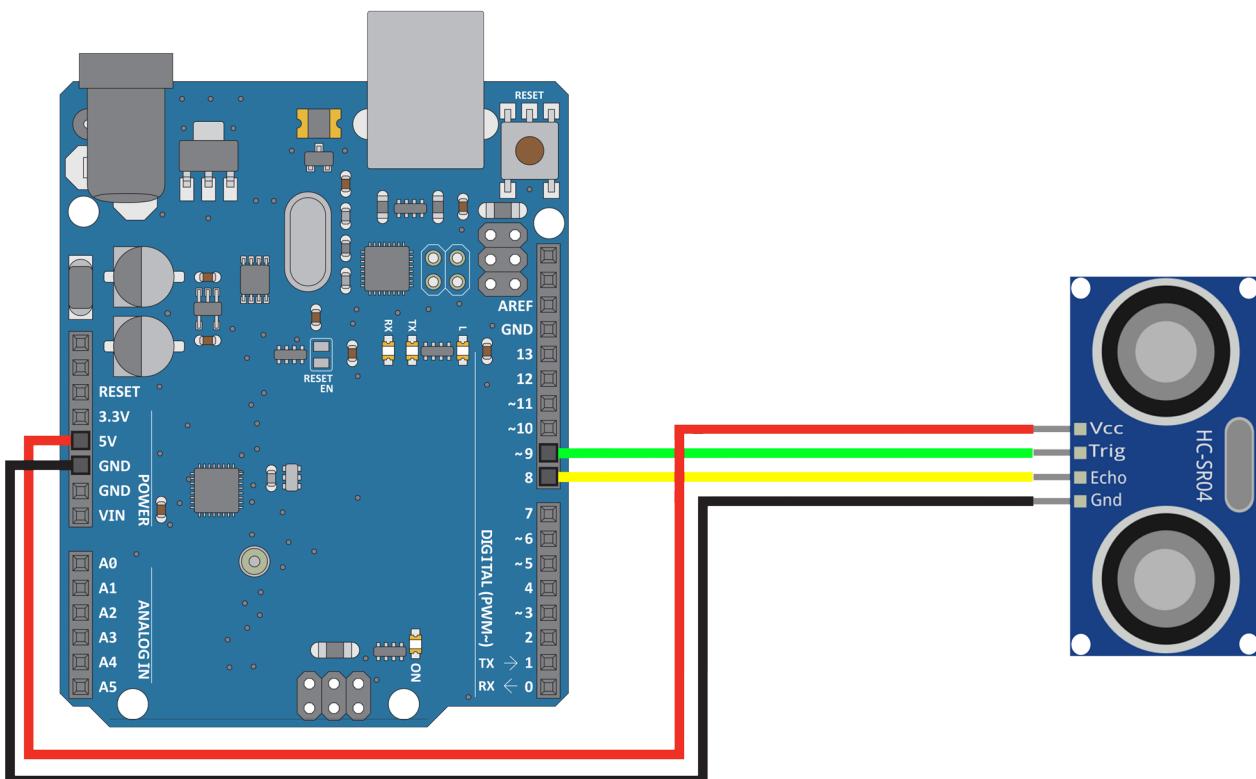


سیم جامپر نری-نری
(۴۰ عدد)



آردوینو





توضیح مدار:

ماژول HC-SR04 چهار پایه دارد. پایه های VCC و GND به ترتیب به پین های ۵ ولت و GND آردوینو متصل می شود. پایه Trig به پین دیجیتال ۹ و پایه Echo به پین دیجیتال ۸ آردوینو متصل می شود.



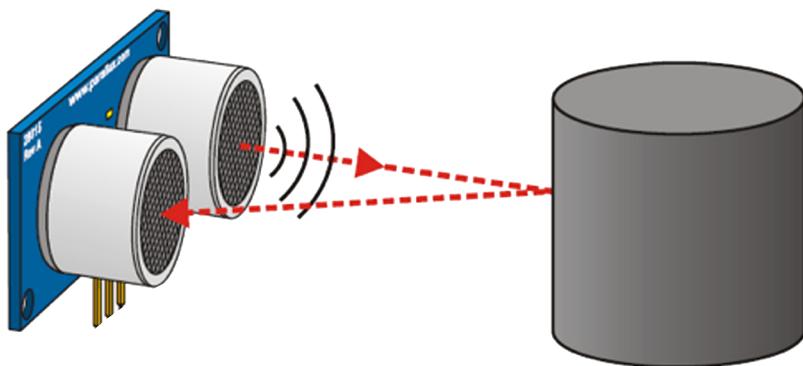
سنسور فاصله سنج اولتراسونیک

۱۴

اندازه گیری توسط موج فراصوتی

با ارسال موج فراصوتی توسط سنسور فاصله سنج اولتراسونیک و محاسبه زمان بازگشت آن پس از برخورد با مانع، می‌توان فاصله سنسور تا مانع را به دست آورد. امواج فراصوتی مانند امواج صوتی، با سرعت 340 متر بر ثانیه در فضای حرکت می‌کنند.

لازم به ذکر است که مسافت طی شده توسط موج فراصوتی از ارسال تا برگشت، دو برابر فاصله سنسور از مانع است؛ در نتیجه برای به دست آوردن فاصله، باید مسافت طی شده توسط موج، در بازه ارسال تا دریافت موج را تقسیم بر دو کرد.



با توجه به سرعت بالای موج فراصوتی، در محاسبات و کدنویسی از واحد میکروثانیه (10^{-6} ثانیه) استفاده می‌شود. به عنوان مثال به جای `delay` که واحد آن میلی ثانیه (10^{-3} ثانیه) است، از دستور `delayMicroseconds` استفاده می‌شود. همچنین به جای متر از سانتیمتر استفاده می‌شود. در نتیجه سرعت صوت به جای 340 متر بر ثانیه، 0.534 سانتیمتر بر میکروثانیه محاسبه می‌شود.

نحوه ایجاد موج فراصوتی

برای ایجاد موج فراصوتی، ابتدا `LOW` قرار داده می‌شود تا اطمینان حاصل شود که پین `Trig` از قبل در وضعیت `HIGH` نیست. سپس به مدت 10 میکروثانیه، این پین در وضعیت `HIGH` قرار داده می‌شود. با استفاده از دستور `pulseIn`، زمان رسیدن پالس ارسالی به پین `Echo` اندازه گیری و به مسافت تبدیل می‌شود

دستور `pulseIn()`

دستور `pulseIn` دو متغیر پین و مقدار مورد اندازه گیری را دریافت می‌کند:

`pulseIn(pin, value);`

متغیر `pin` شماره یا نام پینی است که قرار است پالس بر روی آن خوانده شود.

متغیر `value` نوع پالسی است که قرار است خوانده شود که دو مقدار `HIGH/LOW` خواهد بود.

خروجی دستور، طول پالس به واحد میکروثانیه است.



سنسور فاصله سنج اولتراسونیک

۳

نحوه عملکرد `pulseIn`

دستور `pulseIn` متنظر می‌ماند که `pin`، در وضعیت `value` قرار گیرد؛ به عنوان مثال `pulseIn (echoPin, HIGH);`

در دستور بالا، آردوینو متنظر می‌ماند که پین `echoPin` در وضعیت `HIGH` قرار گیرد وقتی یک پالس `HIGH` به پین می‌رسد، دستور `pulseIn`، مدت زمانی که پین در وضعیت `LOW` بوده تا ابتدای دریافت پالس `HIGH` را برمی‌گرداند. از این عملکرد به نحوه زیر در اندازه گیری فاصله استفاده می‌شود:

۱. `trigPin` به مدت ۲ میکروثانیه در وضعیت `LOW` قرار می‌گیرد. پین `echoPin` نیز پالسی دریافت نکرده و در وضعیت `LOW` قرار می‌گیرد.
۲. `trigPin` به مدت ۱۰ میکروثانیه در وضعیت `HIGH` قرار می‌گیرد و یک موج فراصوتی توسط سنسور ارسال می‌شود.
۳. موج فراصوتی پس از برخورد به مانع، به سمت سنسور برمی‌گردد.
۴. با رسیدن موج برگشتی به سنسور، `echoPin` در وضعیت `HIGH` قرار می‌گیرد

۵. دستور `pulseIn`، مدت زمان میان `HIGH` شدن `trigPin` تا `echoPin` که برابر با زمان طی شده توسط موج (رفت و برگشت از سنسور تا مانع) را برمی‌گرداند.

با تبدیل این زمان به مسافت و تقسیم بر دو، فاصله سنسور از مانع به دست می‌آید:
`duration=pulseIn(echoPin, HIGH);`

فاصله بر حسب سانتیمتر عبارت است از:

$$distance = duration \times 0.034 / 2;$$



سنسور فاصله سنج اولتراسونیک

م

File > Examples > Electroduino > Circuit #26

```
int trigPin = 9;
int echoPin = 8;
long duration;
int distance;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
}

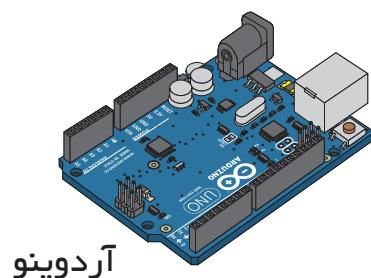
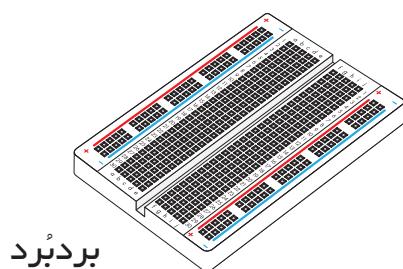
void loop() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance= duration*0.034/2;
    Serial.print("Distance: ");
    Serial.println(distance);
}
```

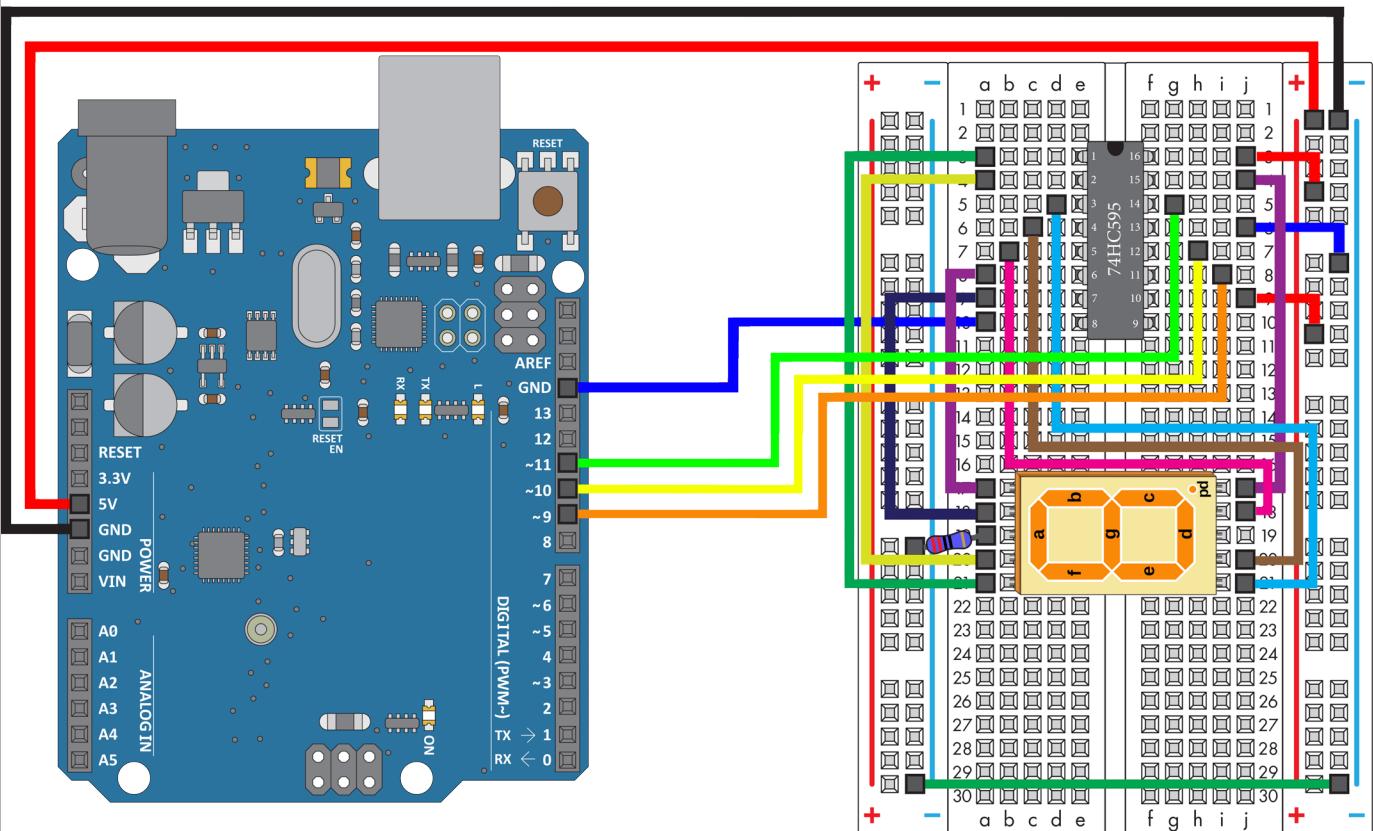


شیفت رجیستر

در پروژه ۲ و پروژه ۸ دیدید که برای راه اندازی مدار، از پین های متعددی در آردوینو استفاده شد.
حال اگر برای پروژه ای به تعداد بیشتری پین نیاز باشد چه باید کرد؟
یکی از راه های حل این مشکل استفاده از شیفت رجیستر است.
نحوه کار شیفت رجیستر در ادامه توضیح داده شده است.
با استفاده از شیفت رجیستر، به جای استفاده از ۸ پین آردوینو، از ۳ پین استفاده می شود.
در این پروژه با نحوه مقداردهی در سیستم های decimal، binary و hexadecimal آشنا خواهید شد.

مطلوبات

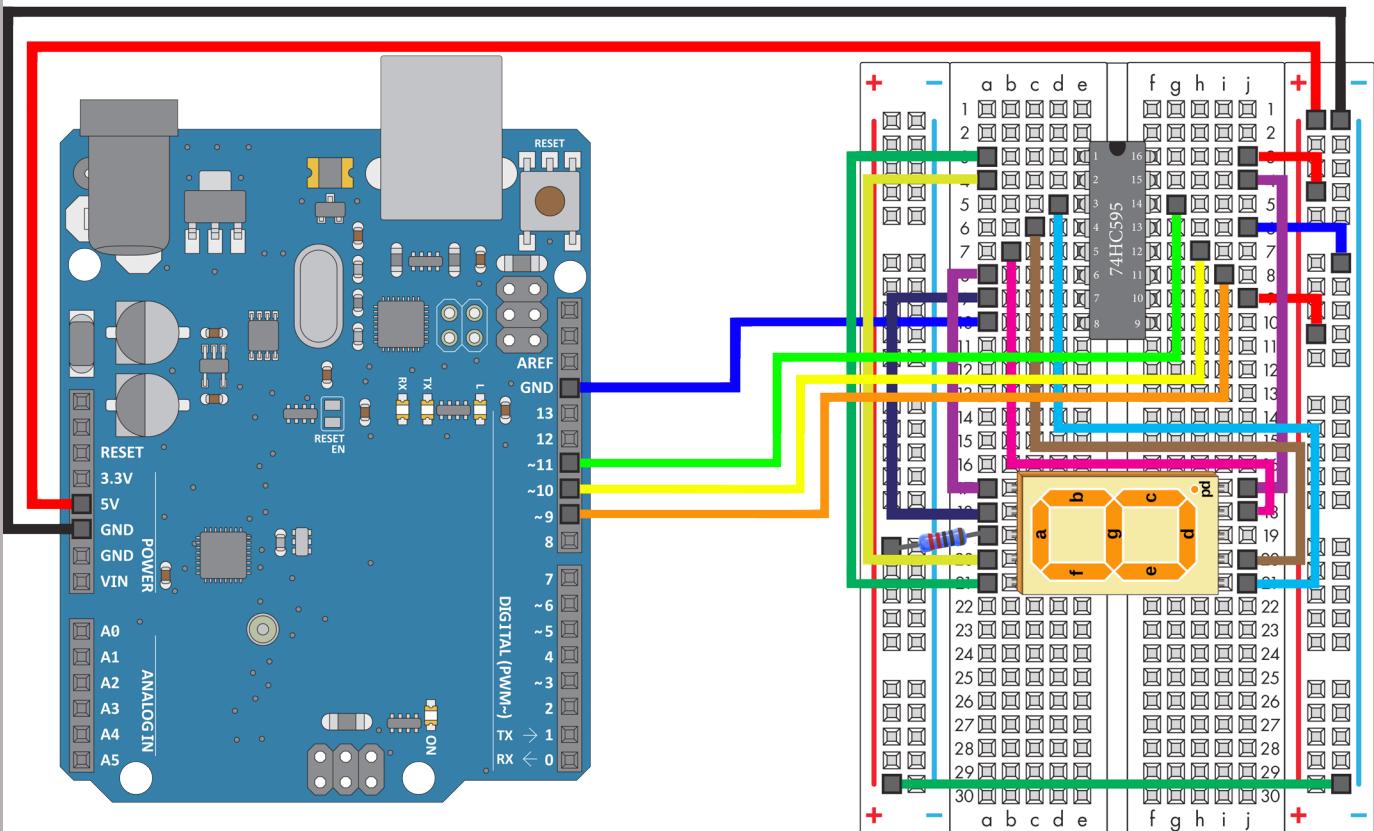




توضیح مدار:

آی سی شیفت رجیستر 74HC595 ۸ امکان کنترل ۸ خروجی توسط ۳ پین اردوینو را فراهم می کند. ۸ خروجی که پایه های ۱-۷ و ۱۵ آی سی هستند، به پایه های a تا pd متصل می شوند. از پین های شماره ۱۱، ۱۰، ۹ آردوینو به پایه های شماره ۱۴، ۱۲ و ۱۱ آی سی متصل می شوند. پایه های ۱۶ و ۱۰ به VCC و پایه های ۸ و ۱۳ به GND وصل می شوند.





