آموزش مقدماتی آردوینو و اینترنت اشیاء

با استفاده از کیت ProMake®

تهیه کننده

شرکت دانش بنیان گیگا پرداز پارس

فهرست

پیش گفتار

با پیشرفت فناوری و کاهش هزینه‌های تولید در حوزه نیمه هادی‌ها و ریزپردازنده‌ها، شاهد حضور پررنگ آنها در زندگی روزمره و فضای پیرامون خود هستیم. امروزه در منزل، خودرو، مغازه و حتی روی بدن خود به هرجایی که نگاه می‌کنیم شاهد حضور یک پردازنده هستیم که اغلب به اینترنت نیز متصل هستند یا یا به زودی متصل خواهند شد.

ساعت هوشمند، گوشی موبایل، تلویزیون هوشمند، یخچال و دیگر لوازم خانگی هوشمند، دربازکن تصویری، مولتی مدیا خودرو، پایانه های پرداخت بانکی و ... نمونه هایی از نفوذ تراشه های برنامه پذیر و متصل به زندگی مدرن انسان هستند. اینترنت اشیاء (یا همان IoT) بیانگر این واقعیت است که این نفوذ در سال‌های پیش رو افزایش چشمگیر تری خواهد یافت و شاید در آینده نزدیک دیگر هیچ شئ در اطراف خود پیدا نکنیم که درون آن تراشه ای هوشمند جاسازی نشده باشد.

به جهت اینکه دانش آموزان، دانشجویان و کلیه علاقه مندان به حوزه اینترنت اشیاء بتوانند خود را با این سیر پر سرعت فناوری همگام نمایند؛ بر آن شدیم که ضمن طراحی بستر سخت افزاری مناسب جهت آموزش و پیاده سازی عملی ایده‌های IoT ، درس‌نامه‌هایی را جهت تسهیل و روشمند شدن فرایند آموزش و یادگیری آماده نماییم.

هدف از درس‌نامه‌ای که پیش روی شماست، آموزش مقدمات برنامه‌نویسی برای سخت افزار، یا همان برنامه‌نویسی سیستم‌های نهفته(Embedded) است. هر چند که تلاش کردیم تا حد ممکن مفاهیم علمی و برنامه‌نویسی مورد نیاز را شرح دهیم؛ اما آشنایی با برنامه‌نویسی به زبان C/C++ و آشنایی مقدماتی با مدارات الکترونیکی و مفاهیم مربوطه و همچنین آشنایی نسبی با مفاهیم شبکه و پروتکل‌های ارتباطی به عنوان پیشنیاز یا همنیاز این درس‌نامه توصیه می‌شود.

کیت سخت افزاری که در کنار این درس‌نامه ارائه شده یک بستر ماژولار و منعطف است که در مراحل بعدی فرایند آموزش نیز در کنار شما خواهد بود و تنها با تهیه ماژول‌های جدید می‌توانید درس‌نامه‌های بعدی را نیز بگذرانید. همچنین این کیت را می توانید در پروژه‌های واقعی و عملی خود مورد استفاده قرار دهید و ایده‌های شما دیگر در حد یک نمونه آزمایشگاهی باقی نخواهند ماند.

اگر اهل تحقیق و مطالعه هستید این درس‌نامه را می‌توانید به صورت خود آموز نیز استفاده کنید.

سلب مسئولیت

در درس‌ها و سخت افزارهای مورد استفاده در این درس‌نامه سعی شده تدابیر امنیتی تا حد زیادی در نظر گرفته شود تا محیطی امن برای آموزش‌پذیر و هنرجو فراهم شود تا مخاطبین از گستره سنی و دانشی وسیع‌تری بتوانند فرایند آموزش را طی کنند. به طور مثال برای کار با کیت سخت افزاری نیاز به هیچ گونه استفاده از هویه وجود ندارد و ولتاژهای مورد استفاده در بازه کمتر از ۱۲ ولت می‌باشند.

اما همواره توصیه اکید می‌شود که لحاظ نمودن تدابیر امنیتی بر عهده مخاطب می‌باشد. توصیه می‌شود دانش‌آموزان و افراد کم سن و سال حتماً در حضور فرد بزرگسال و مربی این دوره را بگذرانند.

استفاده از کیت سخت افزاری در محیط غیر آزمایشگاهی نیازمند به داشتن مهارت و تجربه‌ی کافی است. لذا بی گدار به آب نزنید و از افراد با تجربه کمک بگیرید و همواره ملاحظات امنیتی را رعایت نمایید.

شرکت سازنده کیت سخت افزاری هیچ گونه مسئولیتی در قبال حوادث و خسارات ناشی از استفاده از این محصولات ندارد. لذا خریدار و استفاده کننده خود مسئول بکارگیری درست و ایمن از این محصولات می‌باشد و مسئولیت هرگونه حادثه و یا خسارت احتمالی بر عهده وی می‌باشد.

معرفی

آیا تابحال به خاموش و روشن کردن چراغ اتاقتان با تلفن همراه فکر کرده اید؟ یا کولری که متناسب با دمای مدنظر شما تنظیم شود؟ یا گلدانی که به طور خودکار از گیاهان شما مراقبت کرده و زمانی که آنها به آب نیاز دارند را اطلاع بدهد؟ و یا هزاران هزار ایده‌ی دیگر؟ آردوینو ابزاری برنامه پذیر است که امکان عملی کردن ایده‌های مارا به آسان‌ترین شکل ممکن، فراهم می‌کند. بسته اینترنت اشیا ProMake مبتنی بر آردوینو، مجموعه سخت افزارهای مورد نیاز را به صورت ماژول‌های تست شده در اختیار شما قرار می‌دهد و برای شما امکان پیاده‌سازی گستره‌ی وسیعی از پروژه‌ها را در کوتاه‌ترین زمان ممکن فراهم می‌کند. لذا دیگر نیازی به یادگیری طراحی برد، خرید جداگانه حسگرها برای کاربردهای خاص، سیم‌کشی میان قطعات و... ندارید.

آردوینو، دارای یک نرم‌افزار متن باز اختصاصی برای برنامه‌نویسی بردهای خود می‌باشد و بنابر این مستندات و آموزش‌های متنی و تصویری فراوان به زبان‌های مختلف و با سطح دانش‌های مختلف و همچنین نمونه کدها و پروژه‌های پیاده‌سازی شده متعددی برروی اینترنت برای Arduino وجود دارد. در این سند سعی شده است توضیحات و آموزش‌هایی در جهت آشنایی با مفاهیم پایه سخت افزار و نرم افزار تهیه و ارائه شود.

محتویات بسته سخت افزاری

ProMake Arduino Nano Kit	بستر قرارگیری ماژول آردوینو و ماژول‌های ProMake در کنار یکدیگر
ProMake Sensor TAG	ماژول دارای سنسورهای دما، رطوبت و نور
ProMake GAS MQ	ماژول شناسایی وجود انواع گاز در محیط
ProMake WiFi ESP12	ماژول ارتباط با شبکه‌های WiFi
ProMake 10A Relay 1CH	ماژول رله یک کاناله
Arduino Nano+USB Cable	برد آردوینو به همراه کابل USB جهت اتصال به رایانه[1]
OLED 0.91 Inch	نمایشگر OLED با ارتباط I2C و اندازه ۰٫۹۱ اینچ[2]

کریر بورد ProMake Arduino Nano Kit

کریر بورد ماژولار آردوینو نانو ProMake اولین و تنها برد توسعه‌ای Arduino Nano طراحی و ساخته شده در کشور عزیزمان ایران است که می‌تواند میزبان ماژول آردوینو نانو شما در کنار سه ماژول ProMakeباشد. با داشتن "ProMake Arduino Nano Kit" قادر خواهید بود با استفاده از ماژول‌های ۱۰۰% تست شده‌ی ProMakeبدون نیاز به هیچ گونه لحیم کاری و یا طراحی مدارات الکترونیکی سخت افزار مطلوب پروژه خود را در اختیار داشته باشید. پس با خیال راحت مستقیماً به سراغ نوشتن کدهای پروژه خود بروید و آنها را برروی ماژول Arduino Nanoخود بارگذاری نمایید و شاهد نتایج آن باشید.

ماژول Arduino Nano از بردهای الکترونیکی کوچک و پرطرفدار است که کاربردهای متنوعی دارد. اما هنگام کار با این ماژول باید با استفاده از برد آموزشی (breadboard)مدار مورد نیاز خود را آماده کنیم که مشکلات زیر را به همراه دارد:

- نیاز به سیم کشی که بخش قابل توجهی از زمان با ارزش پروژه را به ساخت و راه اندازی مدار معطوف خواهد کرد.
- یک اتصال اشتباه می‌توان باعث سوخت و خراب شدن آردوینو یا دیگر المان‌های با ارزش مدار بشود که تاخیر زمانی و هزینه اضافی به پروژه تحمیل می‌کند.
- با بزرگ شدن مدار، سیم کشی‌ها شلوغ و درهم برهم می‌شود و با هر دست بردن و تغییر در مدار احتمال قطع شدن اتصالات یا به هم ریختن ناخواسته مدار وجود دارد.

[1] بسته به نوع و قیمت کیتی که تهیه کرده اید، ممکن است این اقلام در بسته کیت شما موجود نباشد، اما برای استفاده از این درسنامه و کیت به آنها نیاز خواهید داشت. لذا می‌بایست خودتان نسبت به تهیه آنها اقدام کنید.

[2] بسته به نوع و قیمت کیتی که تهیه کرده اید، ممکن است این اقلام در بسته کیت شما موجود نباشد، اما برای استفاده از این درسنامه و کیت به آنها نیاز خواهید داشت. لذا می‌بایست خودتان نسبت به تهیه آنها اقدام کنید.

- با بروز هر یک از مشکلات فوق و صرف زمان برای عیب‌یابی و رفع اشکال، زمان باقی مانده برای برنامه نویسی پروژه محدودتر می‌شود که باعث افت کیفیت محصول می‌شود.
- احتمال کار نکردن سخت افزار در هنگام تحویل(دمو) پروژه به علت قطعی یا شل شدن اتصالات بسیار زیاد است.
- امکان استفاده از محصول در خارج از آزمایشگاه وجود ندارد و نمی‌توان به راحتی در محیط واقعی محصول را تست نمود و عملکرد آن را مشاهده نمود.