توضیح مدار:

مدار پروژه سون سگمنت آند مشترک، مشابه مدار سون سگمنت کاتد مشترک است، با این تفاوت که پایه مشترک سون سگمنت به جای اتصال به GND به 5V متصل می شود.



شیفت رجیستر

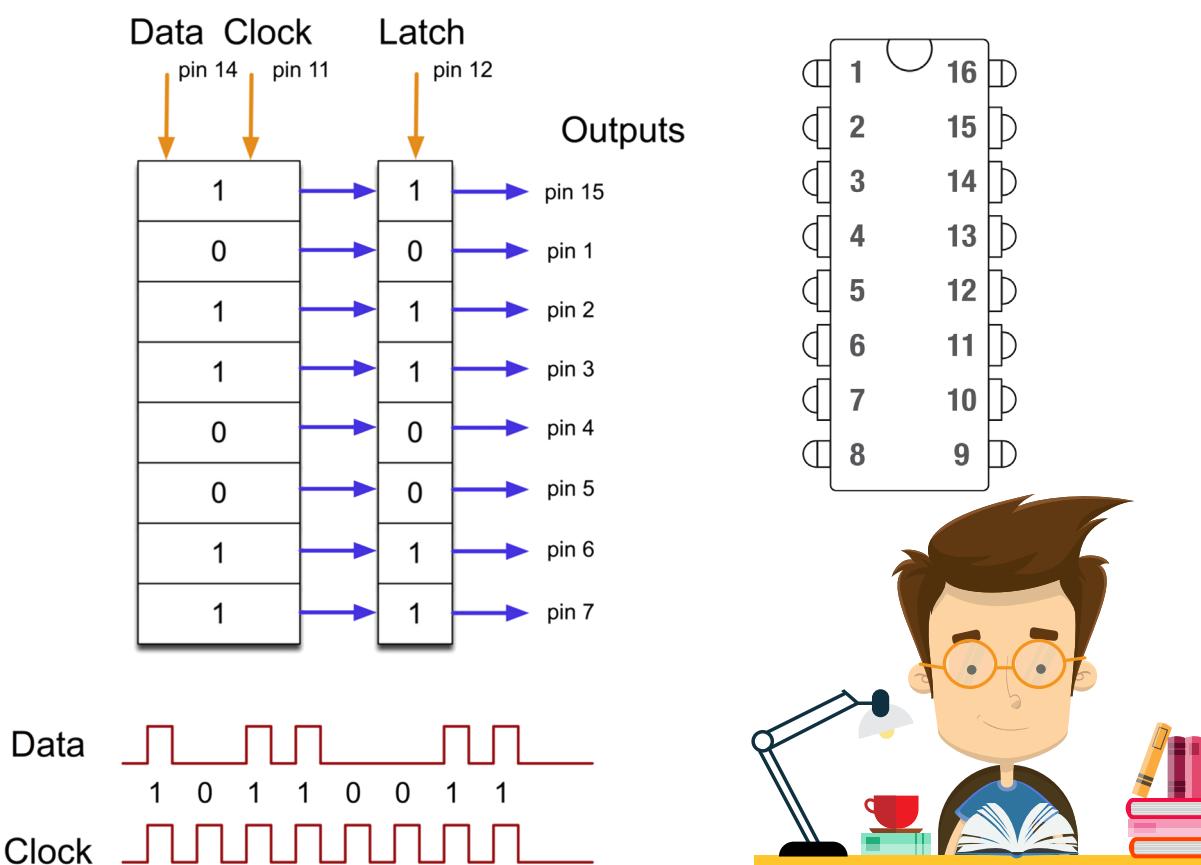
شیفت رجیستر چگونه کار می کند؟

همانطور که در مقدمه گفته شد، برد آردوینو برای پردازش اطلاعات و تصمیم گیری از میکروکنترلر ATMEGA328 از خانواده AVR استفاده می کند. تبادل داده با محیط خارجی، به جز از پین های ورودی / خروجی می تواند از طریق واحد ارتباطی سریال نیز انجام شود. در واقع پین های ورودی و خروجی به صورت موازی و واحدهای ارتباطی سریال به صورت سریال با محیط پیرامون میکروکنترلر در ارتباط هستند.

کوچکترین واحد داده در پردازنده ها "بیت" یا "bit" است. هر 8 بیت یک "بايت" یا "byte" تشکیل می دهد. اطلاعات در ارتباط سریال به صورت بیت های پشت سر هم ارسال و دریافت می شوند. برای کامپیوتر و دیگر پردازنده ها یک بیت به تنها یعنی معنی ندارد و بايت ها با معنی هستند.

شیفت رجیستر این مجموعه از نوع 74HC595 با ظرفیت 8 بیت است که به شیوه ارتباط سریال همزمان (synchronous serial communication) کار می کند. در این شیوه در کنار پین مربوط به داده (Data Pin)، یک پین دیگر به نام ساعت (Clock Pin) وجود دارد. همزمان با ارسال داده از پین Data به صورت ۰ یا ۱، یک پالس توسط پین Clock ارسال می شود. وضعیت پین Data در زمان هر پالس پین Clock ذخیره می شود.

برای دریافت داده، پین Latch در وضعیت **LOW** قرار داده می شود. در این حالت شیفت رجیستر متوجه دریافت اطلاعات است. بعد از دریافت ۸ پالس (۸ بیت - ۱ بايت)، پین Latch در وضعیت **HIGH** قرار می گیرد و ۸ بیت دریافت شده مطابق شکل به پین ها نوشته می شود. **دقت داشته باشید که شماره پین های نوشته شده در تصویر زیر، پین های شیفت رجیستر است نه پین های آردوینو.**



حال که با منطق و نحوه عملکرد شیفت رجیستر آشنا شدید، نحوه کدنویسی و ارسال داده به شیفت رجیستر توضیح داده می شود.

نحوه تعریف پین ها مشابه پروژه های پیش است:

```
int latchPin = 11;
int dataPin = 10;
int clockPin = 9;
```

متغیر byte

می توان برای ارسال اطلاعات به شیفت رجیستر از آرایه و حلقه برای ارسال بیت به بیت استفاده کرد و مشابه پروژه قبل اطلاعات سگمنت ها را جدا جدا ارسال کرد؛ ولی در این پروژه از متغیر byte برای ارسال یکجای وضعیت سگمنت استفاده می شود. همانگونه که پیشتر گفته شد، یک بایت از ۸ بیت تشکیل شده که هر کدام از این بیت ها می توانند مقادیر ۰ یا ۱ داشته باشند. با توجه به اینکه پین های دیجیتال دو وضعیت **LOW/HIGH** دارند، آردوینو عدد ۰ را به معنای وضعیت **LOW** و عدد ۱ را به معنای وضعیت **HIGH** در نظر می گیرد. در هنگام تعریف، با ۰ قرار دادن این متغیر، تمامی بیت ها برابر با ۰ (به معنی خاموش بودن تمام سگمنت ها) می شوند.

```
byte segment = 0;
```

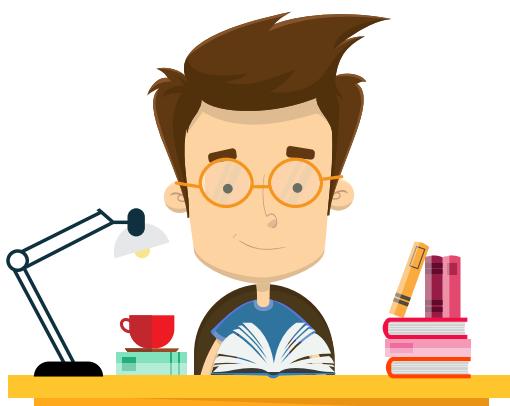
تابع **()ShiftOut**

برای استفاده از شیفت رجیستر، از تابع **ShiftOut** استفاده می شود. این تابع ۴ پارامتر می گیرد: **ShiftOut** (**dataPin**, **clockPin**, **LSBFIRST**, **segment**);

پارامترهای اول و دوم، پین های مربوط به **clock** و **data** هستند.

پارامتر سوم تعیین می کند که بیت های دریافتی، از کدام سمت مورد استفاده قرار گیرد. **LSBFIRST** که مخفف Least Significant Bit First و به معنای ابتداء کم ارزش ترین بیت است (بیت سمت راست ضریب ۲^۰ است) تعیین می کند که آخرین بیت (بیت سمت راست) ابتداء مورد استفاده قرار گیرد. اگر پارامتر سوم **MSBFIRST** (مخفف Most Significant Bit First) به معنای ابتداء بالا ارزش ترین بیت) باشد، اولین بیت مورد استفاده بیت سمت چپ است (بیت سمت چپ ضریب ۷^۰ برابر با ۱۲۸ است).

پارامتر چهارم داده ای است که باید به شیفت رجیستر فرستاده شود که در اینجا بایت segment است.



شیفت رجیستر

استفاده از متغیر byte مزایایی دارد که از مهمترین آنها استفاده از مقدار معادل عددی وضعیت سگمنت‌ها است. فرض کنید که سون سگمنت باید عدد ۸ و نقطه اعشار را نمایش دهد، بدین صورت وضعیت سگمنت‌ها به صورت زیر است (**HIGH** برابر با ۱ و **LOW** برابر با صفر در نظر گرفته شده است):

a	b	c	d	e	f	g	pd
۱	۱	۱	۱	۱	۱	۱	۱
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$							
F				F			

وضعیت فوق به صورت عدد مبنای ۱۰ یا decimal خواهد بود:
segment=255;

وضعیت فوق به صورت یک عدد در مبنای ۲ یا سیستم binary برابر با ۱۱۱۱۱۱۱۱ خواهد بود. برای اینکه آردوینو بداند که یک عدد به صورت binary است باید بدین صورت مقدار دهی شود:
segment = B11111111;

گاهی اوقات از سیستم مبنای ۱۶ یا hexadecimal و یا hex استفاده می‌شود. در مبنای ۱۶ از اعداد ۰-۹ و حروف A-F استفاده می‌شود. بدین صورت:

A	B	C	D	E	F
۱۰	۱۱	۱۲	۱۳	۱۴	۱۵

برای تبدیل عدد از مبنای ۲ به ۱۶، ۱۶ رقم ۴ رقم جدا کرده و عدد یا حرف معادل آن ۴ رقم نوشته می‌شود. بدین صورت وضعیت فوق برابر با FF خواهد بود. برای اینکه آردوینو بداند عدد در مبنای ۱۶ یا سیستم Hex است:

segment = 0xFF;

در ادامه اعداد متناظر با وضعیت سون سگمنت در حالت MSBFIRST ارائه شده است.

دقت داشته باشید که این مقادیر، متناظر با مدار این پروژه است. در صورتی که به طریق دیگر سگمنت‌ها به پایه‌های شیفت رجیستر متصل شده باشد، مقادیر باید با توجه به پین‌های آن مدار محاسبه شود.



	pd	g	f	e	d	c	b	a	binary	decimal	hex
0	0	0	1	1	1	1	1	1	B00111111	63	0x3F
1	0	0	0	0	0	1	1	0	B00000110	6	0x06
2	0	1	0	1	1	0	1	1	B01011011	91	0x5B
3	0	1	0	0	1	1	1	1	B01001111	79	0x4F
4	0	1	1	0	0	1	1	0	B01100110	102	0x66
5	0	1	1	0	1	1	0	1	B01101101	109	0x6D
6	0	1	1	1	1	1	0	1	B01111101	125	0x7D
7	0	0	0	0	0	1	1	1	B00000111	7	0x07
8	0	1	1	1	1	1	1	1	B01111111	127	0x7F
9	0	1	1	0	1	1	1	1	B01101111	111	0x6F
A	0	1	1	1	0	1	1	1	B01110111	119	0x77
b	0	1	1	1	1	1	0	0	B01111100	124	0x7C
C	0	0	1	1	1	0	0	1	B00111001	57	0x39
d	0	1	0	1	1	1	1	0	B01011110	94	0x5E
E	0	1	1	1	1	0	0	1	B01111001	121	0x79
F	0	1	1	1	0	0	0	1	B01110001	113	0x71
H	0	1	1	1	0	1	1	0	B01110110	118	0x76
J	0	0	0	1	1	1	1	0	B00011110	30	0x1E
L	0	0	1	1	1	0	0	0	B00111000	56	0x38
n	0	1	0	1	0	1	0	0	B01010100	84	0x54
o	0	1	0	1	1	1	0	0	B01011100	92	0x5C
P	0	1	1	1	0	0	1	1	B01110011	115	0x73
U	0	0	1	1	1	1	1	0	B00111110	62	0x3E

مقادیر بالا با در نظر گرفتن خاموش بودن نقطه اعشار و
حالت MSBFIRST در تابع ShiftOut برای سون سگمنت کاتد
مشترک محاسبه شده است. شما می توانید با روش توضیح
داده شده، مقادیر مربوط به روشن بودن اعشار و LSBFIRST
را محاسبه کنید.

با توجه به اینکه بیت مربوط به نقطه اعشار (pd) ضریب 2^7
یعنی 128 است، با افزودن عدد 128 به مقادیر decimal
جدول بالا، مقادیر مربوط به روشن بودن نقطه اعشار بدست
خواهد آمد.



شیفت رجیستر

یکی از مزایای استفاده از شیفت رجیستر و متغیر byte، ساده سازی کدنویسی است، بدین صورت که بدون نیاز به نوشتن تابع برای هر حالت از سون سگمنت با استفاده از تعریف وضعیت تک تک سگمنت ها، با استفاده از یک عدد می توان خروجی مورد نظر را ایجاد کرد. به عنوان مثال عدد ۶۳ به تنها بی به معنای روشن شدن سگمنت های a, b, c, d, e, f و نمایش عدد ۰ بر روی سون سگمنت است. همچنین در صورتی که نیاز به روشن کردن سگمنت خاصی باشد، به راحتی با استفاده از عدد متناظر با آن سگمنت می توان آنرا روشن کرد. اعداد مربوط به هر کدام از سگمنت ها (با توجه به مدار این پروژه و حالت MSBFIRST) عبارتند از:

a	b	c	d	e	f	g	pd
۱	۲	۴	۸	۱۶	۳۲	۶۴	۱۲۸

به عنوان مثال اگر بخواهید که پیش از نمایش اعداد هر کدام از سگمنت ها به ترتیب روشن و خاموش شوند، به سادگی می توان با استفاده از عدد مربوط به آن ها این کار را انجام داد. در ادامه خواهید دید که کد مربوط به پروژه ۸ به چه سادگی در این پروژه خلاصه شده است.

سون سگمنت آند مشترک

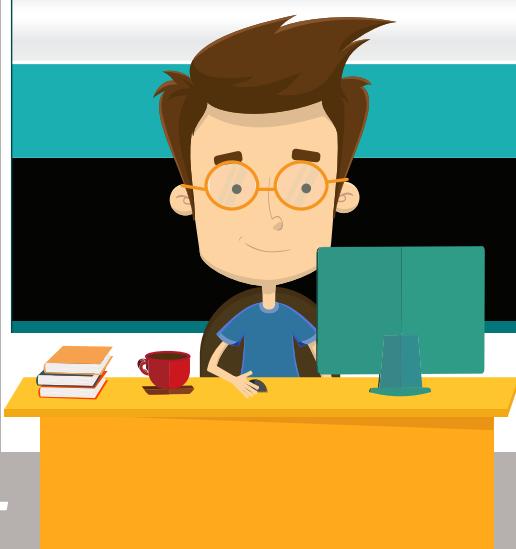
برای استفاده از سون سگمنت آند مشترک، به جای اینکه برای سگمنت های روشن وضعیت ۱ منظور شود، وضعیت ۰ در نظر گرفته می شود. به عنوان مثال در حالی که برای روشن کردن سگمنت a در سون سگمنت کاتد مشترک از عدد ۱ استفاده می شود، در سون سگمنت آند مشترک برای روشن کردن سگمنت a از عدد ۲۵۴ استفاده می شود یا در صورت ارسال عدد ۰ به سون سگمنت کاتد مشترک، همه سگمنت ها خاموش و در صورت فرستادن عدد ۱ به سون سگمنت کاتد مشترک، همه سگمنت ها روشن می شوند.

در نتیجه برای تبدیل کد سون سگمنت کاتد مشترک به آند مشترک، کافی است عدد کاتد مشترک را از ۲۵۵ کسر کنید.

همچنین، برخلاف نوع کاتد مشترک که برای روشن کردن نقطه اعشار، عدد ۱۲۸ به عدد حالت افزوده می شد، در نوع آند مشترک برای روشن کردن نقطه اعشار، عدد ۱ از عدد حالت کسر می شود. مجددا خاطر نشان می شود اعداد ارائه شده در جدول مربوط به مدار این پروژه است و در صورت تفاوت اتصال پایه ها می بایست مجددا عدد ها محاسبه شوند.



```
int dataPin = 11;
int latchPin = 10;
int clockPin = 9;
byte segment = 0;
int i;
int data[]={ 63, 6, 91, 79, 102, 109, 125, 7, 127, 111, 119, 124, 88, 57, 94, 121, 113,
116, 118, 30, 56, 84, 92, 115, 103, 80, 28, 62};
void setup() {
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}
void loop() {
    for (i=0; i<28; i++) {
        segment=data[i];
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST, segment);
        digitalWrite(latchPin, HIGH);
        delay(1000);
    }
}
```



File > Examples > Electroduino > Circuit_27_2

```
int dataPin = 11;
int latchPin = 10;
int clockPin = 9;
byte segment = 0;
int i;
int data[]={ 192, 249, 164, 176, 153, 146, 130, 248, 128, 144, 136, 131, 167, 198, 161,
134, 142, 139, 137, 225, 199, 171, 163, 140, 152, 175, 227, 193};
void setup() {
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}
void loop() {
    for (i=0; i<28; i++) {
        segment=data[i];
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST, segment);
        digitalWrite(latchPin, HIGH);
        delay(1000);
    }
}
```



سون سگمنت چهارتایی

۲۰

در پروژه ۸ دیدید که پایه مشترک (آند یا کاتد) به پین **LOW** یا **HIGH** متصل شده و با اعمال سیگنال به پایه های مربوط به سگمنت ها، اعداد یا حروف نشان داده می شد.

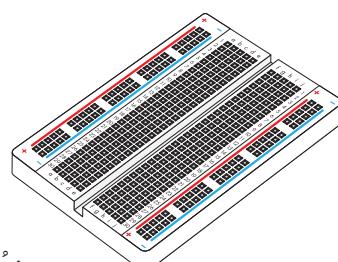
اصل کاری سون سگمنت چهارتایی مشابه سون سگمنت تکی است، با این تفاوت که به جای یک پایه مشترک، چهار پایه مشترک (آند یا کاتد) برای هر کدام از چهار موقعیت دارد.

در واقع در لحظه تنها یک عدد روی سون سگمنت ها نشان داده می شود.. با توجه به سرعت بالای اجرای برنامه توسط آردوبینو، اگر پایه های مشترک با سرعت بالا روشن و خاموش شوند، چشم انسان متوجه خاموش و روشن شدن نشده و هر چهار سون سگمنت را در حال نمایش اعداد یا حروف می بیند.

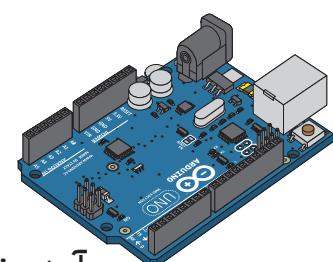
مطلوبات



سیم جامپر نری-نری
(۲۱ عدد)



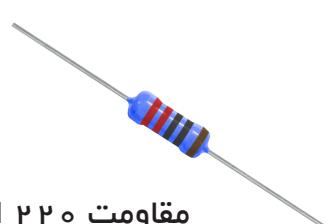
بردبرد



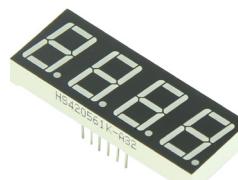
آردوبینو



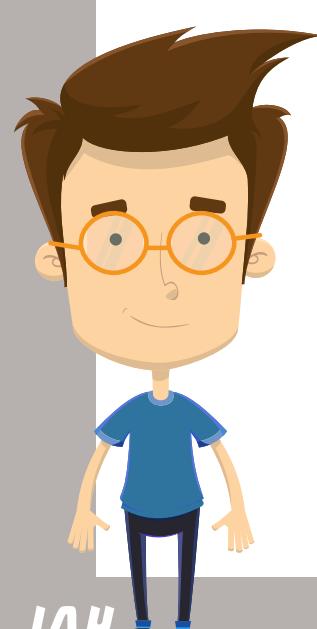
شیفت رجیستر
(۱ عدد)

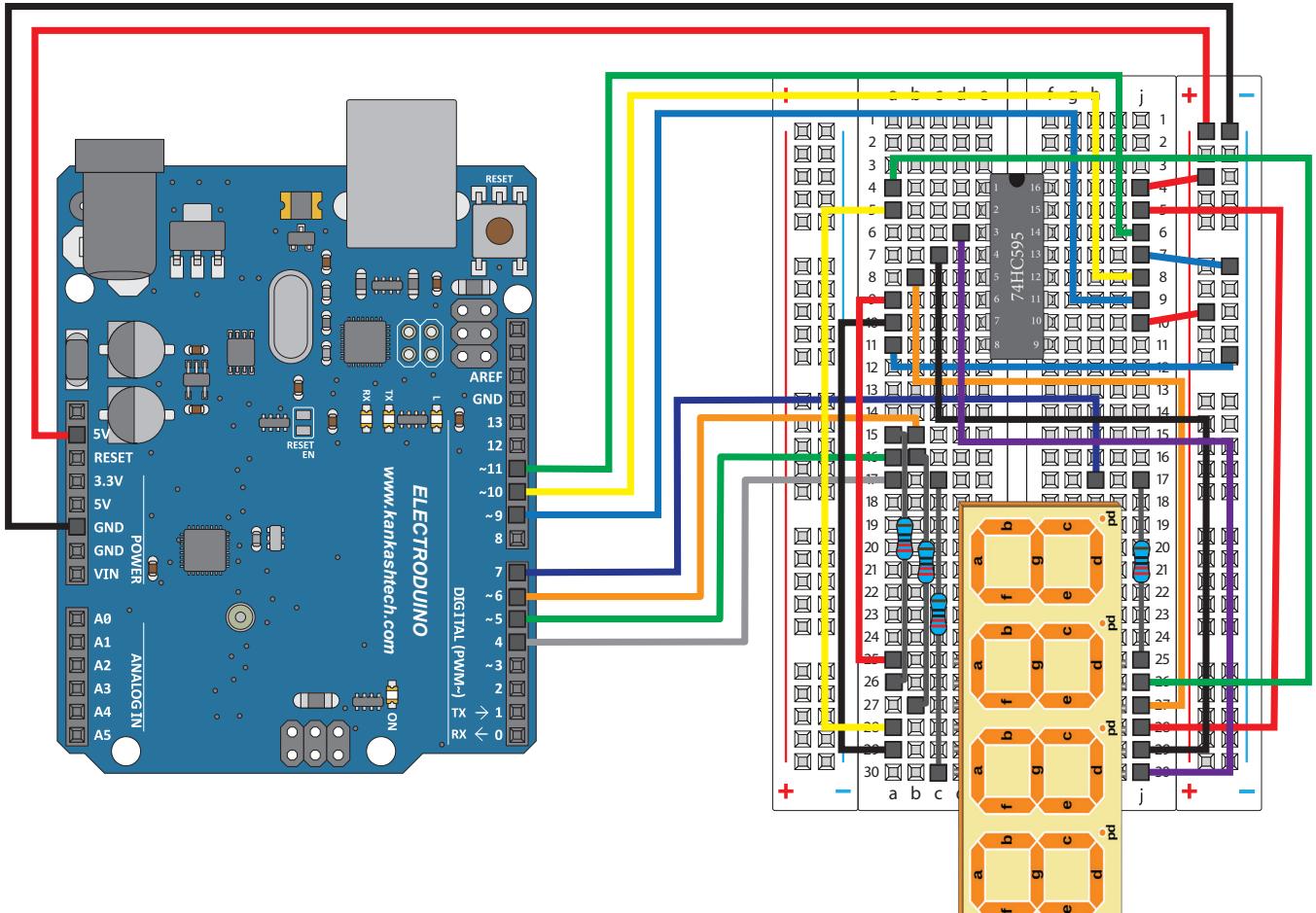


مقاومت ۲۲۰ اهم
(۴ عدد)



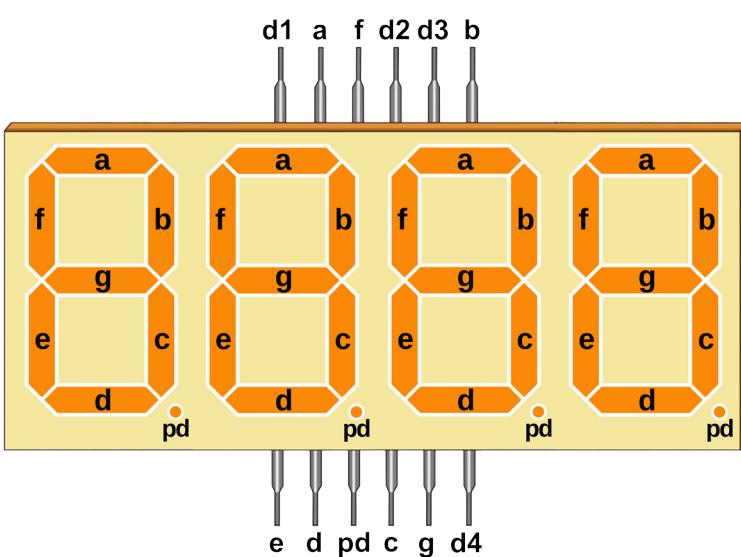
سون سگمنت
(۱ عدد)





توضیح مدار:

مدار این پروژه مشابه پروژه ۲۷ است؛ با این تفاوت که به جای اتصال کاتد مشترک به GND، هر کدام از کاتدهای مشترک d1-d4 به یکی از پین های دیجیتال ۷-۴ متصل شده و با LOW کردن آن پین، سون سگمنت متناظر با آن پین روشن می شود.



همانگونه که در مقدمه این پروژه گفته شد، برای نمایش هر کدام از چهار سون سگمنت، باید پایه کاتد مشترک مربوط به آن عدد در وضعیت **LOW** قرار گیرد. پایه های مربوط به هر کدام از سگمنت های a-pd برای هر چهار موقعیت یکی است. در واقع در یک لحظه تنها یک عدد می تواند نمایش داده شود. اگر هر ۴ کاتد مشترک به صورت همزمان در وضعیت **LOW** قرار گیرند، هر چهار موقعیت یک رقم را نشان خواهند داد. ولی در صورتی که با سرعت زیاد پایه کاتد مشترک **LOW** جابجا شود، چشم انسان نمی تواند خاموش بودن سگمنت ها را تشخیص داده و می توان در لحظه تنها یک سگمنت روشن بوده و مقدار متمایزی به آن سگمنت اعمال کرد.

برای انجام این موضوع، ۴ کار باید انجام شود:

۱. ابتدا همه سگمنت ها خاموش شود.
۲. موقعیت عدد اول انتخاب شده و کاتد مشترک آن موقعیت، در وضعیت **LOW** قرار گیرد
۳. رقمی که باید در این موقعیت نمایش داده شود تعیین شود.
۴. به مدت کوتاه (۵ میلی ثانیه) توقف انجام شود.

سپس این کار برای موقعیت های ۲ تا ۴ تکرار شود. با اجرای پیوسته موارد فوق، هر چهار موقعیت در حال نمایش عدد دیده می شود. تعریف پایه ها و متغیرها مشابه پروژه ۹ است با این تفاوت که پین های ۴-۷ نیز برای موقعیت سگمنت ها استفاده می شوند.

برای نمایش عدد، باید از هر کدام از رقم های عدد را مشخص کرد. اگر ۶ یک عدد چهار رقمی از نوع int باشد، تفکیک رقم ها به صورت زیر انجام می شود:

۱. عدد یکان: باقیمانده تقسیم n بر 10 : $(n\%10)$
۲. عدد دهگان: ابتدا باقیمانده تقسیم n بر 100 تعیین می شود. خارج قسمت تقسیم این عدد دو رقمی بر 10 ، عدد مربوط به دهگان خواهد بود: $(n\%100)/10$
۳. عدد صدگان: ابتدا باقیمانده تقسیم n بر 1000 تعیین می شود. خارج قسمت تقسیم این عدد 3 رقمی بر 100 ، عدد مربوط به صدگان خواهد بود: $(n\%1000)/100$
۴. عدد هزارگان: خارج قسمت تقسیم n بر 1000 ، عدد مربوط به هزارگان خواهد بود: $(n/1000)$



گاهی اوقات نیاز است که در کنار تابع `loop` که به طور پیوسته در حال تکرار است، یک تابع دیگر نیز بدون وابستگی به سرعت اجرای تابع `loop` اجرا شود.

به عنوان مثال اگر شما بخواهید از سون سگمنت چهارتایی استفاده کنید که یک ثانیه شمار را نشان دهد، اگر در تابع `loop` از دستور `delay` در حلقه افزایش یک متغیر در ثانیه استفاده شود، اجرای برنامه با وقفه های ۱ ثانیه ای مواجه شده و در نتیجه فقط یکی از چهار سون سگمنت روشن دیده می شود اگر این افزایش در تابع دیگری نوشته شده و در هر بار اجرای تابع `loop` فرآخوانی شود، افزایش با سرعت زیادی انجام می شود و نمی توان از آن به عنوان ثانیه شمار استفاده کرد. برای حل این مشکل از تابع `Timer1` استفاده می شود. با استفاده از تابع `Timer1` می توان در بازه های مشخص، اجرای برنامه را متوقف کرده و یک تابع خاص را اجرا کرد.

Timer1

برای استفاده از این دستور، در ابتدای کد، باید کتابخانه مربوط به آن فرآخوانی شود:

```
#include <TimerOne.h>
```

برای اینکه این دستور عمل کند، شما باید مراحل نصب کدهای راهنمای کتابخانه ها که در بخش مقدمه توضیح داده شده است را انجام داده باشید. پس از فرآخوانی کتابخانه، باید فاصله زمانی اجرای دستور توقف و تابعی که در زمان توقف اجرا می شود را مشخص کنید:

```
Timer1.initialize();
```

عدد استفاده شده در دستور فوق در واحد میکروثانیه است. در حالت پیش فرض مقدار فاصله زمانی، ۱ ثانیه است. می توان در صورت نیاز فاصله زمانی را مقادیر دلخواه تنظیم کرد. در حالت بالا هر ۱ ثانیه، دستورات متوقف و تابع تعیین شده اجرا می شود:

```
Timer1.attachInterrupt( add );
```

تابعی که در هنگام وقفه اجرا می شود تابع `(add)` است.

```
void add() {
    n++;
    if(n == 10000){
        n = 0;
    }
}
```





Circuit_28

```
#include <TimerOne.h>
int latchPin = 10;
int dataPin = 11;
int clockPin = 9;
int d4 = 7;
int d3 = 6;
int d2 = 5;
int d1 = 4;
byte segment = 0;
int n = 0;
void setup() {
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    pinMode(d1, OUTPUT);
    pinMode(d2, OUTPUT);
    pinMode(d3, OUTPUT);
    pinMode(d4, OUTPUT);
    Timer1.initialize(1000);
    Timer1.attachInterrupt( add );
}
```



File > Examples > Electroduino > Circuit _28



Circuit_28

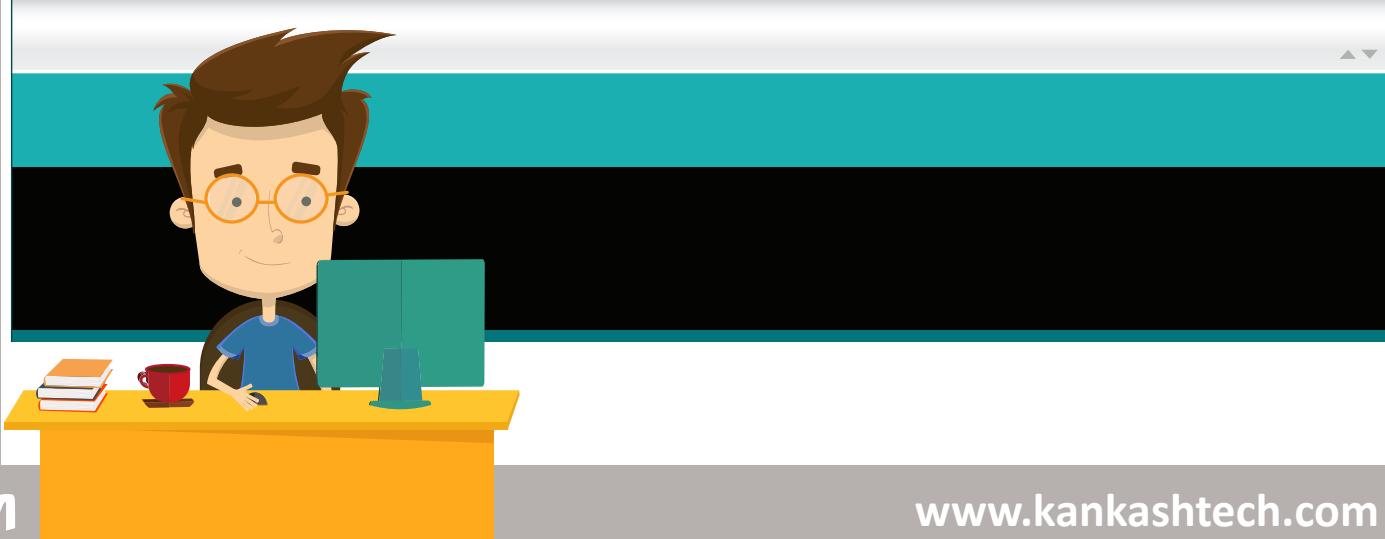
```
void loop() {
    clearLEDs();
    pickDigit(0);
    pickNumber((n/1000));
    delay(5);
    clearLEDs();
    pickDigit(1);
    pickNumber((n%1000)/100);
    delay(5);
    clearLEDs();
    pickDigit(2);
    pickNumber(n%100/10);
    delay(5);
    clearLEDs();
    pickDigit(3);
    pickNumber(n%10);
    delay(5);
}
void clearLEDs() {
    segment=0;
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, segment);
    digitalWrite(latchPin, HIGH);
}
```