این مشکلات باعث شده که برخی ترجیح دهند زیر بار طراحی مدار چاپی (PCB)و هزینه‌های زیاد ساخت و رفع ایرادات بروند.

ما با طراحی "کریر بورد ماژولار آردوینو نانو پرومیک" این مشکلات را حل کردیم و شما را از شر breadboardو سیم کشی‌ها درهم و برهم و هزینه‌های زیاد ساخت مدار چاپی نجات دادیم! تا با صرف کمترین هزینه و زمان، قابل اتکاترین سخت افزار برای شروع کار و حتی تست در میدانی (Field Test)در اختیار شما باشد.

با توجه به حذف پیچیدگی‌های طراحی سخت افزار، همه علاقه‌مندان به تکنولوژی با هر سطحی از دانش (از دانش آموزان گرفته تا دانشگاهیان و افراد حرفه‌ای) می‌توانند به راحتی از این کریر بورد برای انجام پروژه‌های خود و یادگیری اینترنت اشیاء بهره ببرند. از این محصول می‌توان حتی برای اجرا پروژه‌های کوچک و متوسط هوشمندسازی نیز استفاده کنید.

با استفاده از ماژول‌های ProMake در کنار "کریر بورد ماژولار آردوینو نانو پرومیک" می‌توان قابلیت‌های زیادی از قبیل:

- بسترهای ارتباطی متنوع (مثل LoRa ، WiFi، Ethernet، GSMو ...)
- سنسورهای متنوع (مثل دما، رطوبت، فشار، نور و ...)
- عملگرهای متنوع(مثلRelay ، Servo Motor، DC Motorو...)

را به سرعت به برد Arduino افزود.

در طراحی "ProMake Arduino Nano Kit" پورت‌های توسعه‌ای QWIIC و Grove برای گسترش ارتباط با ماژول‌ها و سنسورهای I2C در نظر گرفته شده است تا امکان افزایش سنسورهای متصل به کریر بورد فراهم شود .همچنین روی برد یک ماژول DC-DC نیز در نظر گرفته شده است تا امکان استفاده از تغذیه‌های DC از 7تا ۱۴ ولت در پروژه‌های مختلف به راحتی فراهم شود. این ویژگی جذاب برای

کاربردهایی مثل ردیاب خودرو و سایر مکان‌هایی که تنها تغذیه بالای ۵ ولت موجود است، بسیار مفید خواهد بود.

با استفاده از "کریر بورد ماژولار آردوینو نانو پرومیک" می‌توانید ماژول Arduino خود را به یک کیت توسعه اینترنت اشیاء تبدیل کنید و به راحتی یک Sensor Node یا یک ربات متصل به اینترنت بسازید و آن را از دور کنترل کنید.

مشخصات فنی

- پشتیبانی از انواع ماژول‌های Arduino Nano استاندارد
  - Arduino NANO RP2040
  - Arduino NANO 33 IOT
- امکان استفاده از سه ماژول ProMake به صورت هم زمان
  - ماژول شماره ۱ به علت محدودیت پین های Arduino ارتباط UART ندارد.
- دو عدد RGB LED برروی برد
- Buzzer برروی برد
- کلید فشاری
- کانکتورهای توسعه ای QWIIC و Grove(I2C)
- پشتیبانی از صفحه نمایش I2C OLED
- روش های تامین تغذیه :
  - کانکتور USB Type-C روی برد
  - USB Port ماژول آردوینو
- حداکثر جریان تغذیه LDO ۳٫۳ ولت روی برد: ۶۰۰mA
- ابعاد: 90mm x 70mm x 10mm

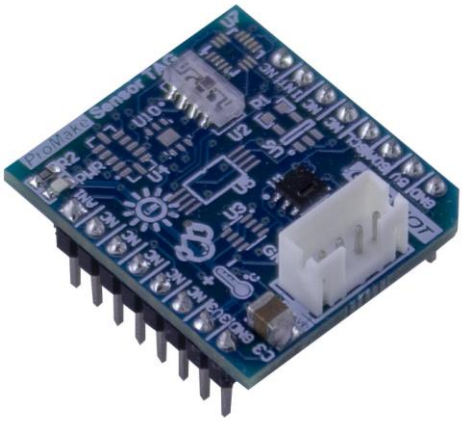
ماژول ProMake Sensor TAG

ماژول ProMake Sensor Tag قابلیت نصب سنسورهای مختلفی را دارد. مدل پایه با قابلیت اندازه گیری پارامترهای دما ، رطوبت و نور محیط ارایه می شود. در مدل پیشرفته امکان اندازه گیری شتاب حرکت ، فشار هوا ، فاصله سنج با تکنولوژی TOF نیز قابل ارایه می‌باشد.ارتباط ماژول با سیستم میزبان از طریق باس رابط I2C می باشد که می تواند به راحتی توسط سیستم‌های مختلف مورد استفاده واقع شود

این ماژول رطوبت را در محدوده ۰ تا 100 %RH و دما را در محدوده 40- درجه سانتیگراد تا 125+ درجه سانتیگراد با دقت به ترتیب ± 3 % RH و ±0.3 °C اندازه گیری می‌کند. چیپ SHT20 در حین کار، مقدار کمی انرژی مصرف می‌کند و مقادیر اندازه‌گیری شده را از طریق واسط I2C در اختیار پردازنده مرکزی قرار می‌دهد.

ماژول ProMake Sensor Tag دارای سنسور نور محیطی VEML7700 با دقت بالا (ALS)با رابط I2C است. این سنسور از چندین فناوری اختصاصی برای اطمینان از اندازه گیری دقیق شدت نور با پاسخ طیفی بسیار نزدیک به چشم انسان استفاده می کند. این سنسور با استفاده از یک دیود نوری حساس، تقویت کننده کم نویز و یک مبدل A/D با دقت 16بیتی، می‌تواند داده‌ها را مستقیماً بدون نیاز به محاسبات پیچیده ارائه دهد. محدوده دینامیکی برای سنسور نور محیط بسیار وسیع است، از ۰ لوکس تا حدود 167Kلوکس را شامل می‌شود. محدوده دینامیکی بالا همراه با پاسخ خطی به منابع مختلف نور، به این سنسور اجازه می‌دهد تا در پشت شیشه تیره یا پانل‌های ساخته شده از مواد نیمه شفاف دیگر قرار گیرد.

مشخصات فنی



- دارای دو حسگر اصلی
  - دما و رطوبت
  - نور
- دارای گرید صنعتی(Industrial)
- ولتاژ کاری: ۳٫۳ ولت
- دارای LED نشانگرPWR
- ابعاد: 25mm x 28mm x 10mm
- پروتکل ارتباطیI2C
- دارای کانکتورGrove برای ارتباط و تغذیه بیرون برد میزبان
- قابل استفاده برروی بردهای آموزشی(breadboard)
- قابل استفاده برروی کلیه اسلات‌های تمام کریر بوردهای آموزشیProMake

ماژول ProMake GAS MQ

ماژول ProMake GAS MQ برای تشخیص گازهای مختلف از سری سنسورهای MQ استفاده می‌کند که توانایی تشخیص انواع مختلف گازهای CO, CO2, NH3, قابل اشتعال و ... را دارد. بروی این ماژول سنسور MQ2 بصورت پیش فرض ارائه می‌گردد که برای تشخیص گازهای قابل اشتعال مناسب می‌باشد.

ماده حساس سنسور گاز SnO2 است که در هوای پاک رسانایی کمتری دارد .هنگامی که گاز قابل احتراق مورد نظر وجود داشته باشد، رسانایی حسگر همراه با افزایش غلظت گاز بیشتر می‌شود. رسانایی با افزایش سطح گاز قابل احتراق افزایش می‌یابد. محدوده تشخیص سنسور 200ppm-10000ppm است. برای کالیبره کردن سنسور در محیطی که از آن استفاده می‌کنید، ماژول دارای یک پتانسیومتر کوچک است که به شما امکان می‌دهد مقاومت بار مدار سنسور را تنظیم کنید. دقت کنید که برای کالیبراسیون دقیق، سنسور باید پیش گرم شود (پس از روشن شدن، بیش از ۲۴ ساعت طول می‌کشد تا به دمای مناسب برسد)

ماژول ProMake GAS MQ به همراه سنسور MQ2 که گازهای LPG، بوتان، پروپان، متان، الکل، هیدروژن و دود را تشخیص می‌دهد، ارائه می‌شود .لذا از این سنسور در دستگاه تشخیص نشت گاز در بازارهای مصرف کننده و صنعت می‌توان بهره برد. این حسگر دارای حساسیت بالا و زمان پاسخگویی سریع است .حساسیت را می‌توان با پتانسیومتر تنظیم کرد. خروجی سنسور متناسب با چگالی گاز است. برای خواندن داده‌های این سنسور خروجی آنالوگ به برد توسعه میزبان مستقیم ارسال می شود و درضمن با استفاده یک قطعه ADC داده دیجیتال معادل خوانش از طریق I2C در اختیار میکروکنترلر قرار می‌گیرد.