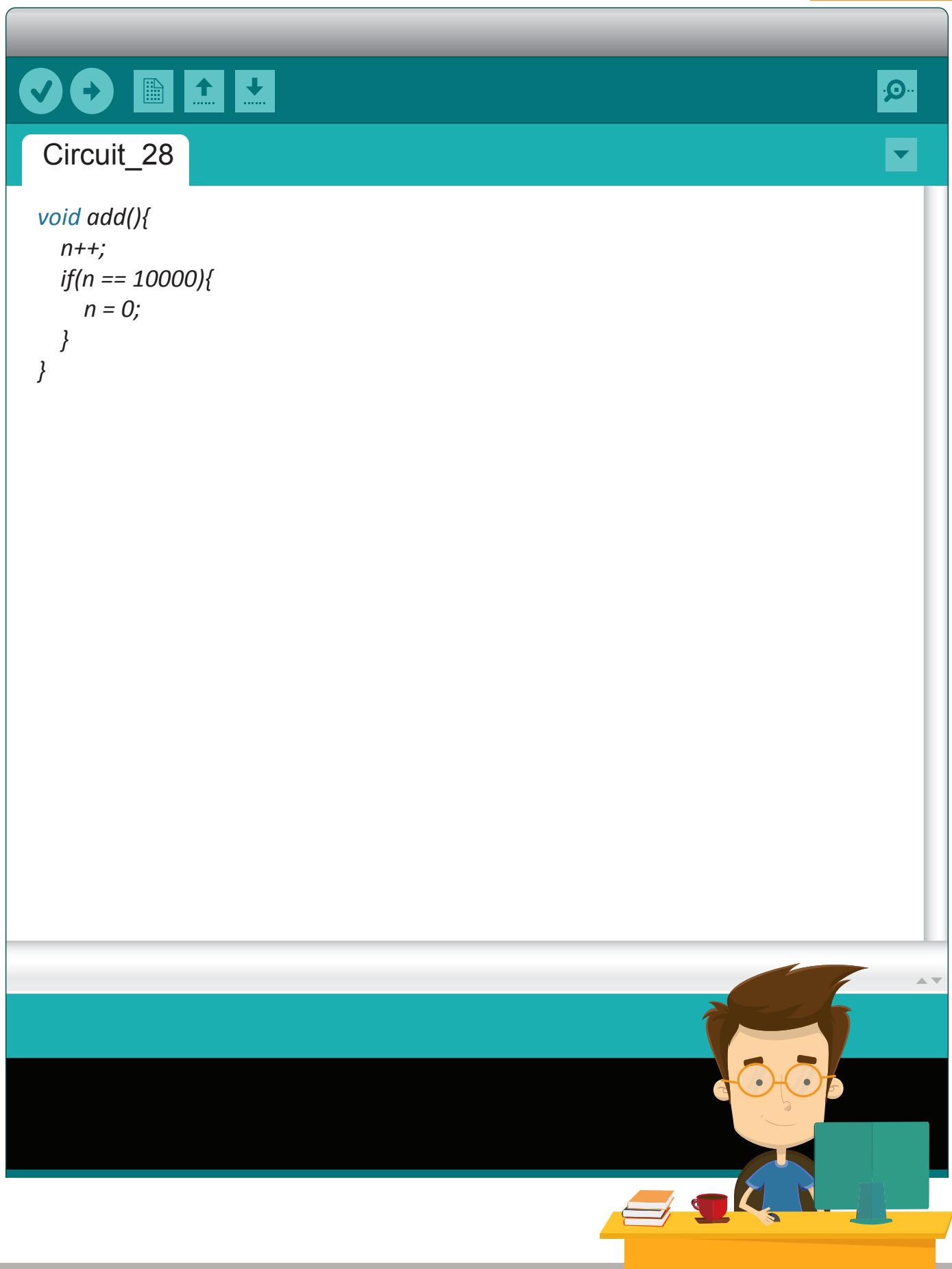
```
Circuit_28

void pickDigit(int x) {
    int digit[] = {d1, d2, d3, d4};
    for (int j=0; j<4; j++){
        if (j==x){
            digitalWrite(digit[j], LOW);
        }
        else {
            digitalWrite(digit[j], HIGH);
        }
    }
}

void pickNumber(int x){
    int data[] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};
    segment = data[x];
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, segment);
    digitalWrite(latchPin, HIGH);
}
```



File > Examples > Electroduino > Circuit _28



```
void add(){
    n++;
    if(n == 10000){
        n = 0;
    }
}
```

ماژول TM1637

۳۹

ماژول TM1637 یک سون سگمنت چهارتایی آند مشترک با درایور TM1637 است. برای کنترل چهار موقعیت این ماژول، به جز VCC و GND، تنها به دو پین data و clock نیاز است. با توجه به این موضوع یعنی به کارگیری تنها دو پین از آردوینو، ماژول بسیار مناسبی برای ساخت سیستم های با قابلیت نمایش عدد چهار رقمی است.

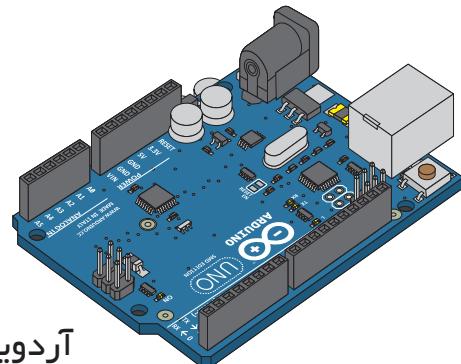
همچنین با داشتن دو نقطه در میان موقعیت های دوم و سوم، می توان برای نمایش زمان یا کاربردهای تایмер از آن استفاده کرد.

در این پروژه به جای نوشتن توابع، با فراخوانی کتابخانه اختصاصی این ماژول، از دستورات پیش فرض آن استفاده می شود.

مطلوبات



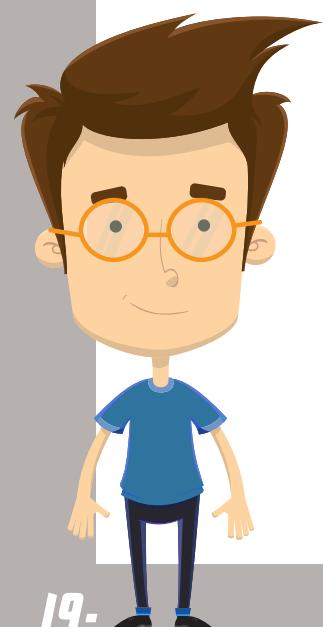
سیم جامپر نری-مادگی
(۴ عدد)

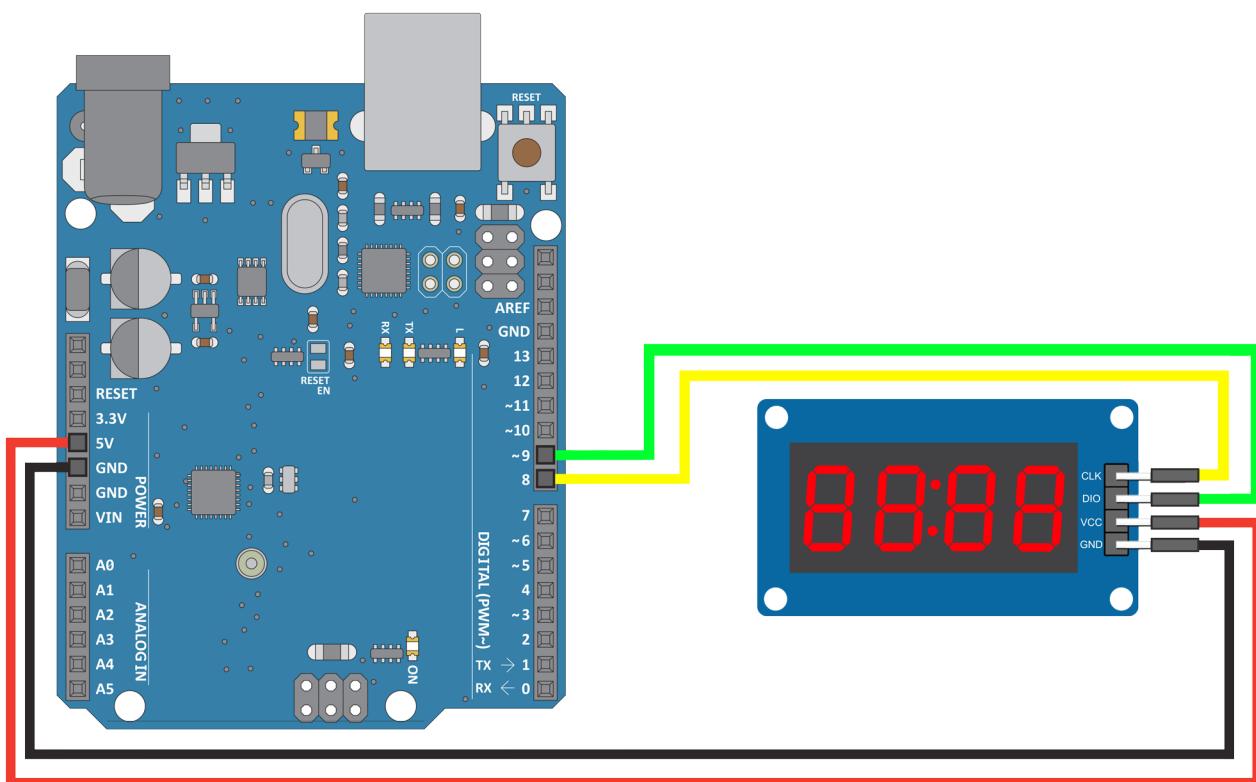


آردوینو



نمایشگر TM1637
(۱ عدد)





توضیح مدار:

مدار این پروژه بسیار ساده است. پایه های VCC و GND مازول به ترتیب به پین های ۵ ولت و GND آردوینو متصل می شود. پایه CLK به پین دیجیتال ۸ و پایه DIO که پایه data است به پین دیجیتال ۹ آردوینو متصل می شود.



برای استفاده از ماژول TM1637 ابتدا باید کتابخانه این ماژول فراخوانده شود:

```
#include <TM1637Display.h>
```

سپس پین های مربوط به CLK و DIO تعریف شود:

```
int CLK= 8;  
int DIO= 9;
```

حال باید ماژول به عنوان یک نمایشگر از نوع TM1637 تعریف شود:

```
TM1637Display myTM(CLK, DIO);
```

برای استفاده از این ماژول از یک آرایه ۴ جزئی از نوع بایت استفاده می شود. دقت داشته باشید که سگمنت های این ماژول نقطه اعشار ندارند. لذا حداقل مقدار آنها (برای نمایش عدد ۸) برابر با ۱۲۷ خواهد بود.

البته دو نقطه میان موقعیت های دوم و سوم، مانند نقطه اعشار سگمنت های پروژه های قبل است که به سون سگمنت موقعیت دوم متصل است. لذا عدد ۲۵۵ برای موقعیت دوم به معنای نمایش عدد ۸ و روشن بودن دو نقطه خواهد بود. در تعریف اولیه، مقادیر متناظر با صفر و روشن بودن دو نقطه وسط استفاده می شود (برای مرور، به پروژه ۲۷ مراجعه نمایید):

```
byte data[] = { 63, 191, 63 , 63 };
```

هر کدام از آیتم های آرایه data یک بایت هستند.
برای شروع به کار با ماژول، باید میزان نور آن تعیین شود. میزان نور بین ۰-۷ خواهد بود به صورتی که ۰ به معنای سگمنت های خاموش و ۷ به معنای حداقل درخشندگی سگمنت هاست.
myTM.setBrightness(7);

برای نمایش عدد بر روی ماژول، از دستور **setSegments** استفاده می شود.
myTM.setSegments(data);

البته می توان با استفاده از دستور **showNumberDec** مستقیما عدد مورد نظر را بر روی ماژول نمایش داد:
myTM.showNumberDec(4559, false, 2, 0)

با توجه به اینکه این دستور، امکان روشن و خاموش کردن دو نقطه وسط را ندارد، در این پروژه از دستور **setSegments** استفاده می شود



در این پروژه از مازول به عنوان یک ثانیه شمار استفاده خواهد شد. بدین صورت که از لحظه ۰ در هر ثانیه یک واحد به رقم چهارم (اولین رقم از سمت راست) افزوده خواهد شد. با رسیدن رقم چهارم به ۱۵، این رقم صفر شده و یکی به رقم سوم افزوده خواهد شد.
با رسیدن رقم سوم به عدد ۶، این رقم صفر شده و یکی به رقم دوم افزوده می شود. همین اتفاق برای رقم های دوم و اول نیز تکرار می شود.

برای چشمک زدن دو نقطه وسط، در زمانی که عدد رقم چهارم زوج است (باقیمانده تقسیم رقم چهارم بر ۲ (یا ۱۰٪۲) برابر ۰ است)، ۱۲۸ واحد به مقدار رقم دوم افزوده می شود. در این حالت ضمن حفظ مقدار نمایش داده شده در رقم دوم، نقطه های وسط نیز روشن می شوند. ۱ ثانیه بعد، عدد رقم چهارم فرد شده و نقطه ها خاموش می شوند.

برای افزایش رقم ها از کد زیر استفاده می شود:

```
if (n4==10) {
    n4=0;
    n3++;
    if (n3==6) {
        n3=0;
        n2++;
        if (n2==10){
            n2=0;
            n1++;
            if (n1==6) {
                n1=0;
            }
        }
    }
}
```



File > Examples > Electroduino > Circuit _29

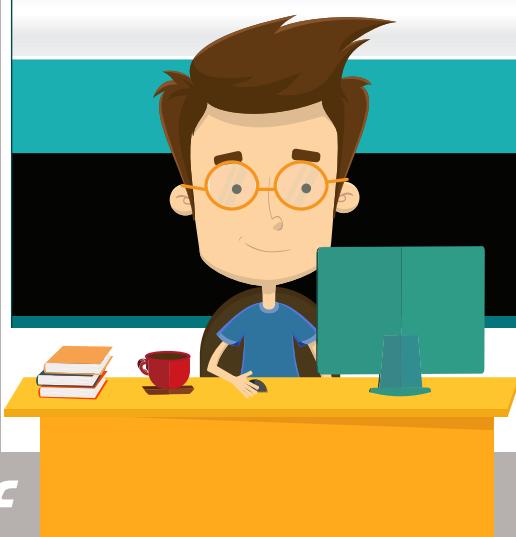


Circuit_29



```
#include <TM1637Display.h>
int CLK= 8;
int DIO= 9;
TM1637Display myTM(CLK, DIO);
byte data[] = { 63, 63, 63 , 63 };
int n1 = 0;
int n2 = 0;
int n3 = 0;
int n4 = 0;

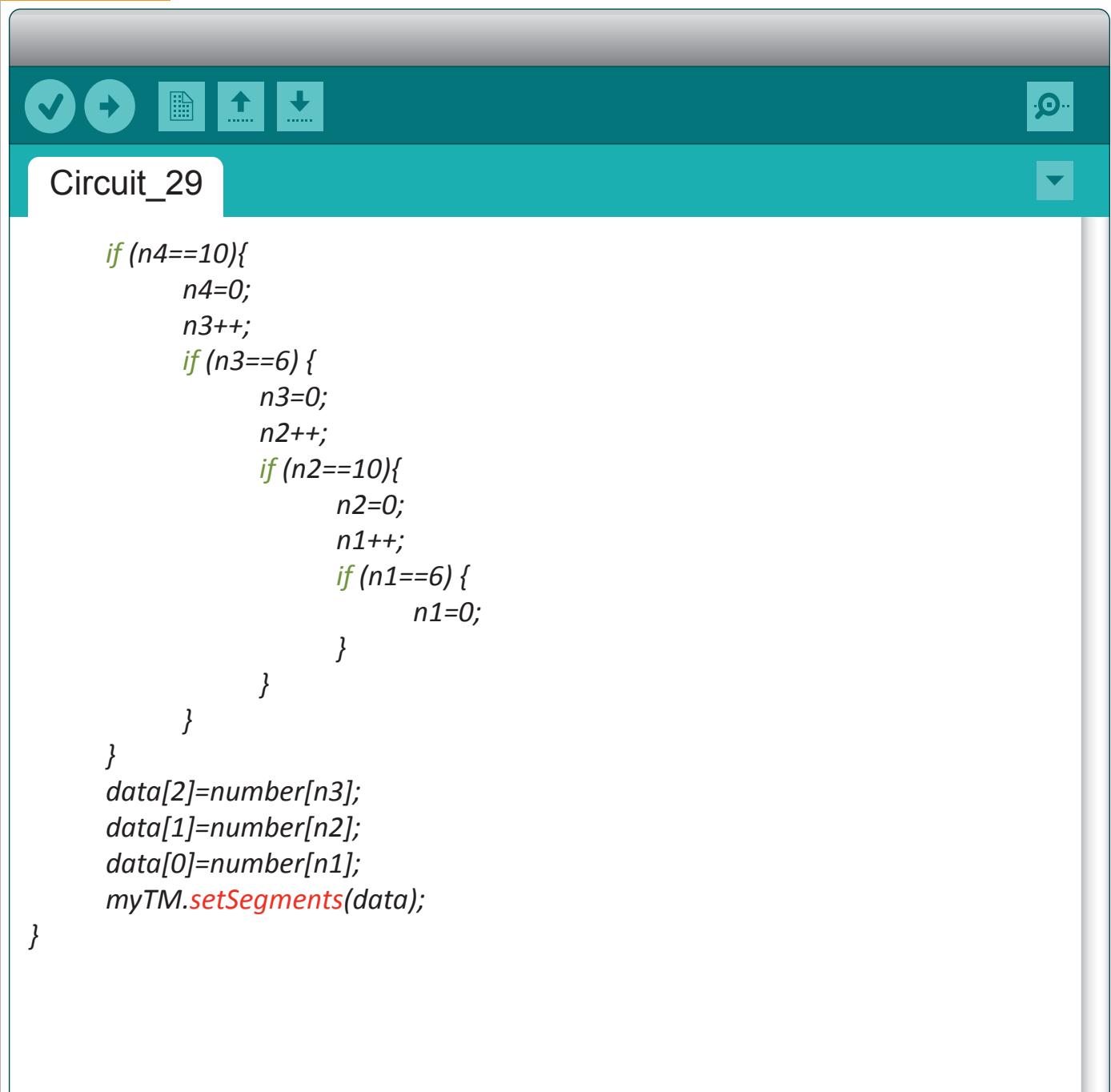
void setup() {
    myTM.setBrightness(7);
    myTM.showNumberDec(0 , true, 4, 0);
    delay (2000);
    myTM.showNumberDec(10, false, 4, 0);
    delay (2000);
    myTM.showNumberDec(100, false, 4, 0);
    delay (2000);
    myTM.showNumberDec(1000, false, 4, 0);
    delay (2000);
}
```



File > Examples > Electroduino > Circuit _29

```
void loop() {
    myTM.setBrightness(7);
    int number[]={63, 6, 91, 79, 102, 109, 125, 7, 127, 111};
    for (n4=0; n4<10; n4++) {
        data[3]=number[n4];
        int mod=n4%2;
        if (mod==0) {
            data[1]=number[n2];
        }
        else {
            data[1]=number[n2]+128;
        }
        myTM.setSegments(data);
        delay(1000);
    }
}
```