مشخصات فنی

- پشتیبانی از کلیه سنسورهای سری MQ
- دارای کانکتور جهت تعویض آسان سنسور
- دارای پتانسیومتر جهت کالیبراسیون
- دارای خروجی آنالوگ برای ارائه به برد توسعه میزبان
- دارای قطعه ADC با اینترفیس I2C
- دارای LED نشانگر PWR
- ولتاژ کاری: ۵ ولت
- ابعاد: 25mm x 28mm x 30mm
- قابل استفاده برروی بردهای آموزشی(breadboard)

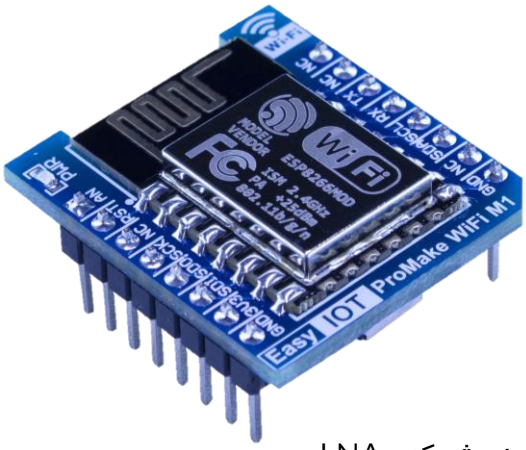
ماژول ProMake WiFi ESP12

ماژول ProMake WiFi ESP12 یک راه حل کامل ارتباط شبکه بی سیم کامپیوتری (WiFi)بر پایه ماژول معروف ESP8266 می‌باشد. این ماژول از استاندارد IEEE802.11 b/g/n پشتیبانی می‌کند و دارای قابلیت Wi-Fi Direct (P2P)و soft-AP است .این ماژول پشته پروتکلی TCP/IP را به طوری کامل پیاده‌سازی نموده، لذا می‌توانید از آن برای ارتباط دادن میکروکنترلر با شبکه WiFi استفاده کنید.

این ماژول از طریق اینترفیس سریال UART با میکروکنترلر برد میزبان ارتباط برقرار می‌کند و می‌توانید از طریق ارسال دستورات AT Command آن را کنترل نموده و فرامین لازم را برای برقرار ارتباط و ارسال و دریافت داده‌ها برای آن بفرستید.

این ماژول برای کار با منبع تغذیه ۳٫۳ ولت طراحی شده است و مصرف انرژی در حالت آماده به کار آن کمتر از ۱٫۰ میلی‌وات برای پروژه‌های اینترنت اشیا و دستگاه‌هایی که به عمر باتری طولانی نیاز دارند، ایده‌آل است. قیمت بسیار مناسب این ماژول از دلایل محبوبیت بالای آن است.

مشخصات فنی



- دارای هسته ESP8266
- پشتیبانی از استاندارد IEEE802.11 b/g/n
- دارای پشته پروتکل TCP/IP
- نمایشگر LED برای تغذیه
- ولتاژ کاری: ۳٫۳ ولت
- توان آماده بکار کمتر از ۱ میلی وات
- دارای آنتن از جنس PCB
- دارای سوئیچ TR ، بالن RF ، تقویت کننده و تطبیق دهند شبکه و LNA
- ابعاد: 25mm x 28mm x 10mm
- قابل استفاده برروی بردهای آموزشی(breadboard)

ماژول ProMake 10A Relay 1CH

ماژول ProMake 10A Relay 1CH یک رله ۱۰ آمپر در اختیار کنترلر میزبان قرار می‌دهد که می‌تواند رنج وسیعی از تجهیزات و دستگاه‌های خانگی و حتی صنعتی را کنترل نماید. استفاده از رله ۱۰ آمپر امکان کنترل انواع تجهیزات مثل موتور و پمپ‌ها را فراهم می‌کند و MCU براحتی و ایمن قادر به کنترل آنها خواهد بود. البته برای بارهای با جریان بالاتر می‌توان براحتی خروجی رله را به عنوان تحریک کنتاکتور مناسب درایو تجهیزات جریان بالا قرار داد و در پروژه‌های صنعتی از این ماژول استفاده کرد.

با توجه به استفاده از ترانزیستور در مسیر فرمان رله، کمترین جریان ممکن از پایه کنترلر برای درایو رله‌ها کشیده خواهد شد. این ماژول را می‌توان در کریر بوردهای آردوینو و رزبری و همچنین روی برد آموزشی نصب نمود و پروژه‌های مختلفی را با آن انجام داد. اصولا در دنیای هوشمندسازی یکی از مهمترین نیازمندی‌ها کنترل پاور ورودی می‌باشد، که با توجه به استفاده از برق AC لازم است رعایت احتیاط و دقت کافی در بکارگیری ماژول انجام شود تا پروژه بدون خطر و در ایمنی کامل با موفقیت انجام شود.

توجه : در استفاده از این ماژول لازم است به جریان مصرفی مداری که قصد کنترل آن را داریم و حداکثر جریان قابل تحمل رله توجه ویژه داشته باشیم تا رله ها دچار سوختگی و تخریب نشوند. سعی کنید همیشه یک حاشیه اطمینان در انتخاب رله در نظر بگیرید و جریان قابل تحمل رله را مقداری بیشتر از جریان مصرف کننده انتخاب کنید.

هشدار: استفاده از رله در مداراتی که جریان مصرفی آنها بالاتر از تحمل رله باشد می‌تواند خطرات و خسارات جدی مثل مرگ، آتش سوزی و ... را به دنبال داشته باشد. در زمان اعمال ولتاژ AC به رله، به برد دست نزنید.

مشخصات فنی

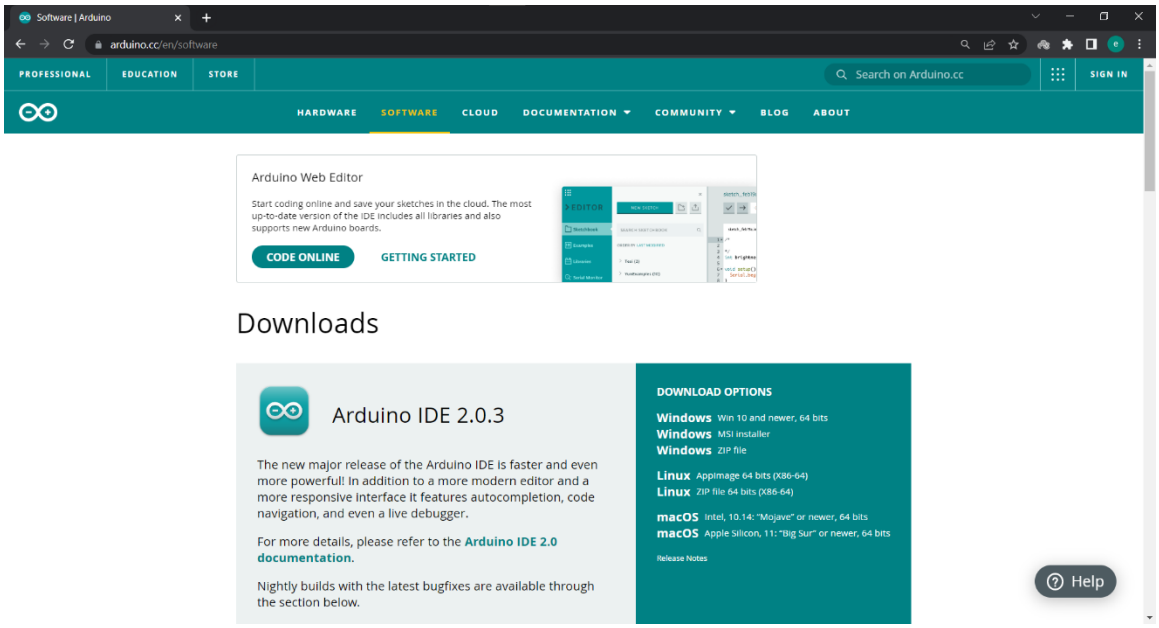
- امکان سوئیچ بارهای تا ۲۵۰ vAC یا ۳۰ vDC
- حداکثر جریان قابل کنترل توسط رله تا ۱۰ A
- دارای LED وضعیت رله و تغذیه ماژول
- ولتاژ کنترل رله: ۵ ولت
- اینترفیس GPIO
- دارای کانکتور ترمینال پیچی برای بخش کنترل تجهیزات
- دارای مدار فرمان ترانزیستوری برای عدم ارتباط با پایه کنتلر و حداقل نمودن جریان فرمان
- ابعاد: 25mm x 28mm x 25mm
- قابل استفاده برروی بردهای آموزشی (breadboard)

برنامه نویسی آردوینو

محیط برنامه نویسی آردوینو

نصب نرم افزار آردوینو در ویندوز

برای شروع باید نرم افزار IDE آردوینو را داشته باشید. می‌توانید با مراجعه به سایت آردوینو نرم‌افزار را دانلود کنید. خوشبختانه این نرم‌افزار برای سیستم عامل‌های مختلف موجود است. برای ویندوز می‌توانید نسخه نصبی و یا نسخه فشرده را دانلود کنید. پیشنهاد می‌شود از نسخه نصبی استفاده کنید تا همزمان، درایورهای USB هم نصب شود. در صورتی که نسخه فشرده را دانلود کنید، باید درایور مربوطه را نیز به صورت دستی نصب کنید.



تابع‌های اصلی

در آردوینو به هر برنامه ، یک Sketch یا طرح گفته می‌شود. زبان کدنویسی برای آردوینو زبان C و C++ است که برای راحتی کار با آن، تا حد زیادی روش مند و ساده‌سازی شده است. یک برنامه آردوینو حداقل شامل دو تابع void setup و void loop است. محتوای هر کدام از این توابع بین دو آکولاد تعریف می‌شود.

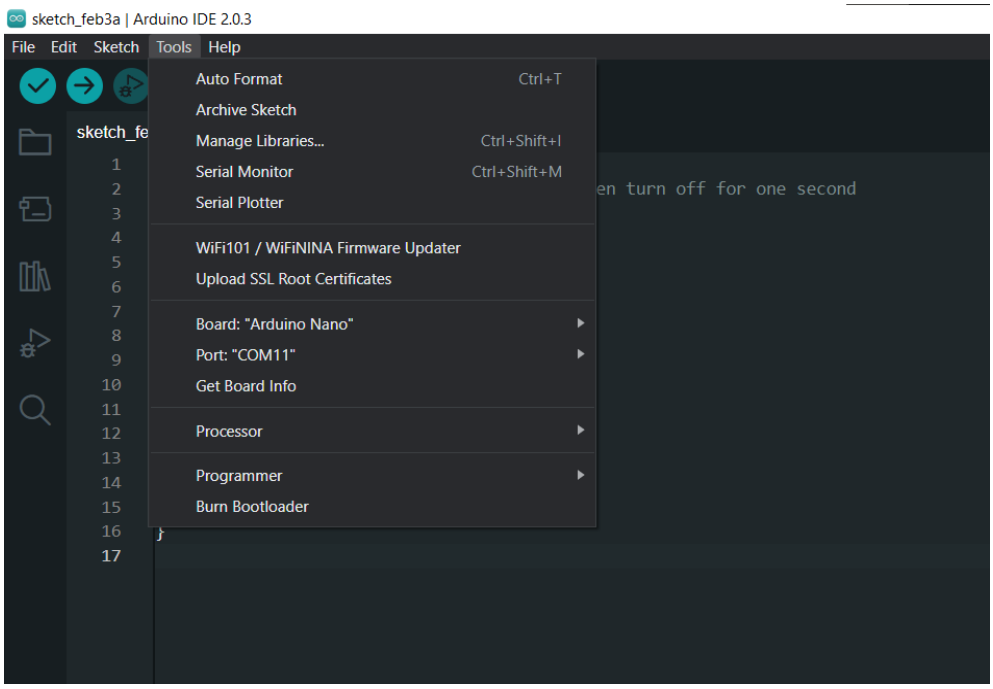
تابع setup مجموعه‌ای از کدها را در بر می‌گیرد که هرگاه آردوینو به تغذیه متصل شود یا کلید Reset فشرده شود، تنها یک بار در ابتدای کار اجرا می‌شوند. از این تابع برای آماده‌سازی کتاب‌خانه‌ها، تعریف و مقداردهی اولیه به متغیرها و پیکربندی پایه‌ها استفاده می‌شود.

تابع loop در یک حلقه‌ی بی‌نهایت مدام فراخوانی می‌شود، در نتیجه کدهای درون آن به صورت تکراری تا زمانی که تغذیه برقرار است، اجرا می‌شوند و برای کنترل فعال آردوینو از این تابع استفاده می‌شود.

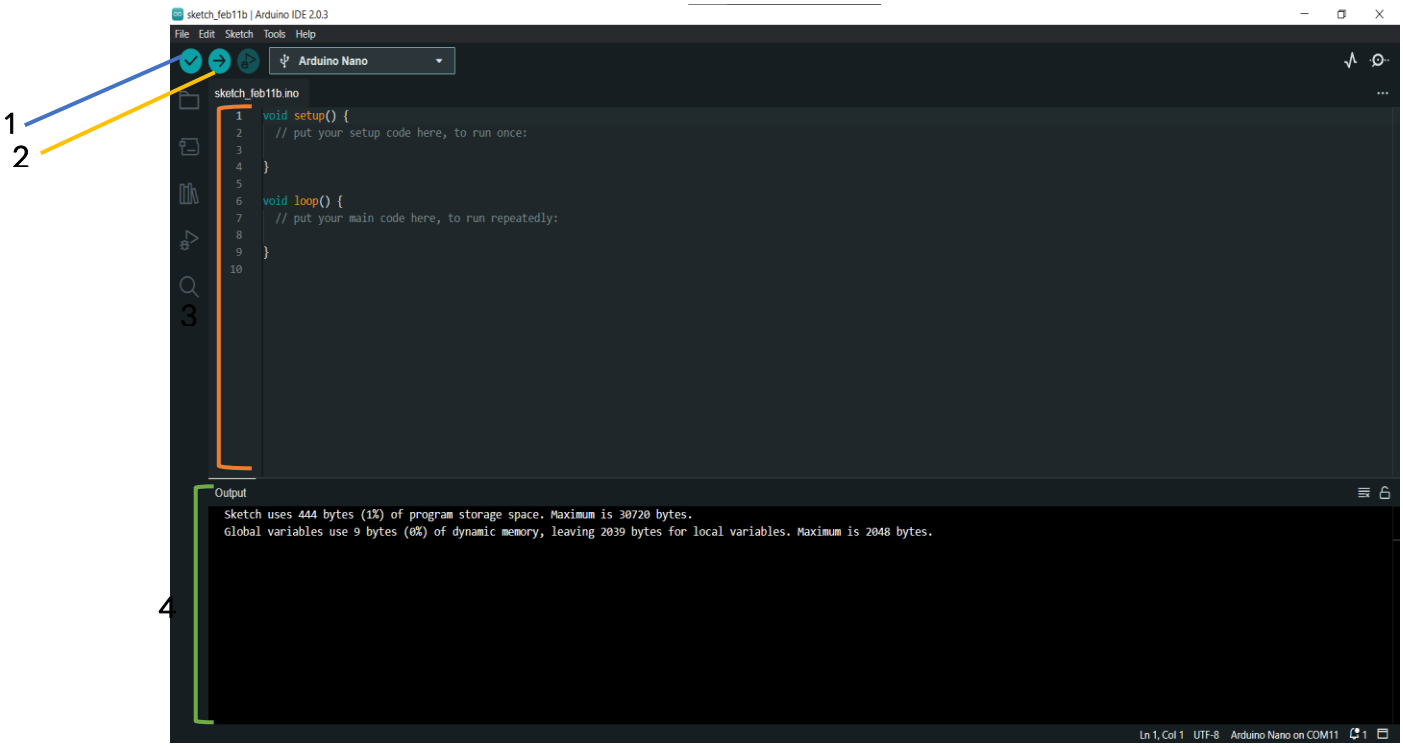
```
void setup() {
// کدهایی که کافیست یکبار در ابتدای کار اجرا شوند
}
void loop() {
 // کدهای اصلی برنامه که تا بینهایت تکرار میشوند
}
```

انتخاب پورت و بورد

هر زمان که یک بورد آردوینو را به رایانه متصل می‌کنید، قبل از هر چیز باید برد و پورت مورد استفاده را به برنامه معرفی کنید. برای این منظور از مسیر Tools → Board مدل و از مسیر Tools → Port پورتی که آردوینو به آن متصل شده است را انتخاب کنید.



دکمه‌ها و بخش‌های محیط برنامه



۱-دکمه (Verify): این دکمه برای کامپایل و شناسایی خطاهای برنامه استفاده می‌شود.

۲-دکمه (Upload): این دکمه برای آپلود کدهای کامپایل شده بر روی برد آردوینو استفاده می‌شود.

۳-بخش (Sketch): این مکان برای نوشتن کدهای برنامه می‌باشد.

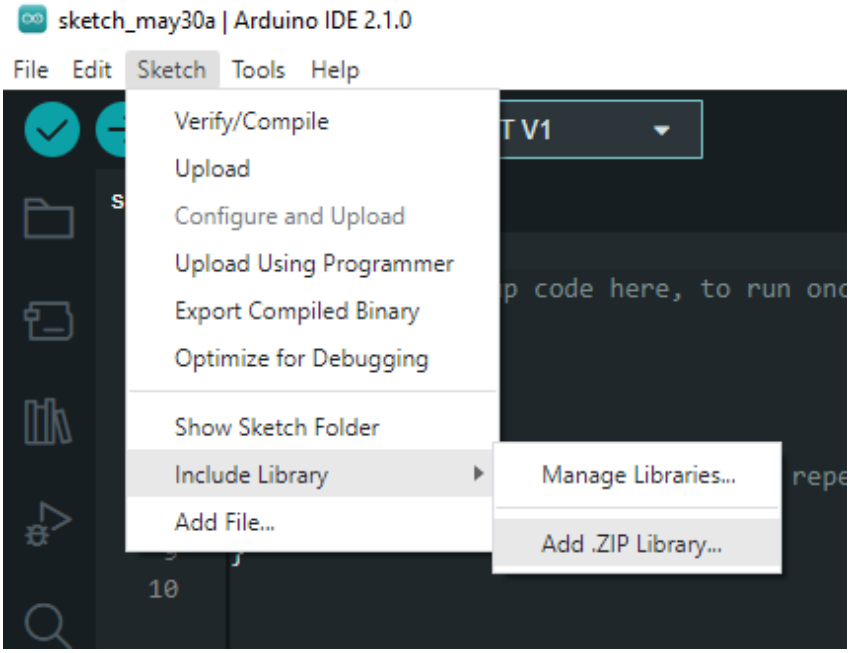
۴-بخش (Output): این مکان عملیات در حال انجام، خطاهای برنامه و اخطارها را به ما نشان می‌دهد.

کتابخانه مورد نیاز

برای استفاده آسان تر از ماژول های ProMake کتابخانه ای نرم افزاری توسط شرکت سازنده تهیه شده است که از آدرس زیر قابل دانلود است.

https://github.com/easyiot-official/ProMake-Modbus-Arduino/archive/refs/heads/main.zip

پس از دریافت فایل زیپ، آن را از منوی Sketch –>Include Library –> Add .ZIP Library در محیط برنامه آردوینو اضافه کنید تا نصب و قابل استفاده شود.


sketch_may30a | Arduino IDE 2.1.0
File
Edit
Sketch
Tools
Help
Verify/Compile
Upload
Configure and Upload
Upload Using Programmer
Export Compiled Binary
Optimize for Debugging
Show Sketch Folder
Include Library
Add File...
Manage Libraries...
Add .ZIP Library...
T V1
p code here, to run onc
repe
10

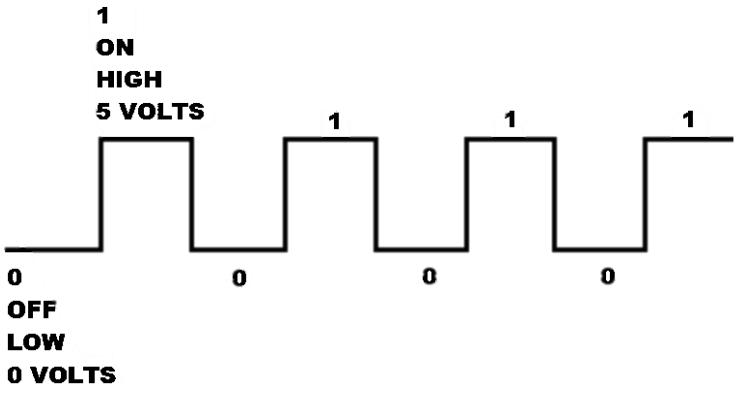
بخش اول: کنترل خروجی‌های دیجیتال

درس اول: چشمک زدن LED روی ماژول آردوینو

پیش نیاز: سیگنال دیجیتال

به طور کلی سیگنال به هر چیزی گفته می‌شود که برای انتقال یک پیام استفاده می‌شود. سیگنال دیجیتال(Digital Signal) سیگنالی است که در مقابل سیگنال آنالوگ (Analog Signal) تعریف می‌شود و داده‌ها را به صورت دنباله‌ای از مقادیر گسسته‌ی مشخص نشان می‌دهد.