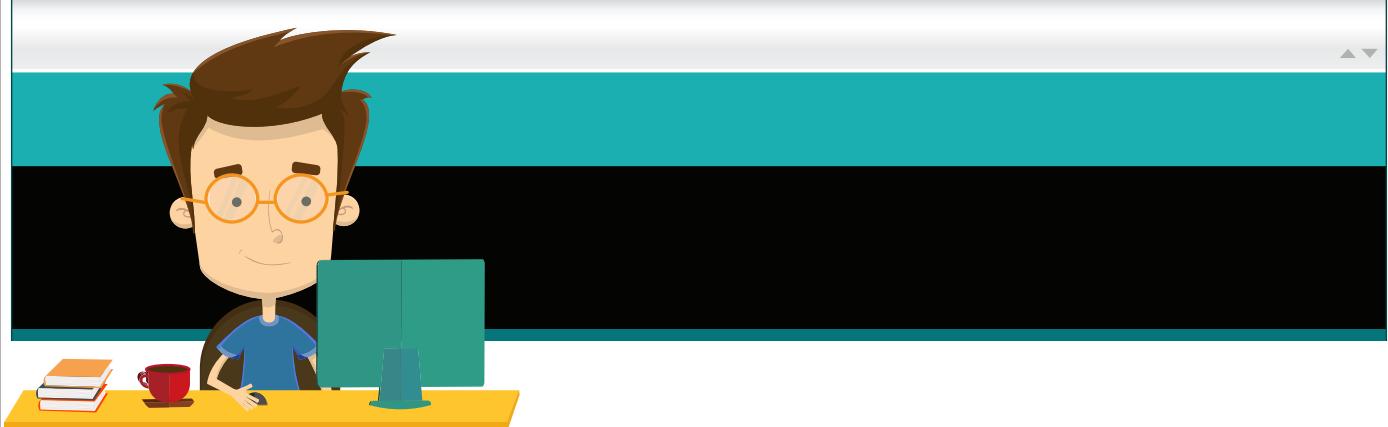




The image shows the Arduino IDE interface with the following details:

- Toolbar:** Includes icons for checkmark, refresh, file, upload, download, and settings.
- Title Bar:** Displays "Circuit_29".
- Code Area:** Contains the following C++ code for the TM1637 library:

```
if (n4==10){  
    n4=0;  
    n3++;  
    if (n3==6) {  
        n3=0;  
        n2++;  
        if (n2==10){  
            n2=0;  
            n1++;  
            if (n1==6) {  
                n1=0;  
            }  
        }  
    }  
}  
data[2]=number[n3];  
data[1]=number[n2];  
data[0]=number[n1];  
myTM.setSegments(data);  
}
```



ماژول ساعت



ماژول ساعت DS1302، از طریق ارتباط سریال با میکروکنترلر، زمان دقیق در لحظه را به دست می‌آورد. این زمان شامل ثانیه، دقیقه، ساعت، روز، ماه و سال است. این ماژول در صورتی که آردوبینو هم خاموش باشد، با استفاده از باتری تعبیه شده در ماژول، و اطلاعات ذخیره شده می‌تواند محاسبه لحظه به لحظه زمان را انجام دهد. در این پروژه، زمان به دست آمده از ماژول ساعت بر روی LCD1602 نمایش داده می‌شود.

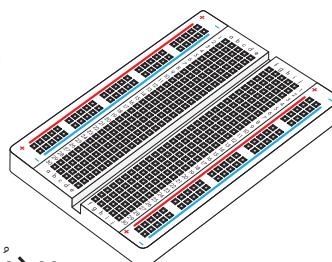
قطعات



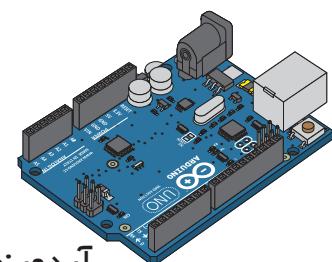
LCD1602
(۱ عدد)



سیم جامپر نری-نری
(۲ عدد)



برد



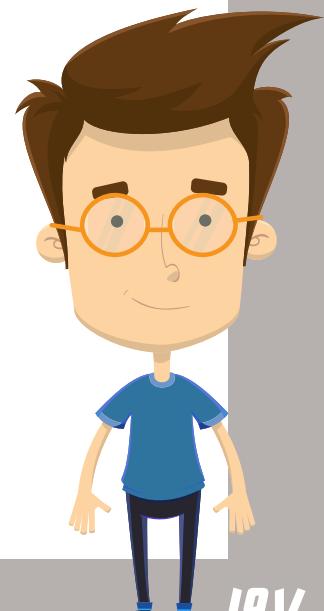
آردوبینو

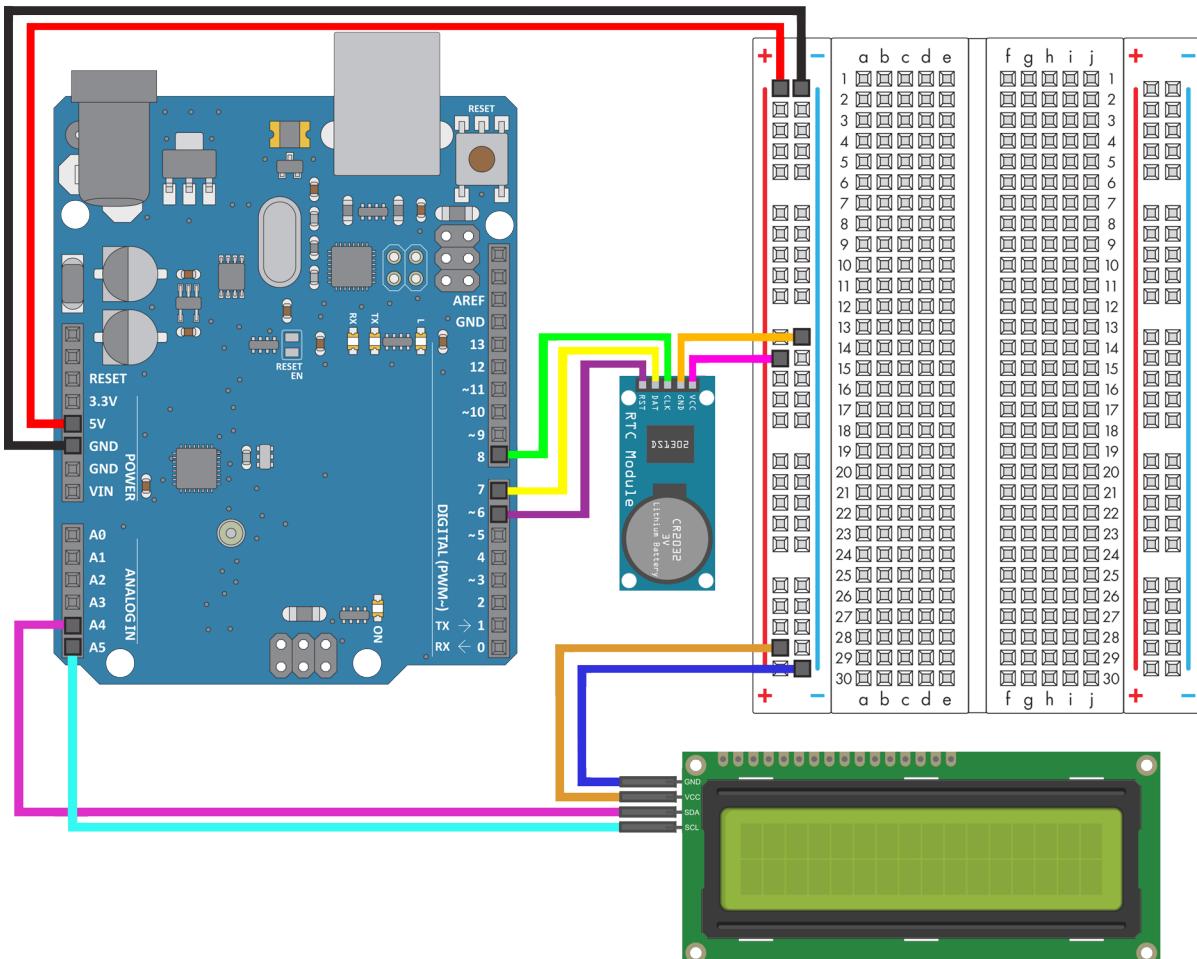


ماژول ساعت
(۱ عدد)



سیم جامپر نری-مادگی
(۹ عدد)





توضیح مدار:

برای نمایش زمان تولیدشده توسط ماژول DS1302 بر روی LCD1602 ابتدا پایه‌های VCC و GND هر دو ماژول DS1302 و LCD1602 به خانه‌های ستون‌های متناظر مثبت و منفی بر روی برد مصل می‌شود. این ستون‌ها به ترتیب به +5V و آردوینو متصل شده‌اند. پایه‌های SCL و SDA ماژول LCD1602 به ترتیب به پین‌های آنالوگ A4 و A5 آردوینو متصل می‌شود.

پایه RST ماژول DS1302 به پین شماره ۶، پایه DAT به پین شماره ۷ و پایه CLK به پین شماره ۸ آردوینو متصل می‌شود.



ماژول ساعت



برای استفاده از ماژول DS1302، از کتابخانه مربوط به این ماژول استفاده می‌شود. نحوه کدنویسی این ماژول پیچیدگی‌هایی دارد ولی با استفاده از دستورات کتابخانه، به سادگی می‌توان از این ماژول خروجی گرفت.

ابتدا کتابخانه این ماژول فراخوانی می‌شود:

```
#include <DS1302RTC.h>
```

سپس یک قطعه از نوع DS1302 تعریف می‌شود:

```
DS1302RTC myRTC (8, 7, 6);
```

سپس زمان شامل تاریخ و ساعت، تنظیم می‌شود:

```
myRTC.setDS1302Time(00, 30, 10, 2, 27, 6, 2017);
```

ترتیب پارامترها از راست به چپ به صورت زیر است:

سال میلادی(عدد صحیح)، عدد ماه (۰-۱۲)، عدد روز ماه (۰-۳۰)، عدد روز هفته (۰-۶)، ساعت (۰-۲۴)، دقیقه (۰-۵۹)، ثانیه (۰-۵۹)

برای نمایش زمان، ابتدا در هر حلقه، باید به روزرسانی انجام شود:

```
myRTC.updateTime();
```

برای نمایش اجزا مختلف زمان به صورت زیر عمل می‌شود:

عدد روز ماه (۰-۳۰) :

```
myLCD.print (myRTC.dayofmonth);
```

عدد ماه (۰-۱۲) :

```
myLCD.print (myRTC.month);
```

عدد سال:

```
myLCD.print (myRTC.year);
```

عدد ساعت (۰-۲۴) :

```
myLCD.print (myRTC.hours);
```

عدد دقیقه (۰-۵۹) :

```
myLCD.print (myRTC.minutes);
```

عدد ثانیه (۰-۵۹) :

```
myLCD.print (myRTC.seconds);
```





File > Examples > Electroduino > Circuit_30



Circuit_30



```
#include <DS1302RTC.h>
#include <LiquidCrystal_I2C.h>

DS1302RTC myRTC(6, 7, 8);
LiquidCrystal_I2C myLCD(0x3F,16,2);

void setup() {
    myRTC.setDS1302Time(00, 30, 10, 2, 27, 6, 2017);
    myLCD.init();
    myLCD.backlight();
}
```

File > Examples > Electroduino > Circuit _30

```
void loop() {  
    String weekDay[]={“SAT”, “SUN”, “MON”, “TUE”, “WED”, “THU”, “FRI”};  
    myRTC.updateTime();  
    myLCD.setCursor(1,0);  
    myLCD.print (weekDay[myRTC.dayofweek-1]);  
    myLCD.print (“ ”);  
    myLCD.print(myRTC.dayofmonth);  
    myLCD.print("-");  
    myLCD.print(myRTC.month);  
    myLCD.print("-");  
    myLCD.print(myRTC.year);  
    myLCD.setCursor(2,1);  
    myLCD.print(myRTC.hours);  
    myLCD.print(":");  
    myLCD.print(myRTC.minutes);  
    myLCD.print(":");  
    myLCD.print(myRTC.seconds);  
    delay( 1000);  
}
```



ماتریس ال ای دی ۸×۸

۳۱

ماتریس های LED به علت مصرف کم جریان، عمر بالا، ضد آب بودن، درخشندگی زیاد، زاویه دید زیاد و دلایل متعدد دیگر، از قطعات پرکاربرد هستند.

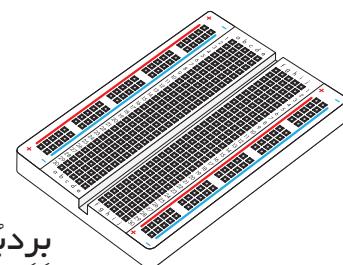
در این پروژه نحوه کار با یک ماتریس ال ای دی ۸×۸ توضیح داده می شود. اصول کاری مشابه پروژه های پیشین است، با این تفاوت که برای روشن کردن یک خانه، باید سطر آن خانه HIGH و ستون متناظر با آن خانه LOW شود.

با توجه به چیدمان ۸ تایی سطرهای و ستون های می توان از متغیر byte برای مقداردهی به سطر و ستون های استفاده کرد. در این حالت می توان وضعیت یک سطر را با عدد متناظر با آن (decimal, hexadecimal, binary) تعیین کرد و تصویر های مختلف ایجاد کرد.

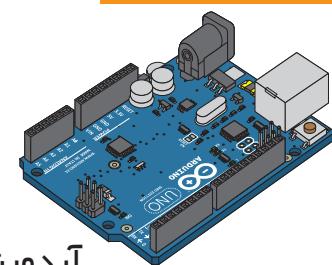
مطلوبات



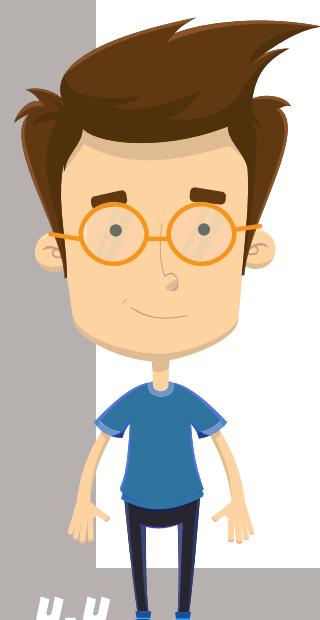
سیم جامپر نری-نری
(۱۷ عدد)



بردبرد



آردوینو



مقاومت ۲۲۰ اهم
(۸ عدد)



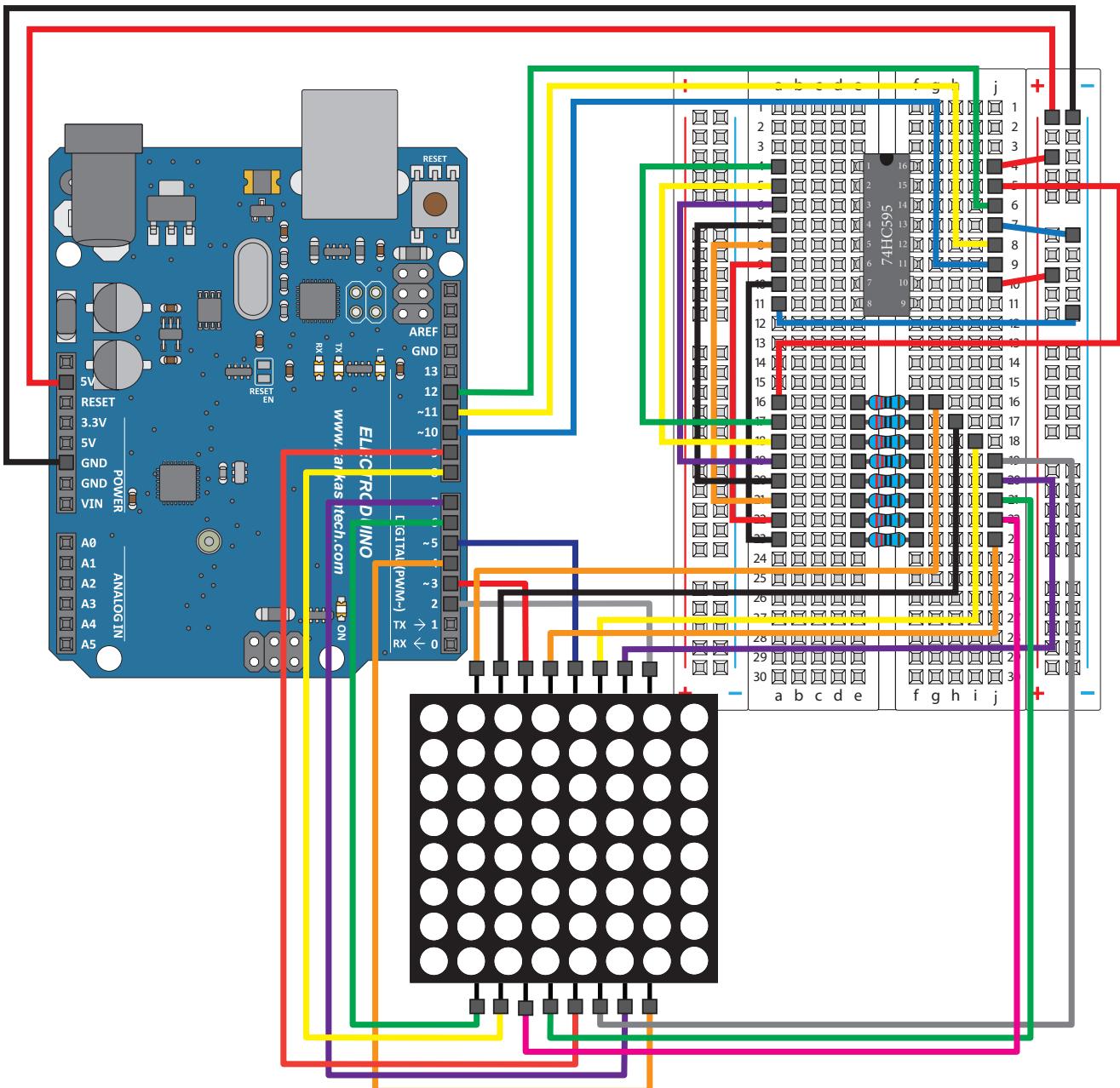
سون سگمنت
(۱ عدد)