بیشتر سیگنال‌های دیجیتال فقط دو سطحِ دامنه (معادل صفر و یک منطقی) دارند، برای مثال در بسیاری از سیستم‌های الکترونیکی از سیگنال دیجیتال با مقدار صفر ولت برای اعلام وضعیت خاموش و مقدار ۵ ولت برای وضعیت روشن استفاده می‌شود، و هیچ عددی بین صفر و ۵ ولت مخابره نمی‌شود.



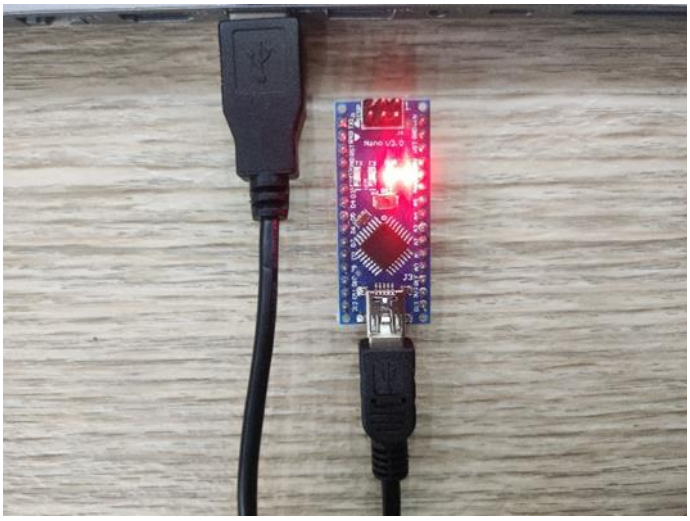
شکل ۱: سیگنال دیجیتال باینری

اقلام مورد نیاز

آردوینو نانو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول آردوبینو نانو را توسط کابل mini USB به کامپیوتر متصل کنید.



کدنویسی و شرح کد

در محیط Arduino IDE کد زیر را نوشته و اجرا نمایید.

```
int ledPin = 13; // متغیری که شماره پایه متصل به ال ای دی را ذخیره می کند
void setup() {
 pinMode(ledPin, OUTPUT); // تنظیم پایه به عنوان خروجی
}
void loop() {
 digitalWrite(ledPin, HIGH); // قراردادن ولتاژ معادل ۱ منطقی برروی پایه
 delay(1000);          // تاخیر ۱۰۰۰ میلی ثانیه ای = معادل ۱ ثانیه
 digitalWrite(ledPin, LOW); // قراردادن ولتاژ معادل ۰ منطقی برروی پایه
 delay(1000);          // تاخیر ۱۰۰۰ میلی ثانیه ای = معادل ۱ ثانیه
}
```

تعریف متغیر و مقداردهی اولیه

```
int ledPin = 13 ; //تعریف متغیر و مقداردهی اولیه به آن
```

متغیر یک روش نام دهی به آدرس‌های حافظه می‌باشد؛ تا در طول برنامه به جای استفاده از آدرس‌های حافظه بتوان از نام متغیرها استفاده نمود. برای مثال می‌توان یک متغیر به نام پین وسیله مدنظر تعریف کرد و شماره پین را در آن متغیر ذخیره نمود و در طول برنامه هرجا نیاز به شماره آن پین وجود داشت به جای شماره پین از آن متغیر استفاده کرد. بدین ترتیب تغییر در برنامه به راحتی اعمال شده همچنین برنامه خواناتر می‌شود.

دقت کنید که نام متغیر با اعداد نمی‌تواند آغاز شود و کوچک یا بزرگ بودن حروف نیز اهمیت دارد. هنگام تعریف، متغیرها می‌توانند توسط یک مساوی مقداردهی اولیه شوند. نکته مهم دیگر این است که همیشه (به جز بعضی موارد خاص) باید در پایان هر خط، نقطه‌ویرگول ";" بگذارید که به معنای پایان سطر است.

نوع متغیر تعیین می کند که چه اندازه از فضای حافظه برای ذخیره سازی داده های درون متغییر باید مورد استفاده قرار گیرد. انواع داده‌ها در آردوینو مطابق جدول زیر است.

Type	Bits	Range
int	16	-32768 to -32767
unsigned int	16	0 to 65535
signed int	16	-31768 to 32767
short int	16	-31768 to 32767
unsigned short int	16	0 to 65535
signed short int	16	-32768 to -32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	-2147483648 to 2147483647
signed long int	32	0 to 4294967295
float	32	3.4E-38 to 3.4E+38
double	64	1.7E-308 to 1.7E+308
long double	80	3.4E-4932 to 3.4E+4932
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127

نوع داده‌های float و double برای ذخیره اعداد اعشاری قابل استفاده هستند.

تابع تعیین جهت پین‌ها

```
pinMode(ledPin, OUTPUT);
//pinMode( وضعیت , شماره پین );
```

از آنجا که پین‌های دیجیتال قابلیت ورودی یا خروجی بودن را دارند، همیشه باید قبل از استفاده، جهت آن پین را مشخص نمود. این کار باید در تابع void setup با استفاده از تابع فوق صورت گیرد. در آرگومان وضعیت می‌توان OUTPUT برای خروجی بودن و INPUT برای ورودی بودن را قرار داد. همچنین اگر وضعیت را INPUT_PULLUP قرار دهیم، پین به عنوان ورودی در نظر گرفته می‌شود و پول آپ داخلی فعال می‌شود تا وضعیت پیش فرض پین مدنظر در صورت عدم اعمال ورودی، ۱ منطقی باشد. در برنامه فوق کنترل روشن و خاموش بودن LED با تنظیم پین مربوطه به صورت خروجی انجام می‌شود.

تابع مقداردهی به پین‌ها

```
digitalWrite(ledPin, HIGH);
digitalWrite(ledPin, LOW);
//digitalWrite( وضعیت , شماره پین );
```

با استفاده از این تابع مقدار دهی پایه‌ها انجام می‌گیرد و دو حالت صفر یا یک به پایه مدنظر اختصاص می‌یابد. آرگومان وضعیت می‌تواند HIGH برای ۱ و LOW برای ۰ باشد. و با HIGH کردن ledPin، ولتاژ بر روی پایه متصل به LED قرار گرفته و آن را روشن می‌کند و با LOW کردن آن، ولتاژ پایه مذکور صفر شده و LED خاموش می‌شود.

تابع تولید تاخیر

```
delay(1000);
//delay(مقدار زمان بر حسب میلی ثانیه );
```

این تابع برای ایجاد تاخیر زمانی استفاده می‌شود و آردوینو به اندازه مدت زمانی که در ورودی تابع تعیین می‌کنیم صبر می‌کند و در این مدت تمام متغیرها را در وضعیت قبلی خود نگه می‌دارد. ورودی این تابع بر حسب میلی‌ثانیه تعیین می‌شود پس تابع کد فوق یک ثانیه تاخیر ایجاد می‌کند.

حال به خوبی می تواند درک کرد که چگونه در برنامه بالا LED یک ثانیه روشن و یک ثانیه خاموش می‌شود. با کمتر/بیشتر کردن میزان تاخیر می‌توان چشمک زدن LED را سریع‌تر/کندتر کرد.

بخش دوم: خواندن ورودی‌های دیجیتال

درس دوم: روشن کردن LED با فشرده شدن کلید فشاری

پیش نیاز

کلید فشاری امکان تولید دو حالت منطقی 1(فشرده شده) یا 0(فشرده نشده) را فراهم می کند. این مقادیر یک سیگنال ورودی دیجیتال را تولید می کنند که قصد داریم در برنامه ای در صورت فشرده شدن کلید LED را روشن و در صورت رها شدن کلید LED را خاموش کنیم.

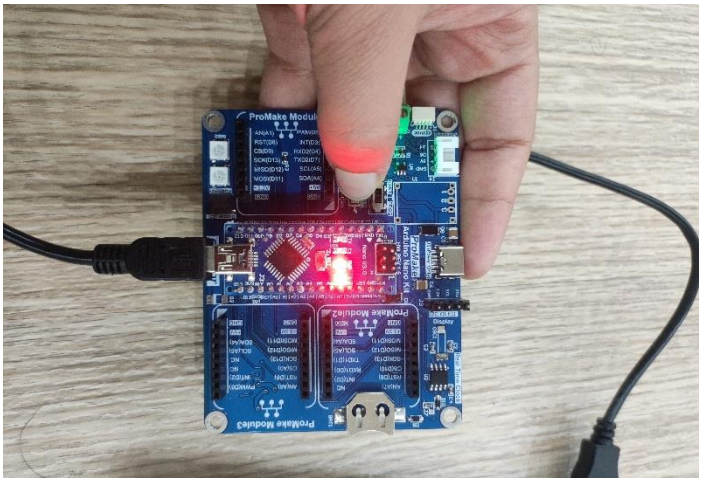
اقلام مورد نیاز

کریر بورد آردوینو نانو ProMake(جهت استفاده از دکمه فشاری روی آن)

آردوینو نانو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول آردوینو نانو را به نحوی که هر یک از پایه‌ها بر روی پایه متناظر خود بر روی کریر بورد قرار گیرد وارد جایگاه مربوطه نمایید. برای تشخیص آسان‌تر جهت ماژول نانو، کانکتور USB آن را با محل نشانه USB CON برروی کریر بورد منطبق نموده و ماژول نانو را وارد کریر بورد نمایید. حال ماژول آردوینو نانو را توسط کابل mini USB به کامپیوتر متصل کنید.



کدنویسی و شرح کد

در محیط Arduino IDE کد زیر را وارد نمایید.

```
// متغیرهای ثابت
int buttonPin = 17; // شماره پایه کلید فشاری
int ledPin = 13;   // شماره پایه LED روی آردوینو

// متغیرهایی که تغییر می کنند
int buttonState = 0; // متغیر وضعیت کلید

void setup() {
 pinMode(ledPin, OUTPUT);  //  تنظیم پین LEDبه عنوان خروجی
 pinMode(buttonPin, INPUT); // تنظیم پین کلید به عنوان ورودی
}
void loop() {
 buttonState = digitalRead(buttonPin); // خواندن پین کلید
 if (buttonState == HIGH) {  // در صورتی که کلید فشرده باشد مقدار آن ۱ یا HIGH است
  digitalWrite(ledPin, LOW); //  روشن شود LED
 } else {
  digitalWrite(ledPin, HIGH); // خاموش شود LED
 }
}
```

مطابق توضیحات درس اول، به منظور تعریف آسان‌تر پایه‌ها و مقداردهی آنها از متغیرها استفاده می‌کنیم. در این کد یک متغیر با نام buttonState تعریف کردیم که در هر لحظه دربردارنده‌ی وضعیت کلید فشاری باشد. و در ادامه نیز با دستور pinMode، پایه متصل به LED به عنوان خروجی و پایه متصل به کلید فشاری به عنوان ورودی تنظیم می‌شود.

خواندن مقادیر دیجیتال پین‌ها

```
buttonState = digitalRead(buttonPin);
//digitalRead(شماره پین );
```

این تابع برای خواندن مقدار دیجیتال پین‌ها به کار می‌رود. آرگومان ورودی شماره پین مربوطه را تعیین می‌کند و خروجی تابع که LOW است یا HIGH حاوی مقدار خوانش شده از پین می‌باشد.

ساختارهای شرطی

```
if (buttonState == HIGH) {       // در صورتی که کلید فشرده باشد مقدار آن ۱ یا HIGH است
  digitalWrite(ledPin, LOW);     //  روشن شود LED
} else {
  digitalWrite(ledPin, HIGH); // خاموش شود LED
}
```

ساختارهای شرطی یا کنترلی برای اجرای یک سری دستورات در صورت صحیح بودن یا نبودن یک شرط استفاده می‌شود. به طور کلی ساختار if ... else به صورت زیر کار می‌کند.

```
if (شرط۱) { // در صورتی که جواب عبارت شرط صحیح یا به عبارتی ۱ شود سراغ انجام دستورات ذیل آن می رود
        // انجام دستورات الف
}
else if (شرط۲) {
        // انجام دستورات ب
}
else { // در غیر این صورت
        // انجام دستورات ج
 }
```

دستور if شرط یا شروط داخل پرانتز را بررسی می‌کند و در صورت درست (true) بودن عبارت، مجموعه دستورات داخل بلوک پس از شرط را اجرا می کند. اما در صورت نادرست(false) بودن عبارت شرطی، در صورت وجود ساختار else if به سراغ عبارت شرطی آن می‌رود و آن را بررسی می‌کند و در صورت صحیح بودن عبارت شرطی، دستورات داخل بلوک مربوطه را اجرا می‌کند. این فرایند تا رسیدن به شرطی که حاصلِ درست(true) داشته باشد، ادامه می‌یابد. در واقع تمامی شرط‌ها یکی یکی به ترتیب تا رسیدن به یک عبارت صحیح چک می‌شوند. هنگامی که نتیجه‌ی چک کردن یک شرط true شد؛ مجموعه کد مربوط به آن اجرا می‌شود و برنامه از بقیه‌ی ساختار if else چشم پوشی می‌کند و به خط بعد از این ساختار می‌رود. اگر هیچ شرطی به جواب true نرسد و بلوک else وجود داشته باشد، کدهای داخل این بلوک اجرا خواهد شد.

انواع عمگرهای منطقی قابل استفاده در عبارات شرطی در جدول زیر وجود دارد.

عملگر	نام	مثال
>	بزرگتر	x>y
>=	بزرگتر یا مساوی	x>=y
<	کوچکتر	x<y
<=	کوچکتر یا مساوی	x<=y
==	متساوی	x==y
!=	نامساوی	x!=y

بخش سوم: تولید فرکانس و ولتاژهای مختلف در خروجی دیجیتال

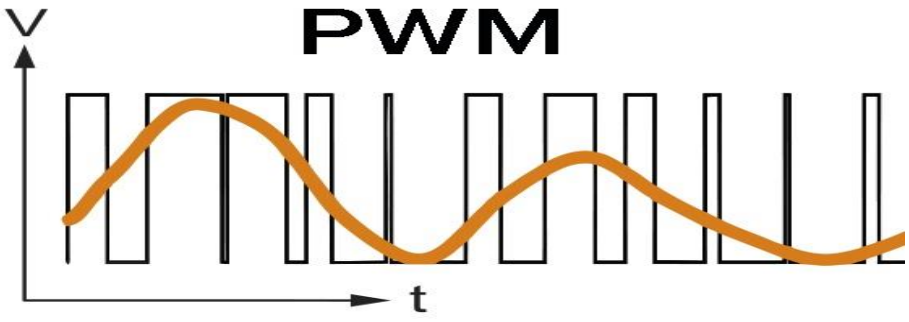
درس سوم: کنترل شدت نور LED روی ماژول آردوینو

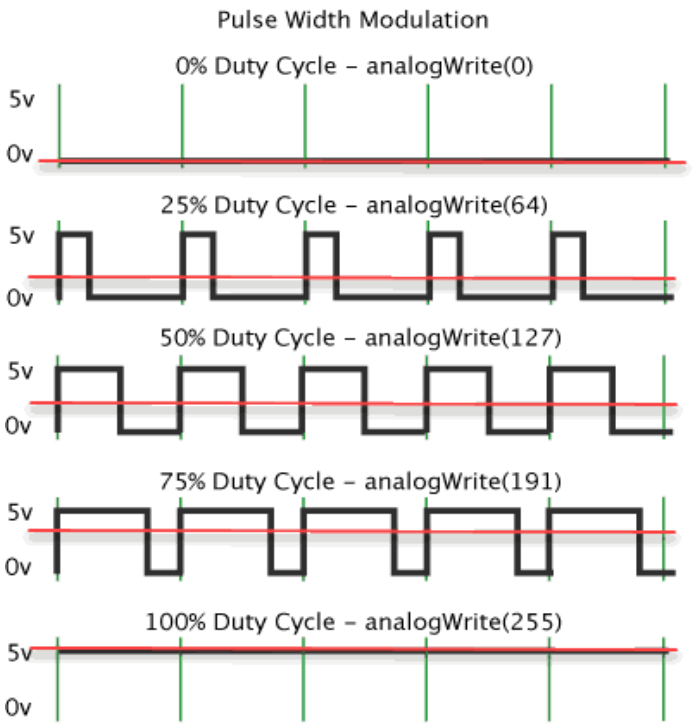
پیش نیاز: سیگنال PWM

PWM یا مدولاسیون عرض پالس روشی برای ایجاد سیگنال آنالوگ توسط یک سیستم دیجیتال است که برای کنترل میزان نور چراغ، دور موتور، کنترل سرو موتور، راه اندازی بازر پسیو و ... استفاده می‌شود.

همان طور که در درس‌های قبل بیان شد خروجی پایه‌های دیجیتال به صورت پالسی و در دو حالت 1 و 0 است که منظور از 1 همان VCC و 0 همان GND است. حال اگر بخواهیم شدت نور یک ال ای دی را کم و زیاد کنیم، عملا با استفاده از ۵ ولت و صفر ولت امکان پذیر نیست. زیرا در یک لحظه ال ای دی خاموش یا در یک لحظه در بالاترین میزان نور خواهد بود، اما به عنوان یک راه کار می‌توانیم مدت زمان 1 بودن سیگنال را کم و زیاد کنیم.

فرض کنیم بخواهیم یک سیگنال ۲ ولت آنالوگ را با یک منبع دیجیتال که می‌توان در ۵ ولت ON و در صفر ولت OFF باشد ایجاد کنیم، برای این کار می‌توان یک PWM که برای ۴۰ درصد زمان‌ها خروجی ۵ ولت تولید می‌کند و در باقی زمان‌ها صفر ولت است تولید کنیم. به درصد زمانی که یک سیگنال PWM مقدار 1 دارد duty cycle گفته می‌شود. اگر دوره تناوب سیگنال PWM به اندازه کافی کوتاه باشد، ولتاژ دیده شده در خروجی همانند یک ولتاژ میانگین به نظر می‌رسد. اگر low دیجیتالی صفر ولت باشد (که معمولا هم هست) آنگاه ولتاژ میانگین را می‌توان با ضرب مقدار high دیجیتال در duty cycle محاسبه کرد، یعنی 5V * 0.4 = 2V.





چهار پارامتر اصلی PWM به قرار زیر هستند:

- TON = ON TIME = عرض پالس= زمانیکه سیگنال در بالاترین حد (HIGH) است.
- TOFF = OFF TIME = زمانیکه سیگنال در پایین‌ترین حد (LOW) است.
- پریود = PERIOD =حاصل جمع زمان TON و TOFF پریود گفته می‌شود.
- چرخه کار = DUTY CYCLE = درصد زمان هنگامی که سیگنال در بالاترین حد (HIGH) است نسبت به مدت زمان پریود.