سیم جامپر نری - مادگی
(۱۶ عدد)



شیفت رجیستر
(۱ عدد)

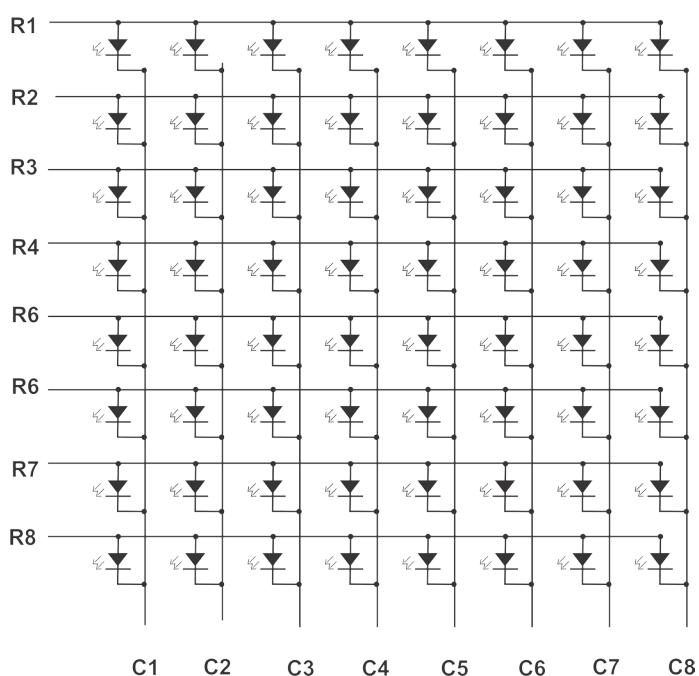
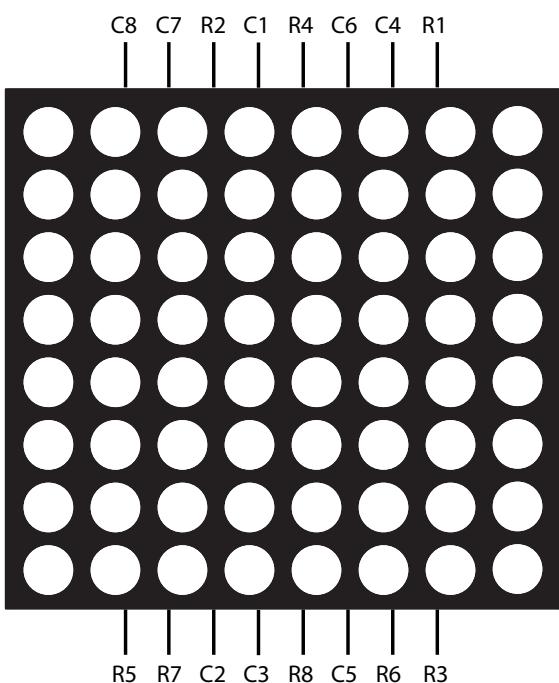


توضیح مدار:

با توجه به اندازه ماتریس، نمی‌توان آنرا بر روی برد برد قرار داد. لذا با استفاده از سیم‌های جامپر نری-مادگی پایه‌های ماتریس به پایه‌های شیفت رجیستر بر روی برد متصل می‌شود.

ماتریس LED دارای ۱۶ پایه است که ۸ پایه آن در بخش پایین و ۸ پایه دیگر در بخش بالای آن قرار دارد. در سطح پایینی ماتریس یک کد حک شده که توسط آن می‌توان جهت ماتریس را مشخص کرد هر کدام از این پایه‌ها منتظر با یکی از سطرهای R1-R8 یا ستون‌های C1-C8 است. مانند سون سگمنت، ترتیب قرار گیری این پایه‌ها منظم نیست و برای تشخیص پایه منتظر با هر کدام از ردیف‌ها یا ستون‌ها باید از راهنمای قطعه کمک گرفت. در تصویر سمت راست شماتیک ماتریس و در تصویر سمت چپ چیدمان پایه‌ها نشان داده شده است:

پایه‌های منتظر با ردیف‌های R1-R8 ماتریس، مستقیماً به پین‌های ۲-۹ آردوینو متصل می‌شوند. پایه‌های منتظر با ستون‌ها با واسطه یک مقاومت ۲۰ اهم به پایه‌های ۱-۷ و ۱۵ شیفت رجیستر متصل می‌شوند



از پین‌های شماره ۱۱، ۱۲ و ۱۰ آردوینو برای پایه‌های ۱۴، ۱۲، ۱۱ و ۱۱ شیفت رجیستر اتصال برقرار می‌شود.

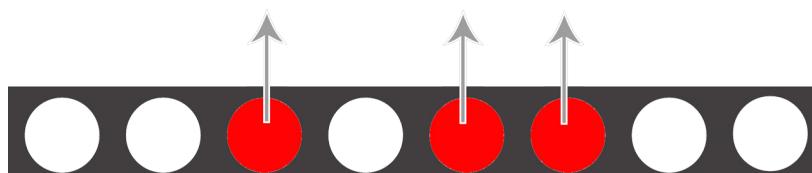


برای نمایش بر روی ماتریس، از اصلی مشابه سون سگمنت چهارتایی استفاده می شود. بدین صورت که ابتدا یک ردیف در وضعیت LOW و سایر ردیف ها در وضعیت HIGH قرار داده می شود. سپس بایت متناظر با خانه های روشن این ردیف ارسال می شود. با تکرار سریع این فرآیند برای همه ردیف ها، چشم انسان تمام ردیف ها را روشن می بیند.

پس از اینکه خانه های مورد نظر از ردیف اول روشن شد، با یک توقف بسیار کوچک، ابتدا کل خانه ها خاموش شده و سپس مراحل بالا برای ردیف دوم تکرار می شود. به علت سرعت بالای این فرآیند، چشم انسان تمام ردیف ها را به صورت روشن مشاهده می کند. با داشتن بایت مربوط به هر ردیف و ذخیره این بایت ها در یک آرایه، می توان با استفاده از یک حلقه، عدد یا تصویر مورد نظر را بر روی ماتریس نمایش داد.

به عنوان مثال اگر قرار باشد ردیفی به شکل زیر روشن شود، بایت ارسالی برابر خواهد بود با:

$$4 + 16 + 32 = 52$$



برای نمایش اعداد ۹-۰ و شکل های قلب و لبخند بر روی ماتریس، یک آرایه دو بعدی تعریف می شود. بُعد اول (۱۲) بیانگر شماره شکل مورد نظر و بُعد دوم (۸) بیانگر سطرهای شکل مذکور است. اعداد سطرها بایت هستند.

byte segments [12][8] = {

```
{60, 102, 66, 66, 66, 66, 102, 60},
{24, 24, 24, 24, 24, 24, 24, 24},
{60, 102, 66, 32, 16, 8, 4, 126},
{60, 98, 96, 60, 96, 64, 98, 60},
{32, 48, 40, 36, 126, 32, 32, 32},
{126, 2, 62, 98, 64, 64, 102, 60},
{60, 66, 2, 62, 66, 66, 66, 60},
{126, 64, 64, 32, 16, 8, 4, 2},
{60, 66, 66, 60, 66, 66, 66, 60},
{60, 66, 66, 66, 124, 64, 66, 60},
{60, 66, 165, 129, 165, 90, 60, 0},
{0, 68, 238, 254, 254, 124, 56, 16}
};
```



با استفاده از آرایه دو بعدی به سادگی می توان شکلی را بر روی ماتریس نشان داد بدین صورت که:

۱. ابتدا در یک حلقه، از بُعد اول آرایه اشکال، شکل مورد نظر انتخاب می شود؛
۲. با استفاده از یک حلقه دیگر، ابتدا ردیف مورد نظر در وضعیت LOW قرار می گیرد و سپس بایت مربوط به آن ردیف نمایش داده می شود.
۳. پس از یک مکث کوتاه (2 میلی ثانیه) ردیف فعلی در وضعیت $HIGH$ قرار گرفته و خاموش می شود.

پلا فاصله ردیف بعدی LOW شده و فرآیند تکرار می شود.
پس از اینکه ردیف هشتم خاموش شد، 16 میلی ثانیه گذشته است. برای اینکه چشم بتواند شکل را به خوبی مشاهده کند، این حلقه 60 بار تکرار می شود که مجموع توقف بر روی هر شکل حدودا 1 ثانیه شود ($16 * 60 = 960$ میلی ثانیه حدودا برابر با 1 ثانیه)



File > Examples > Electroduino > Circuit_31

```
int dataPin = 12;
int latchPin = 11;
int clockPin = 10;
int rowPins[8]={ 2, 3, 4, 5, 6, 7, 8, 9};
byte segments [12][8] = {
    {60,102,66,66,66,66,102,60},
    {24,24,24,24,24,24,24,24},
    {60,102,66,32,16,8,4,126},
    {60,98,96,60,96,64,98,60},
    {32,48,40,36,126,32,32,32},
    {126,2,62,98,64,64,102,60},
    {60,66,2,62,66,66,66,60},
    {126,64,64,32,16,8,4,2},
    {60,66,66,60,66,66,66,60},
    {60,66,66,66,124,64,66,60},
    {60,66,165,129,165,90,60,0},
    {0,68,238,254,254,124,56,16}
};
```





File > Examples > Electroduino > Circuit_31



Circuit_31

```
void setup() {
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
    for (int i=0 ; i<8; i++) {
        pinMode (rowPins[i], OUTPUT);
    }
}
```



File > Examples > Electroduino > Circuit_31



Circuit_31

```
void loop() {
    for(int k=0; k<12; k++) {
        for(int t=0; t<60; t++) {
            for (int j=0; j<8; j++) {
                digitalWrite(latchPin, LOW);
                shiftOut(dataPin, clockPin, LSBFIRST, segments[k][j]);
                digitalWrite(latchPin, HIGH);
                for (int r=0; r<8; r++){
                    if (r==j){
                        digitalWrite (rowPins[r], LOW);
                    }
                    else {
                        digitalWrite (rowPins[r], HIGH);
                    }
                }
                delay(2);
                digitalWrite (rowPins[j], HIGH);
            }
        }
    }
}
```



کنترل از راه دور

۳

با استفاده از یک فرستنده و گیرنده مادون قرمز می توان عملکرد بخش های مختلف مدار را از راه دور کنترل کرد.

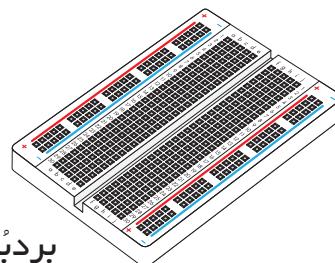
هر کدام از کلیدهای فرستنده، یک داده ۴ بایتی را ارسال می کند. با استفاده از مانیتور سریال یا LCD1602 می توان داده ارسالی را نمایش داد.

در این پروژه داده مرتبط با هر کلید بر روی مانیتور سریال نمایش داده می شود. برای انجام این کار از کتابخانه مربوط به کنترل از راه دور استفاده می شود.

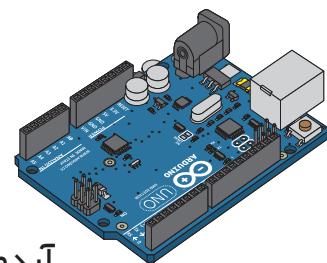
مطلوبات



سیم جامپر نزی-نری
(۱۲ عدد)



بردبرد



آردوینو



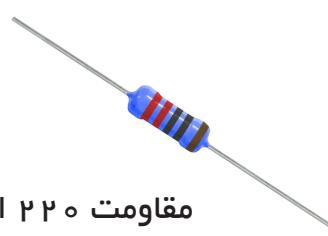
LED
(۸ عدد)



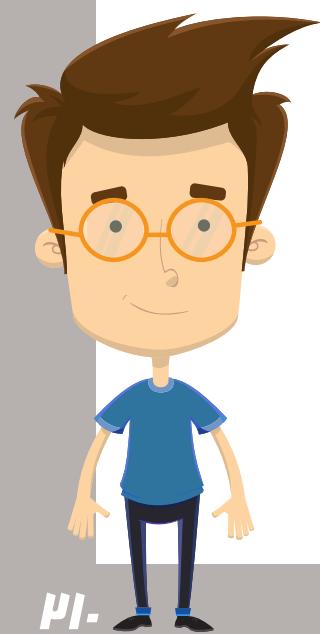
فرستنده مادون قرمز
(۱ عدد)

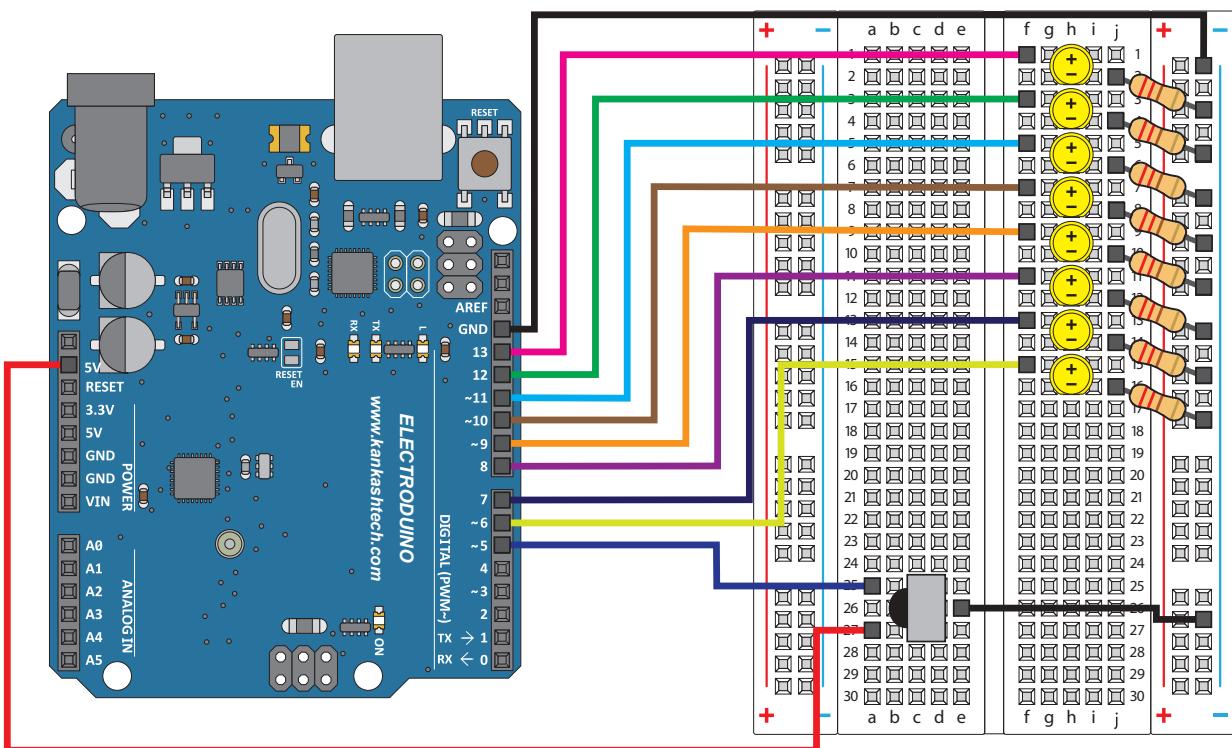


گیرنده مادون قرمز
(۱ عدد)



مقاومت ۲۲۰ اهم
(۸ عدد)





توضیح مدار:

گیرنده IR، سه پایه دارد. پایه های VCC و GND به پین های متناظر روی آردوینو متصل می شوند. پایه سیگنال به پین شماره ۵ آردوینو متصل می شود.





برای استفاده از کنترل از راه دور، ابتدا کتابخانه ان فراخوانی می شود:

```
#include <IRremote.h>
```

سپس کنترل از راه دور تعریف می شود:

```
int irPin = 5;
```

```
IRrecv myIR (irPin);
```

بعد از تعریف کنترل از راه دور، متغیر خروجی تعریف می شود:

```
decode_results irInput;
```

در تابع `setup`، مازول کنترل از راه دور فعالسازی می شود:

```
myIR.enableIRIn();
```

همچنین مانیتور سریال نیز راه اندازی می شود.

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

در تابع `loop`، تعیین می شود که در صورت دریافت سیگنال از کنترل از راه دور، مقدار دریافتی

برروی مانیتور سریال نمایش داده شود:

```
if (myIR.decode(&irInput)) {  
    Serial.println(irInput.value, HEX);  
    myIR.resume();  
}
```

با دانستن مقدار مربوط به هر کلید، می توان برای هر کدام از کلیدهای کنترل از راه دور، یک عملکرد تعیین کرد.



برای کنترل از راه دور LEDها، شماره پین مربوط به LEDها درون یک آرایه ذخیره می‌شود.

```
int ledPin []={13, 12, 11, 10, 9, 8, 7, 6};
```

سپس داده‌های مربوط به کلیدهای ۱-۸ در یک آرایه ذخیره می‌شود. برخی از فرستنده‌ها، در حالت‌های مختلف فشردن یک کلید، داده‌های متفاوتی بر می‌گردانند. می‌توان با استفاده از یک آرایه چند بعدی، همه حالت‌های مربوط به فشردن کلید‌ها را ذخیره کرد و با توجه به موقعیت گزینه منطبق در بُعد دوم آرایه، شماره کلید را تشخیص داد.

```
long controlKey[2][8] ={  
{0xFF30CF, 0xFF18E7, 0xFF7A85, 0xFF10EF, 0xFF38C7, 0xFF5AA5, 0xFF42BD, 0xFF4AB5},  
{0x9716BE3F, 0x3D9AE3F7, 0x6182021B, 0x8C22657B, 0x488F3CBB, 0x449E79F,  
0x32C6FDF7, 0x1BC0157B}  
};
```

می‌توان ابتدا همه داده‌ها را صفر مقدار دهی کرد و سپس با مشاهده بر روی مانیتور سریال، مقادیر صحیح را جایگزین کرد.