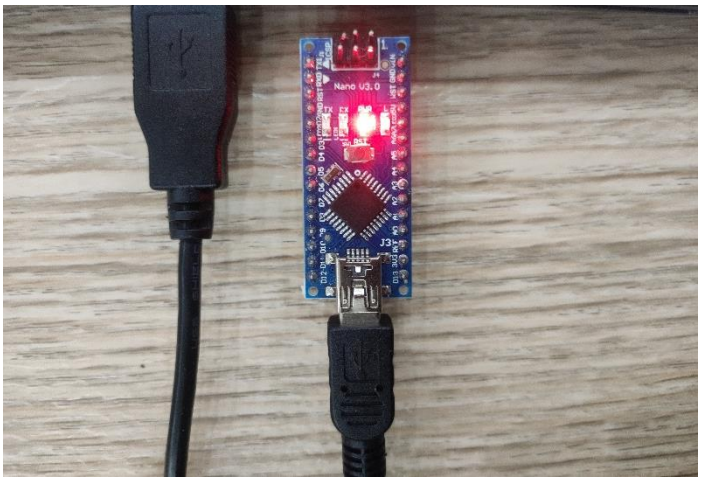
آردوینو Nano دارای ۶ کانال ۸ بیتی PWM می‌باشد. پین‌هایی که با علامت «~» نشانه‌گذاری شده اند از PWMپشتیبانی می‌کنند و پایه‌های ۳ و ۵ و ۶ و ۹ و ۱۰ و ۱۱ هستند و فرکانس پیش فرض PWM آن‌ها ۴۹۰ هرتز است.

اقلام مورد نیاز

آردوینو نانو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول آردوینو نانو را توسط کابل USB به کامپیوتر متصل می‌کنیم.



کدنویسی و شرح کد

ایجاد سیگنال PWM

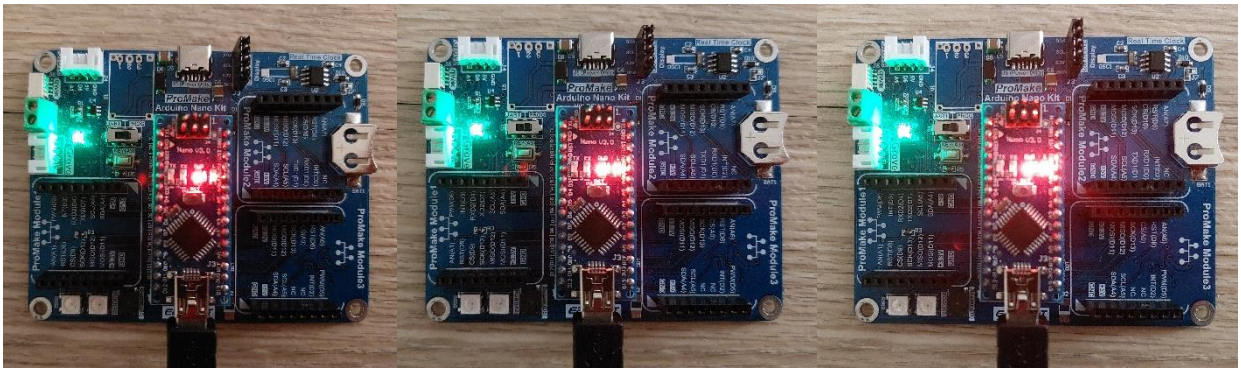
تابع مورد استفاده در آردوینو برای تولید سیگنال PWM دستور analogWrite(Pin, value) است. (دقت کنید که برای تولید سیگنال دیجیتال(DC) از دستور digitalWrite() استفاده می‌کردیم.) در این تابع آرگومان Pin پایه‌ای است که می‌خواهیم روی آن PWM یا سیگنال آنالوگ تولید کنیم و آرگومان value نشان دهنده duty cycle سیگنال تولیدی است که مقداری بین صفر(0% و همیشه خاموش) تا ۲۵۵(۱۰۰% و همیشه روشن) را می‌پذیرد.

اما همانطور که گفته شد LED روی آردوینو با پایه D13 متصل است؛ متاسفانه این پایه از پایه‌هایی که از قابلیت PWM پشتیبانی می‌کنند نمی‌باشد و نمی‌توانیم از تابع analogWrite برروی آن استفاده کنیم. بنابراین اگر بخواهیم شدت نور LED را کنترل کنیم باید شکل موج PWM مورد نظرمان را با صفر و یک کردن از طریق کدنویسی در این پایه تولید کنیم. به این کار در اصطلاح Bit-banging گفته می‌شود. بدین ترتیب در محیط Arduino IDE کد با ساختار زیر نوشته و اجرا می‌کنیم.

```
void setup() {
 pinMode(13, OUTPUT); //  تنظیم پین LEDبه عنوان خروجی
}

int dutyCycle = 50; //متغیر ذخیره کننده طول چرخه کار
void loop() {
```

```
digitalWrite(13, HIGH);        //قراردادن مقدار۱ یا HIGH برروی پایه
delayMicroseconds(dutyCycle);     //تاخیر به مدت چرخه کار
digitalWrite(13, LOW);         //قراردادن مقداره یا LOW برروی پایه
delayMicroseconds(1000 – dutyCycle); //تاخیر به مدت باقی مانده تا ۱ ثانیه
}
```

با تغییر مقدار متغییر dutyCycle می‌توان مدت زمان high یا low بودن شکل موج را تغییر داده و در نتیجه ولتاژ و شدت نور متفاوتی را برای LED به وجود آورد. تصویر نور LED با عدد ۵۰ و ۵۰۰ و ۹۰۰ به شکل زیر است.



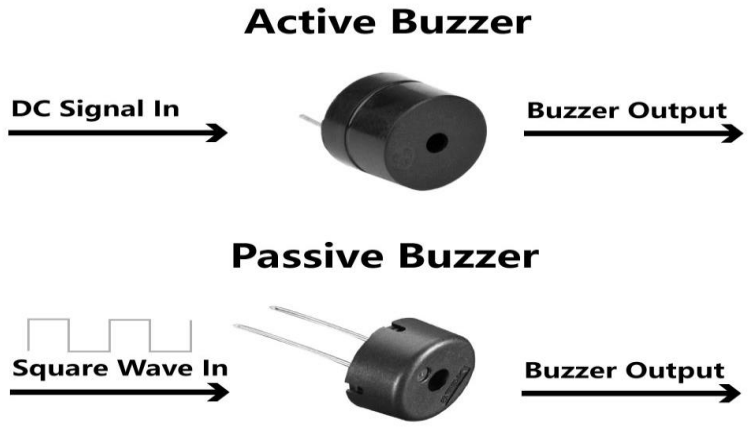
درس چهارم: تولید نوت‌های مختلف موسیقی با بازر روی کریر بورد

پیش نیاز: آشنایی با انواع بازر

بازرها قطعات کوچکی هستند که انرژی الکتریکی را به صدا تبدیل می‌کنند و به دو دسته تقسیم می‌شوند بازرهای فعال [۳] و بازرهای منفعل [۴]. بازرهای فعال بازرهایی هستند که دارای نوسانگر [۵] داخلی هستند و زمانی که برای روشن شدن به منبع تغذیه DC متصل می‌شوند، صدا تولید می‌کنند. این نوع بازر به صورت وسیعی در رایانه‌ها، پرینترها، دستگاه‌های کپی، هشداردهنده‌ها، اسباب‌بازی‌های الکترونیکی، الکترونیک خودرو، تلفن، تایمر و سایر دستگاه‌های الکترونیکی-صوتی استفاده می‌شود.

بازرهای منفعل نوسان گر داخلی ندارند و برای تولید صدا نیاز به سیگنال نوسانی دارند؛ مانند بلندگوهای الکترومغناطیسی. در این نوع بازر به جای تولید یک صدای ثابت، سیگنال ورودی متغیر، صداهای متفاوتی را به وجود می‌آورد و بنابراین برای تولید سیگنال‌های صوتی با تُن‌ها و فرکانس‌های مختلف استفاده می‌شوند. هدف از این درس آشنایی با چگونگی استفاده از این نوع بازر با استفاده از آردوینو است.

بازر استفاده شده در کریر بورد ProMake از نوع Passive یا منفعل می‌باشد. سیگنال موردنیاز آن نیز مطابق توضیحات درس قبل با استفاده از PWM تولید می‌شود.



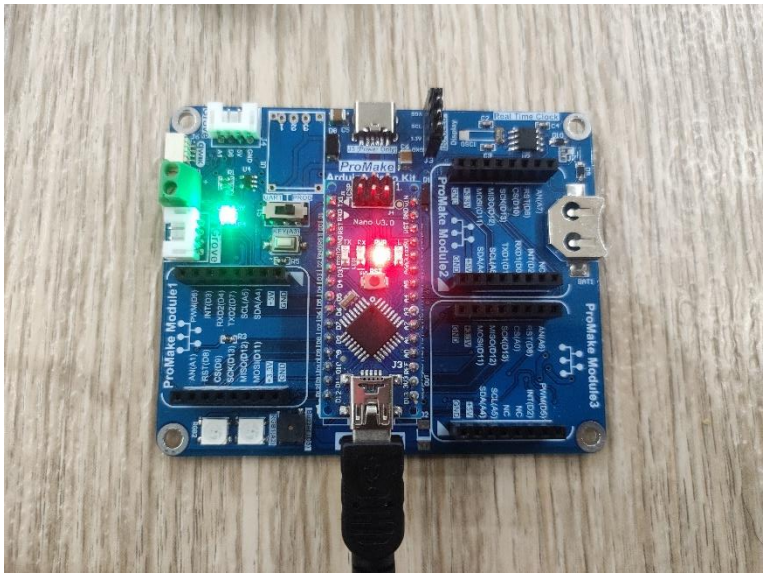
[۳] Active Buzzer
[۴] Passive Buzzer
[۵] Oscillator

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake(برای استفاده از بازر روی آن)
- آردوینو نانو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول آردوینو نانو را با توجه به جهت صحیح بر روی کریر بورد قرار می‌دهیم و توسط کابل USB به کامپیوتر متصل می‌کنیم.



کدنویسی و شرح کد

در محیط Arduino IDE کدی با ساختار زیر نوشته و اجرا می‌کنیم.

```
int BuzzerPin = 5;

void setup() {
        pinMode(BuzzerPin, OUTPUT);
}
void loop() {
        analogWrite(BuzzerPin, 128);
        delay(100);
        analogWrite(BuzzerPin, 0);
        delay(100);
}
```

این برنامه بازر را به تولید صدای بوق غیرممتد وا می‌دارد.