اگر مقدار دریافتی توسط گیرنده با یکی از مقادیر تعریف شده برای دکمه‌ها یکسان بود، LED متناظر با آن روشن می‌شود و شماره LED مذکور در مانیتور سریال نوشته می‌شود:

```
if (irInput.value == controlKey[k][j]) {  
    digitalWrite(ledPin[j], HIGH);  
    Serial.print("LED no.");  
    Serial.print(j+1);  
    Serial.println(" is on.");  
}
```

در دستور بالا، هر دو ردیف از کدهای کلیدها چک شده و بسته به موقعیت آیتم در آن ردیف از اطلاعات، LED متناظر با آن موقعیت روشن می‌شود.





File > Examples > Electroduino > Circuit_32

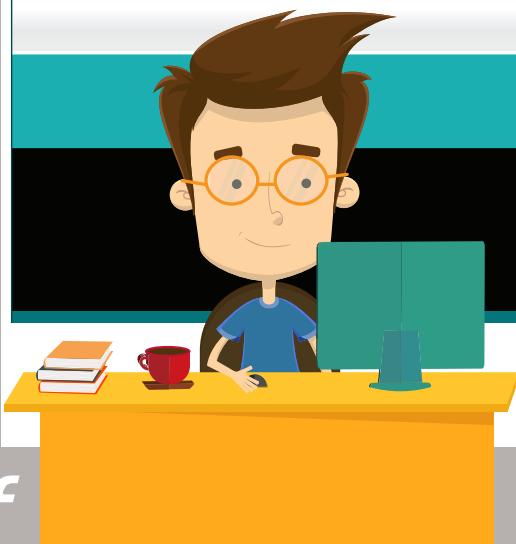


Circuit_32



```
#include <IRremote.h>
int irPin =5;
IRrecv myIR(irPin);
int ledPin[] ={13, 12, 11, 10, 9, 8, 7, 6};
long controlKey[2][8] ={{
    {0xFF30CF, 0xFF18E7, 0xFF7A85, 0xFF10EF, 0xFF38C7, 0xFF5AA5, 0xFF42BD,
     0xFF4AB5},
    {0x9716BE3F, 0x3D9AE3F7, 0x6182021B, 0x8C22657B, 0x488F3CBB,
     0x449E79F, 0x32C6FDF7, 0x1BC0157B}
}};
decode_results irInput;

void setup() {
    Serial.begin(9600);
    for (int i=0; i<8; i++) {
        pinMode(ledPin[i], OUTPUT);
        digitalWrite(ledPin[i], LOW);
    }
    myIR.enableIRIn();
}
```



File > Examples > Electroduino > Circuit_32



Circuit_32

```
void loop() {
    if (myIR.decode(&irInput)) {
        Serial.println(irInput.value, HEX);
        myIR.resume();
    }
    delay(600);
    for (int k=0; k<2; k++) {
        for (int j=0; j<8; j++) {
            if (irInput.value == controlKey[k][j]) {
                digitalWrite(ledPin[j], HIGH);
                Serial.print("LED no.");
                Serial.print(j+1);
                Serial.println(" is on.");
            }
            else {
                digitalWrite(ledPin[j], LOW);
            }
        }
    }
}
```





صفحه کلیدها در دستگاه های مختلف نظیر تلفن، فکس، اجاق مایکروویو و ... به عنوان وسیله ورودی استفاده می شوند.

در این پروژه نحوه اتصال و خواندن کلیدهای یک صفحه کلید ۴x۱۶ ماتریسی و نمایش کد مربوط به فشردن هر کلید در مانیتور سریال و استفاده از آن برای ساخت یک قفل رمزی توضیح داده شده است.

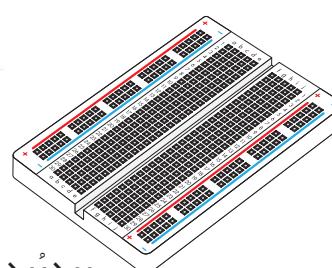
مطلوبات



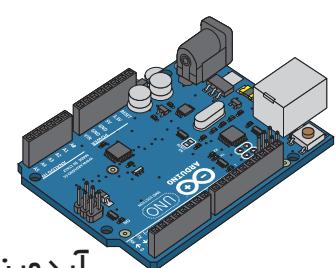
LCD1602
(۱ عدد)



سیم جامپر نری-نری
(۱۲ عدد)



بردبرد



آردوینو



صفحه کلید ۴x۱۶
(۱ عدد)



سیم جامپر نری-مادگی
(۱۴ عدد)



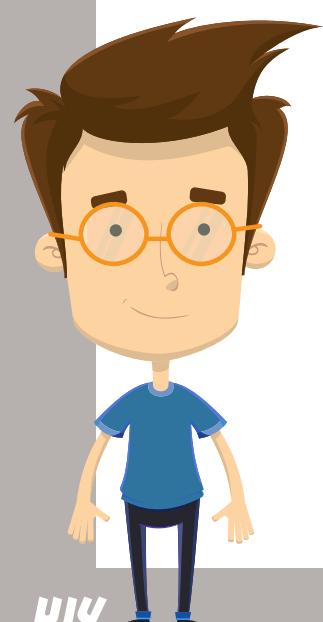
مقاومت ۲۲۰ اهم
(۲ عدد)

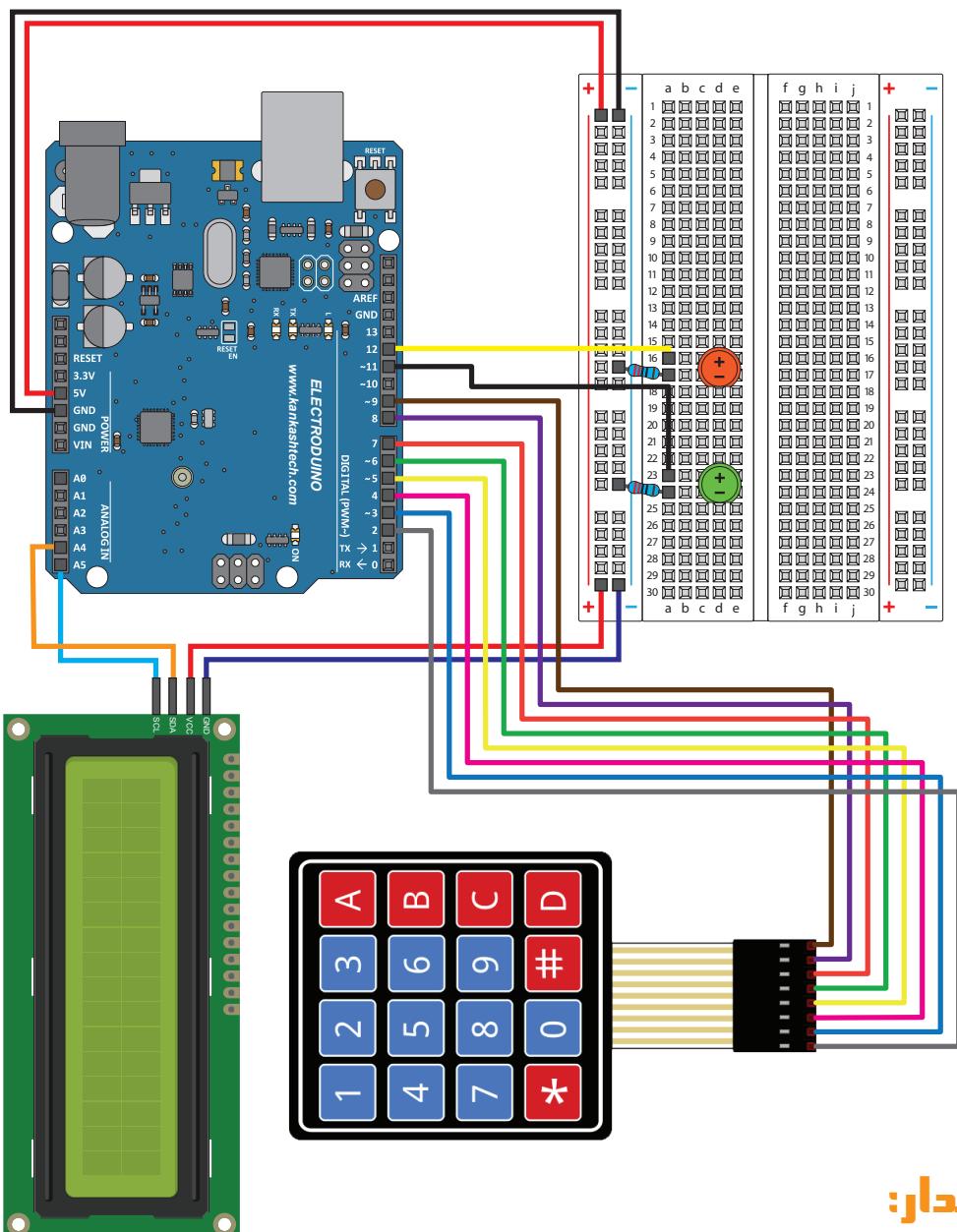


LED سبز
(۱ عدد)



LED قرمز
(۱ عدد)





توضیح مدار:

صفحه کلید ۱۶x۲، ۸ پین اتصال دارد که ۴ پین برای ردیف ها و ۴ پین برای ستون های کلیدهاست. پین ها از راست به چپ به ترتیب به پین های ۹ الی ۲ آردوینو متصل می شوند. یک LED قرمز به پین ۱۲ و یک LED سبز به پین ۱۱ متصل می شود.

پایه های VCC و GND مژول LCD1602 به پایه های متناظر آردوینو و پایه های SCL و SDA به ترتیب به پین های آنالوگ A5 و A4 متصل می شوند.





برای استفاده از صفحه کلید، ابتدا کتابخانه آن فراخوانی می شود:

```
#include <Keypad.h>
```

سپس تعداد ردیف ها و ستون ها تعیین شود:

```
byte rows = 4;  
byte cols = 4;
```

سپس پین های متناظر هر کدام از سطرها و ستون ها تعریف شود:

```
byte rowPins[rows] = {2, 3, 4, 5};  
byte colPins[cols] = {6, 7, 8, 9};
```

سپس علائم هر کدام از کلیدهای صفحه کلید در یک آرایه دو بعدی تعریف می شود:

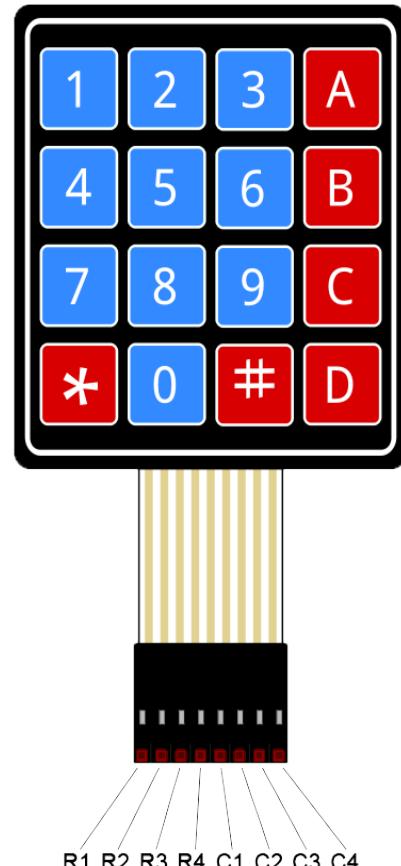
```
char keyNames [rows] [cols] = {  
    {'1', '2', '3', 'A'},  
    {'4', '5', '6', 'B'},  
    {'7', '8', '9', 'C'},  
    {'*', '0', '#', 'D'}  
};
```

سپس یک شیء از نوع صفحه کلید تعریف می شود:

```
Keypad myKeypad (makeKeymap(keyNames), rowPins, colPins, rows, cols);
```

برای خواندن صفحه کلید از دستور `getKey()`. استفاده می شود:

```
char key = myKeypad.getKey();
```



با خواندن کلیدهای فشرده شده و تلفیق آنها، می‌توان از ورودی به عنوان رمز یا فعالساز یک دستور استفاده کرد.

هر کدام از کاراکتر‌ها با یک عدد در حافظه ذخیره می‌شوند که به آن کد ASCII گفته می‌شود. همانگونه که دیده می‌شود، عدد "۱" و کاراکتر "ا" متفاوت هستند.

برای تبدیل کاراکتر یک عدد به عدد متناظر، می‌توان آنرا از کاراکتر "۰" کم کرد. به عنوان مثال برای تبدیل کاراکتر "۵" به عدد ۵ در مبنای ۱۰، باید کاراکتر "۰" را از کاراکتر "۵" کم کرد. در واقع در این محاسبه، عدد ۱۴۸ از ۱۵۳ کم شده و به ۵ عددی تبدیل شده است.

کد ASCII کاراکترهای مختلف در زیر قابل مشاهده است.

Decimal	Hexadecimal	Binary	Char	Decimal	Hexadecimal	Binary	Char	Decimal	Hexadecimal	Binary	Char
0	0	0	[NULL]	48	30	110000	0	96	60	1100000	`
1	1	1	[START OF HEADING]	49	31	110001	1	97	61	1100001	a
2	2	10	[START OF TEXT]	50	32	110010	2	98	62	1100010	b
3	3	11	[END OF TEXT]	51	33	110011	3	99	63	1100011	c
4	4	100	[END OF TRANSMISSION]	52	34	110100	4	100	64	1100100	d
5	5	101	[ENQUIRY]	53	35	110101	5	101	65	1100101	e
6	6	110	[ACKNOWLEDGE]	54	36	110110	6	102	66	1100110	f
7	7	111	[BELL]	55	37	110111	7	103	67	1100111	g
8	8	1000	[BACKSPACE]	56	38	111000	8	104	68	1101000	h
9	9	1001	[HORIZONTAL TAB]	57	39	111001	9	105	69	1101001	i
10	A	1010	[LINE FEED]	58	3A	111010	:	106	6A	1101010	j
11	B	1011	[VERTICAL TAB]	59	3B	111011	;	107	6B	1101011	k
12	C	1100	[FORM FEED]	60	3C	111100	<	108	6C	1101100	l
13	D	1101	[CARRIAGE RETURN]	61	3D	111101	=	109	6D	1101101	m
14	E	1110	[SHIFT OUT]	62	3E	111110	>	110	6E	1101110	n
15	F	1111	[SHIFT IN]	63	3F	111111	?	111	6F	1101111	o
16	10	10000	[DATA LINK ESCAPE]	64	40	1000000	@	112	70	1110000	p
17	11	10001	[DEVICE CONTROL 1]	65	41	1000001	A	113	71	1110001	q
18	12	10010	[DEVICE CONTROL 2]	66	42	1000010	B	114	72	1110010	r
19	13	10011	[DEVICE CONTROL 3]	67	43	1000011	C	115	73	1110011	s
20	14	10100	[DEVICE CONTROL 4]	68	44	1000100	D	116	74	1110100	t
21	15	10101	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	E	117	75	1110101	u
22	16	10110	[SYNCHRONOUS IDLE]	70	46	1000110	F	118	76	1110110	v
23	17	10111	[ENG OF TRANS. BLOCK]	71	47	1000111	G	119	77	1110111	w
24	18	11000	[CANCEL]	72	48	1001000	H	120	78	1111000	x
25	19	11001	[END OF MEDIUM]	73	49	1001001	I	121	79	1111001	y
26	1A	11010	[SUBSTITUTE]	74	4A	1001010	J	122	7A	1111010	z
27	1B	11011	[ESCAPE]	75	4B	1001011	K	123	7B	1111011	{
28	1C	11100	[FILE SEPARATOR]	76	4C	1001100	L	124	7C	1111100	
29	1D	11101	[GROUP SEPARATOR]	77	4D	1001101	M	125	7D	1111101	}
30	1E	11110	[RECORD SEPARATOR]	78	4E	1001110	N	126	7E	1111110	~
31	1F	11111	[UNIT SEPARATOR]	79	4F	1001111	O	127	7F	1111111	[DEL]
32	20	100000	[SPACE]	80	50	1010000	P				
33	21	100001	!	81	51	1010001	Q				
34	22	100010	"	82	52	1010010	R				
35	23	100011	#	83	53	1010011	S				
36	24	100100	\$	84	54	1010100	T				
37	25	100101	%	85	55	1010101	U				
38	26	100110	&	86	56	1010110	V				
39	27	100111	'	87	57	1010111	W				
40	28	101000	(88	58	1011000	X				
41	29	101001)	89	59	1011001	Y				
42	2A	101010	*	90	5A	1011010	Z				
43	2B	101011	+	91	5B	1011011	[
44	2C	101100	,	92	5C	1011100	\				
45	2D	101101	-	93	5D	1011101]				
46	2E	101110	.	94	5E	1011110	^				
47	2F	101111	/	95	5F	1011111	_				





اگر key متغیر نگهدارنده کاراکتر دکمه صفحه کلید از نوع char باشد، برای تبدیل آن به عدد از دستور زیر استفاده می شود.

`key - '0'`

اگر کلید جدیدی فشرده شود، عدد قبلی در 10 ضرب شده و با عدد جدید جمع می شود:
`num = num * 10 + (key - '0');`

به عنوان مثال اگر ابتدا عدد ۵ و سپس ۷ فشرده شود، اتفاقات زیر می افتد:

۱. ابتدا متغیر num برابر با صفر است.
۲. با فشرده شدن کلید ۵، کاراکتر ۵ به عدد ۵ تبدیل می شود و در متغیر num قرار می گیرد.
۳. با فشرده شدن کلید ۷، کاراکتر ۷ به عدد ۷ تبدیل می شود.
۴. از آنجایی که عدد ۵۷ بر روی صفحه کلید فشرده شده است، عدد num قبلی یعنی ۵، در ۱۰ ضرب شده و به دهگان عدد جدید (یعنی ۵۰) تبدیل می شود.
۵. با افزودن مقدار جدید فشرده شده یعنی ۷، عدد مورد نظر ما یعنی ۵۷ تشکیل می شود.

حلقه (While)

یکی دیگر از روش های ایجاد حلقه، استفاده از () While است.

```
While ( key != '#' ) {  
}
```

این حلقه تا زمانی که شرط حلقه برقرار باشد ادامه می یابد. در حلقه فوق تا زمانی که دکمه # فشرده نشده است، حلقه ادامه می یابد



switch...case

اگر بخواهیم برای وضعیت مختلف عملکردهای متفاوتی تعریف کرد، می‌توان از switch...case استفاده کرد. این دستور، مقدار ورودی را با حالت‌های تعریف شده مقایسه کرده و در صورت تطابق، دستورات مربوط به آن حالت را اجرا می‌کند.

```
switch (key) {
    case NO_KEY:
        break;
}
```

در دستور فوق، در صورتی که کلیدی فشرده نشده باشد، حلقه متوقف شده و هیچ دستوری اجرا نمی‌شود.

اگر برای چند حالت مختلف، یک عملکرد مورد انتظار باشد، می‌توان آن حالت‌ها را پشت سر هم نوشت:

```
case 'A': case 'B': case 'C': case 'D':
    myLCD.clear();
    myLCD.print ("Invalid Input");
    break;
```

برای نوشتمن دستورات case‌ها به '}' نیازی نیست و تمام دستورات بعد از case موفق تا پیش از break اجرا می‌شود. در نتیجه حتماً باید برای هر case یک break نوشته، در غیر این صورات همه دستورات بعد از case موفق، حتی دستورات متعلق به case‌های ناموفقی که بعد از case موفق قراردارند اجرا خواهد شد.

در دستورات فوق در صورتی که کلیدهای C، B، A، یا D فشرده شوند، عبارت "Invalid Input" بر روی LCD نمایش داده می‌شود.





File > Examples > Electroduino > Circuit_33

```
Circuit_33

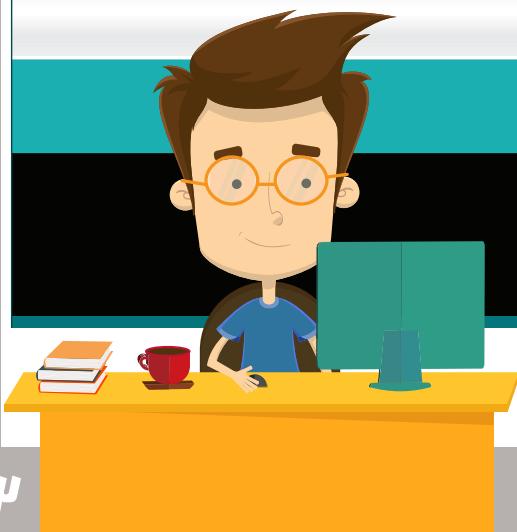
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

byte rows = 4;
byte cols = 4;
byte rowPins [rows] = {2, 3, 4, 5};
byte colPins [cols] = {6, 7, 8, 9};

int redLED = 12;
int greenLED = 11;
int password =1234;

char keyNames [rows] [cols] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

Keypad myKeypad (makeKeymap(keyNames), rowPins, colPins, rows, cols);
LiquidCrystal_I2C myLCD(0x3F, 16, 2);
```



File > Examples > Electroduino > Circuit_33

```
Circuit_33
```

```
void setup() {
    myLCD.init();
    myLCD.backlight();
    myLCD.clear();
    pinMode(redLED, OUTPUT);
    pinMode(greenLED, OUTPUT);
    myLCD.setCursor(4, 0);
    myLCD.print("Welcome");
    delay(2000);
    myLCD.setCursor(1, 1);
    myLCD.print("Enter Password");
    Serial.begin(9600);
}
```





File > Examples > Electroduino > Circuit_33



Circuit_33

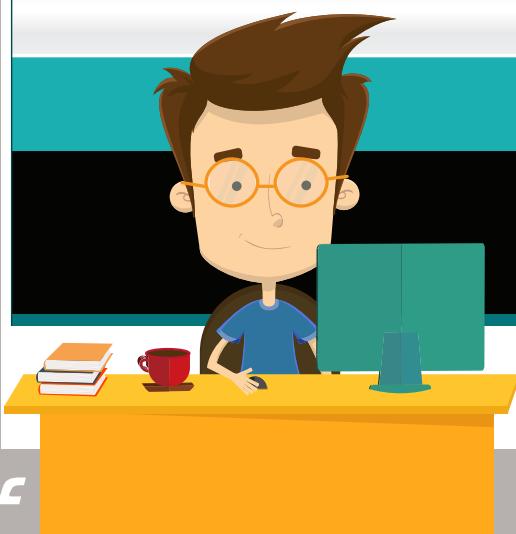
```
void loop(){

long pass= getNumber();

if (pass==password){
    myLCD.clear ();
    myLCD.setCursor(0, 0);
    myLCD.print("**** Verified ****");
    myLCD.setCursor(0, 1);
    myLCD.print("Door Unlocked");
    digitalWrite (greenLED , HIGH);
    digitalWrite (redLED, LOW);
}

else {
    myLCD.clear ();
    myLCD.setCursor(0, 0);
    myLCD.print("Wrong Password");
    myLCD.setCursor(3, 1);
    myLCD.print("Try Again");
    digitalWrite (greenLED , LOW);
    digitalWrite (redLED, HIGH);
}

}
```



File > Examples > Electroduino > Circuit_33

```
long getNumber() {
    long num = 0;
    char key = myKeypad.getKey();
    while(key != '#') {
        switch (key) {
            case NO_KEY:
                break;
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
                myLCD.clear();
                num = num * 10 + (key - '0');
                myLCD.setCursor(5,0);
                myLCD.print(num);
                Serial.println(key);
                break;
            case '*':
                num = 0;
                myLCD.clear();
                break;
            case 'A': case 'B' : case 'C': case 'D':
                myLCD.clear();
                myLCD.print ("Invalid Input");
                break;
        }
        key = myKeypad.getKey();
    }
    return num;
}
```





ماژول RFID

مخفف RFID به معنای شناسایی توسط فرکانس رادیویی است. این یک کاربرد بی سیم برای شناسایی و دنبال کردن تگ (tag) هاست در این پروژه با استفاده از ماژول و تگ RFID، LCD1602، یک LED قرمز و یک LED سبز عملکرد این ماژول توضیح داده می شود.

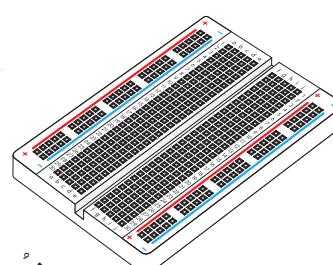
مطلوبات



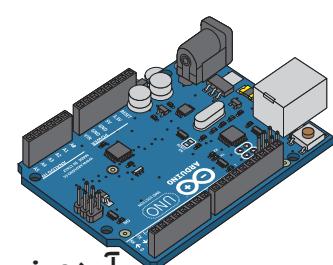
LCD1602
(۱ عدد)



سیم جامپر نری-نری
(۱۲ عدد)



بردبر



آردوینو



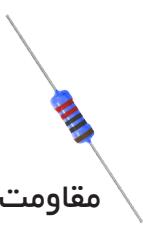
تگ RFID
(۱ عدد)



ماژول RFID
(۱ عدد)



سیم جامپر نری-مادگی
(۱۴ عدد)



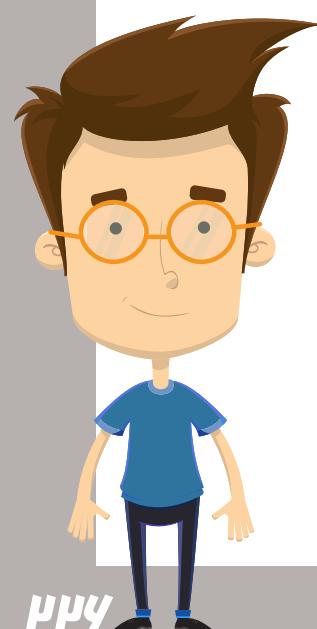
مقاومت ۲۲۰ اهم
(۲ عدد)

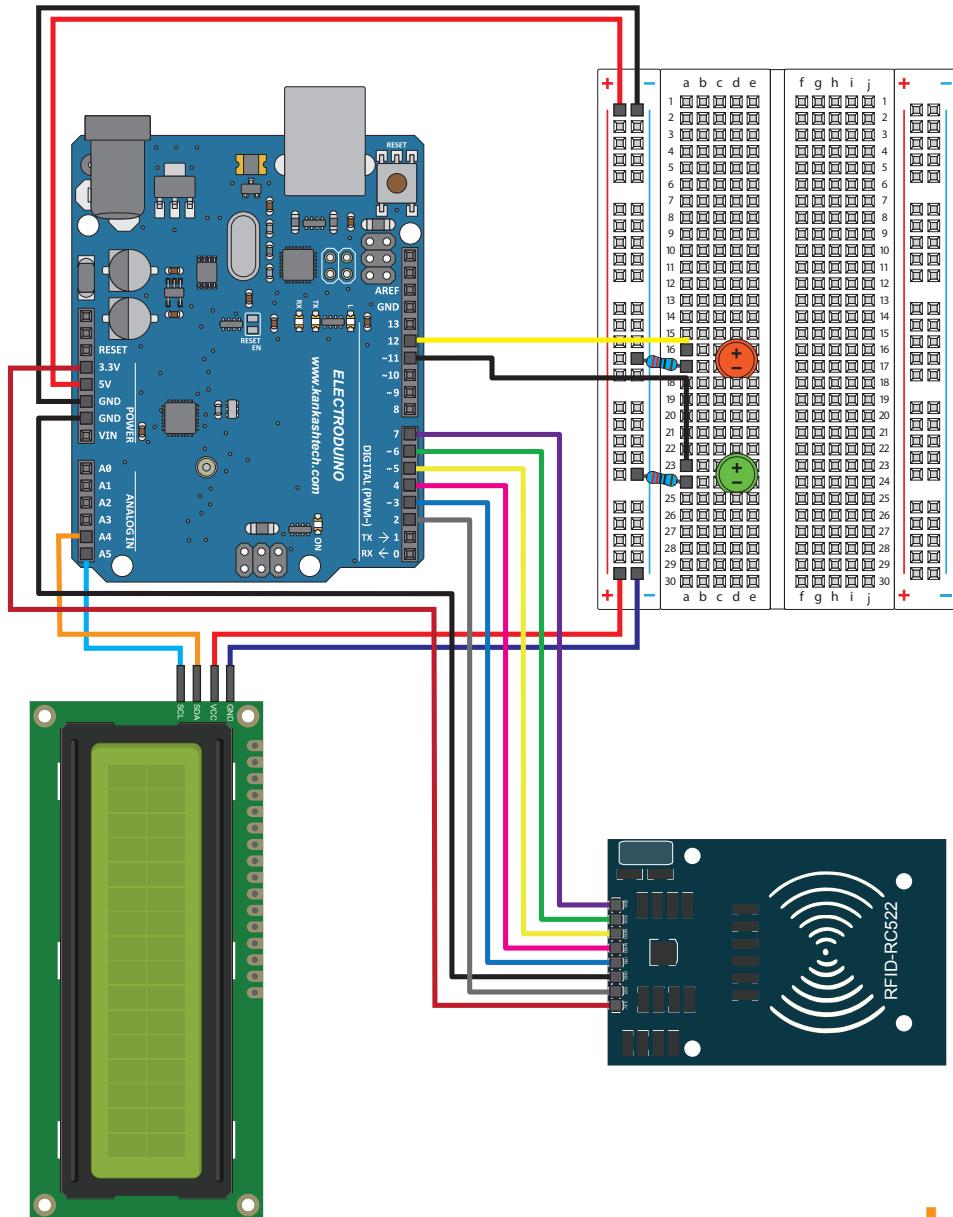


LED سبز
(۱ عدد)



LED قرمز
(۱ عدد)





توضیح مدار:

ماژول RFID، هشت پایه دارد. پایه های اول و سوم مربوط به ولتاژ و GND هستند. سایر پایه ها به ترتیب به پین های ۲ الی ۷ آردوینو متصل می شوند.

باید دقต داشت که این ماژول با ولتاژ ۳.۳ ولت کار می کند و در صورت اتصال به ولتاژ ۵ ولت احتمال آسیب دیدگی آن وجود دارد.

یک LED قرمز به پین ۱۲ و یک LED سبز به پین ۱۱ متصل می شود.

پایه های GND و VCC ماژول LCD1602 به پایه های متناظر آردوینو و پایه های SCL و SDA به ترتیب به پین های آنالوگ A5 و A4 آردوینو متصل می شوند.



تعریف RFID

برای استفاده از ماژول RFID، ابتدا کتابخانه آن فراخوانی می شود:

```
#include <RFID.h>
```

سپس یک شی از نوع RFID تعریف شود:

```
RFID myRFID;
```

سپس ماژول راه اندازی می شود. برای این کار باید پین های آردوینو متصل به هر کدام از پین های ماژول مشخص شود:

```
myRFID.begin(IRQ_PIN, SCK_PIN, MOSI_PIN, MISO_PIN, SDA_PIN, RST_PIN)
```

پس از یک مکث کوتاه، ماژول شروع به کار می کند:

```
myRFID.init();
```

دستور زیر، ID تگ مقابل ماژول را در متغیر str ذخیره می کند.

```
status = myRFID.request(PICC_REQIDL, str);
```

در صورتی که دستور request با موفقیت انجام شود (یک تگ شناسایی شود)، مقدار MI_OK برگردانده می شود. با ذخیره مقدار بازگشته در متغیر status، می توان تشخیص داد که آیا کارتی در مقابل ماژول قرار گرفته است یا خیر.

هر دو متغیر str و status از نوع byte هستند.



خواندن تگ RFID

پیش از اینکه بتوان عملکردی را برای یک تگ تعیین کرد، باید ID آن را استخراج کرد. برای استخراج ID، ابتدا توسط دستور `anticoll`, تگ مقابل ماثول RFID چک می شود.

```
status = myRFID.anticoll(str);
```

سپس با استفاده از دستور `Serial.print`, شناسه یا ID تگ در مانیتور سریال نمایش داده می شود.
`Serial.print(str[0], HEX);`
`Serial.print(str[1], HEX);`
`Serial.print(str[2], HEX);`
`Serial.print(str[3], HEX);`

برای اینکه دستور بالا، ID را در مانیتور سریال نشان دهد، باید ابتدا ارتباط سریال راه اندازی شده باشد.

کد مربوط به خواندن ID در یک sketch و کد مربوط به استفاده از آن در یک sketch دیگر نوشته می شود.

استفاده از ID تگ

ID خوانده شده از تگ (str)، در واقع یک آرایه ۴ تایی است که هر کدام از ۴ آیتم، یک بایت هستند. بعد از خواندن ID مربوط به تگ، و قراردادن آن در کد، می توان دستور خاصی را برای آن تگ تعریف کرد. برای اینکه ID تگ با ID موردنظر مقایسه شود، باید کل آرایه با یکدیگر مقایسه شود. برای این کار باید بایت ها را از یکدیگر جدا کرد.

اگر ID نمایش داده شده در مانیتور سریال C0 EC B4 4B باشد، بدین معناست که:

```
str[0]=0xC0  
str[1]=0xEC  
str[2]=0xB4  
str[3]=0x4B
```

0x اضافه شده به هر بایت برای این است که مشخص شود عدد در مبنای ۱۶ (HEX) است.

حال می توان ID کارت خوانده شده را به صورت بایت به بایت با ID کارت مورد نظر مقایسه و در صورت تطابق دستوراتی را اجرا کرد.





Circuit_34_1

```
#include <RFID.h>
RFID myRFID;
void setup(){
    Serial.begin(9600);
    myRFID.begin(3, 6, 5, 4, 7, 2);
    delay(100);
    myRFID.init();
}
void loop(){
    byte status;
    byte str[4];
    status = myRFID.request(PICC_REQIDL, str);
    if (status != MI_OK){
        return;
    }
    else{
        status = myRFID.anticoll(str);
        Serial.print("The card's number is: ");
        Serial.print(str[0], HEX);
        Serial.print(" ");
        Serial.print(str[1], HEX);
        Serial.print(" ");
        Serial.print(str[2], HEX);
        Serial.print(" ");
        Serial.print(str[3], HEX);
        Serial.println();
        Serial.println();
    }
    delay(500);
}
```



File > Examples > Electroduino > Circuit_34_2

```
#include<RFID.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C myLCD(0x3F,16,2);
RFID myRFID;

int redLED = 12;
int greenLED = 11;

void setup(){
    myLCD.init();
    myLCD.backlight();
    myRFID.begin(3, 6, 5, 4, 7, 2);
    delay(100);
    myRFID.init();
    pinMode (greenLED, OUTPUT);
    pinMode (redLED, OUTPUT);
    digitalWrite (redLED, HIGH);
    digitalWrite (greenLED, LOW);
}
```





Circuit_34_2

```
void loop(){
    byte status;
    byte str[4];
    status = myRFID.request(PICC_REQIDL, str);
    status = myRFID.anticoll(str);

    myLCD.setCursor(3,0);
    myLCD.print(" WELCOME! ");
    digitalWrite(redLED, HIGH);
    digitalWrite(greenLED, LOW);

    if (status != MI_OK) {
        return;
    }
    else {
        if( str[0]==0xC0 && str[1]==0xEC && str[2]==0xB4 && str[3]==0x4B ) {
            digitalWrite(greenLED, HIGH);
            digitalWrite(redLED, LOW);
            myLCD.clear();
            myLCD.setCursor(0,0);
            myLCD.print("Hi KankashTech!");
            myLCD.setCursor(0,1);
            myLCD.print("Door is Unlocked");
            delay(5000);
            myLCD.clear();
        }
    }
}
```



File > Examples > Electroduino > Circuit_34_2

```
else {
    myLCD.clear();
    myLCD.setCursor(1,1);
    myLCD.print("Unknown Card!");
    digitalWrite(redLED, HIGH);
    digitalWrite(greenLED, LOW);
    delay(5000);
    myLCD.clear();
}
}
```