ایجاد سیگنال PWM

```
analogWrite(BuzzerPin,128);
```

همان طور که در درس قبل اشاره شد تابع مورد استفاده در آردوینو برای تولید سیگنال PWM دستور analogWrite (pin, value) است، که Pin پایه‌ای است که می‌خواهیم روی آن PWM یا سیگنال آنالوگ تولید کنیم و value تعیین کننده duty cycle سیگنال تولیدی است که عددی بین صفر (معادل ۰% یا همیشه خاموش) تا ۲۵۵(معادل ۱۰۰% یا همیشه روشن) قابل انتخاب هست.

تابع ریاضی دیگری که می تواند به ما برای محاسبه ی value کمک کند دستور()map می باشد.

تابع ()map یک عدد را از یک محدوده به محدوده دیگر نگاشت می‌کند. به طور مثال یک عدد بین ۱۰ تا ۲۰ را به عددی بین ٤۰ تا ۸۰ تبدیل می‌کند .ساختار این تابع به صورت زیر است.

```
map(value, fromLow, fromHigh, toLow, toHigh);
```

- value: عدد ورودی
- fromLow: مقدار حد پایین بازه‌ی ورودی
- fromHigh: مقدار حد بالا بازه‌ی ورودی
- toLow: مقدار حد پایین بازه‌ی خروجی
- toHigh: مقدار حد بالا بازه‌ی خروجی

راه دیگر برای به صدا درآوردن بازر با استفاده آردوینو نیز استفاده از تابع tone به صورت زیر است.

```
tone (pin, frequency, duration);
```

این تابع سیگنال‌هایی با فرکانس متفاوت با duty cycle پنجاه درصد به مدت زمان تعیین شده تولید می‌کند.

آرگومان pin پایه‌ای که می‌خواهیم سیگنال تولیدی روی آن تولید شود را تعیین کرده و با استفاده از frequency فرکانس سیگنال مدنظر و با duration مدت زمانی که می‌خواهیم سیگنال روی این پایه تولید شود را مشخص می‌کنیم.

آرگومان duration می‌تواند در تابع نوشته نشود تا محدودیت زمانی برای تولید سیگنال وجود نداشته باشد. در این صورت می توان با استفاده از تابع noTone(pin) عملکرد تابع tone را پایان داد.

برنامه زیر به کمک توابع فوق در آردوینو آهنگ happy birthday را پخش می‌کند.

```
// تعریف فرکانس نوت های موسیقی
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
```

```
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
```

```
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
#define REST 0

// این متغیر موسیقی را تند یا کند می کند
int tempo = 140;

// متغیری که پایه متصل به بازر را ذخیره می کند
int buzzer = 5;

// نوت های ملودی به همراه مدت زمان پخش آن ها
// a 4 means a quarter note, 8 an eighteenth, 16 sixteenth, so on
// !!negative numbers are used to represent dotted notes,
// so –4 means a dotted quarter note, that is, a quarter plus an eighteenth!!

// ملودی تولدت مبارک
// برگرفته از سایت https://musescore.com/user/8221/scores/26906
int melody[] = {
 NOTE_C4,
 4,
 NOTE_C4,
 8,
 NOTE_D4,
 –4,
 NOTE_C4,
 –4,
 NOTE_F4,
 –4,
 NOTE_E4,
```

```
-2,
NOTE_C4,
4,
NOTE_C4,
8,
NOTE_D4,
-4,
NOTE_C4,
-4,
NOTE_G4,
-4,
NOTE_F4,
-2,
NOTE_C4,
4,
NOTE_C4,
8,

NOTE_C5,
-4,
NOTE_A4,
-4,
NOTE_F4,
-4,
NOTE_E4,
-4,
NOTE_D4,
-4,
NOTE_AS4,
4,
NOTE_AS4,
8,
NOTE_A4,
-4,
NOTE_F4,
-4,
NOTE_G4,
-4,
NOTE_F4,
```

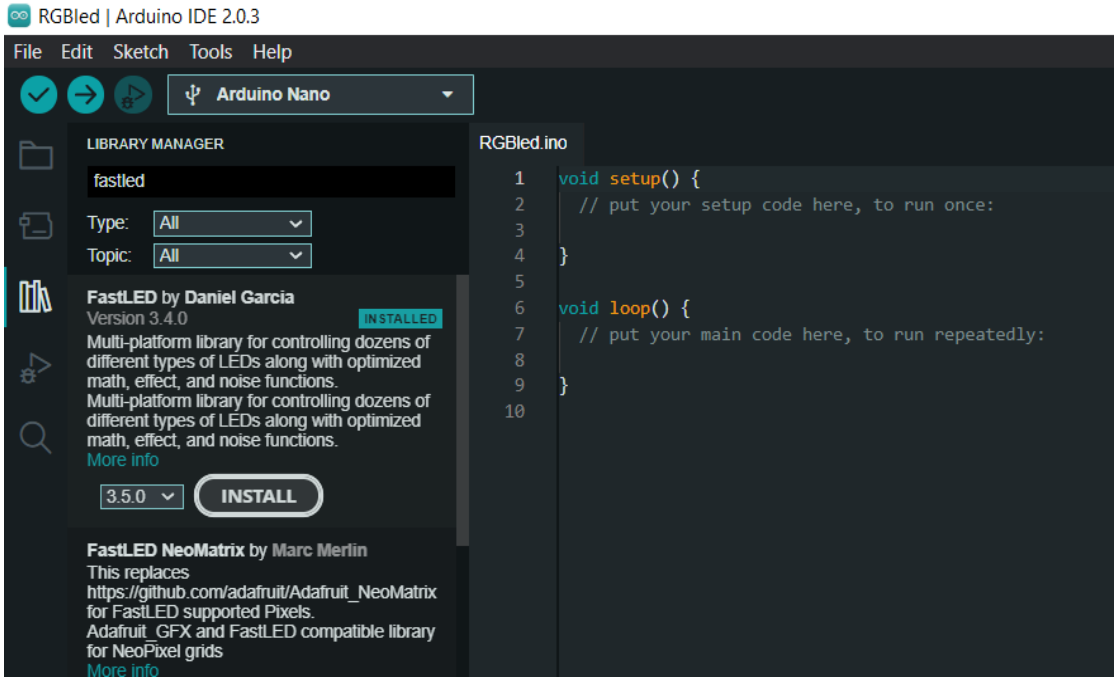
```
 -2,

};

// sizeof gives the number of bytes, each int value is composed of two bytes (16 bits)
// there are two values per note (pitch and duration), so for each note there are four bytes
int notes = sizeof(melody) / sizeof(melody[0]) / 2;

// این متغیرکل مدت زمان یک نوت را محاسبه می کند
int wholenote = (60000 * 4) / tempo;

int divider = 0, noteDuration = 0;

void setup() {
}

void loop() {
 // پیمایش برروی کل نوت های ملودی
 // Remember, the array is twice the number of notes (notes + durations)
 for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
  // محاسبه مدت زمان هرنوت
  divider = melody[thisNote + 1];
  if (divider > 0) {
   // regular note, just proceed
   noteDuration = (wholenote) / divider;
  } else if (divider < 0) {
   // dotted notes are represented with negative durations!!
   noteDuration = (wholenote) / abs(divider);
   noteDuration *= 1.5; // increases the duration in half for dotted notes
  }
  // ما فقط ۹۰ درصد زمان را به پخش نوت می پردازیم و ۱۰ درصد باقی مانده را سکوت می کنیم
  tone(buzzer, melody[thisNote], noteDuration * 0.9);
  // انتظار برای گذشت مدت زمان لازم برای پخش نوت قبل از ورود به نوت بعدی.
  delay(noteDuration);
  // متوقف کردن تولید شکل موج جهت شروع نوت بعدی
  noTone(buzzer);
 }
}
```

درس پنجم: کنترل RGB LEDهای روی کریر بورد

پیش نیاز: نصب کتابخانه

کتابخانه‌های آردوینو مجموعه‌هایی از کدها است که کارهایی مثل ارتباط با سنسور، نمایشگر، ماژول و غیره را آسان می‌کنند و قابلیت‌های متعددی در استفاده از سخت افزار را در اختیار ما می‌گذارند.برای کار با RGB LED های روی کریر بورد از کتابخانه FastLED می‌توان استفاده نمود.

برای نصب این کتابخانه مشهور می‌توان در بخش Sketch —>Include Library —> Mange Libraries نام کتابخانه FastLED را جستجو و نصب نمود.



و سپس با دستور #include می‌توان از کتابخانه نصب شده استفاده نمود.

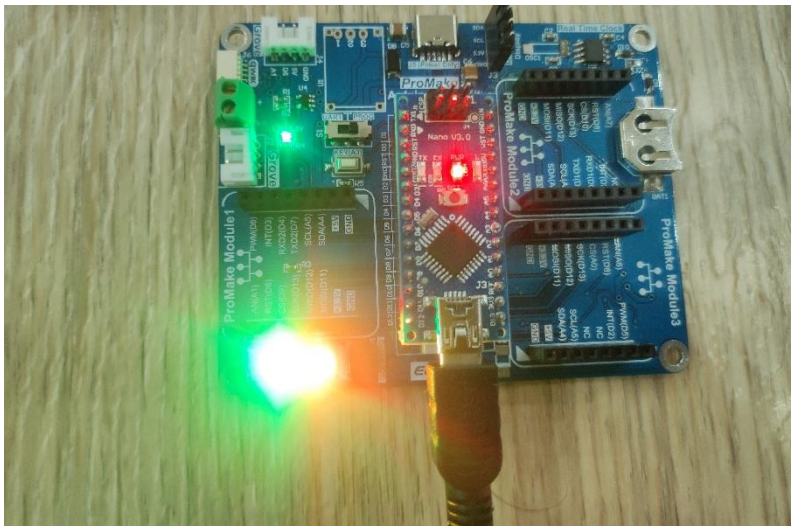
```
#include <FastLED.h>
```

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake(برای استفاده از RGB LED های روی آن)
- آردوینو نانو + کابل USB

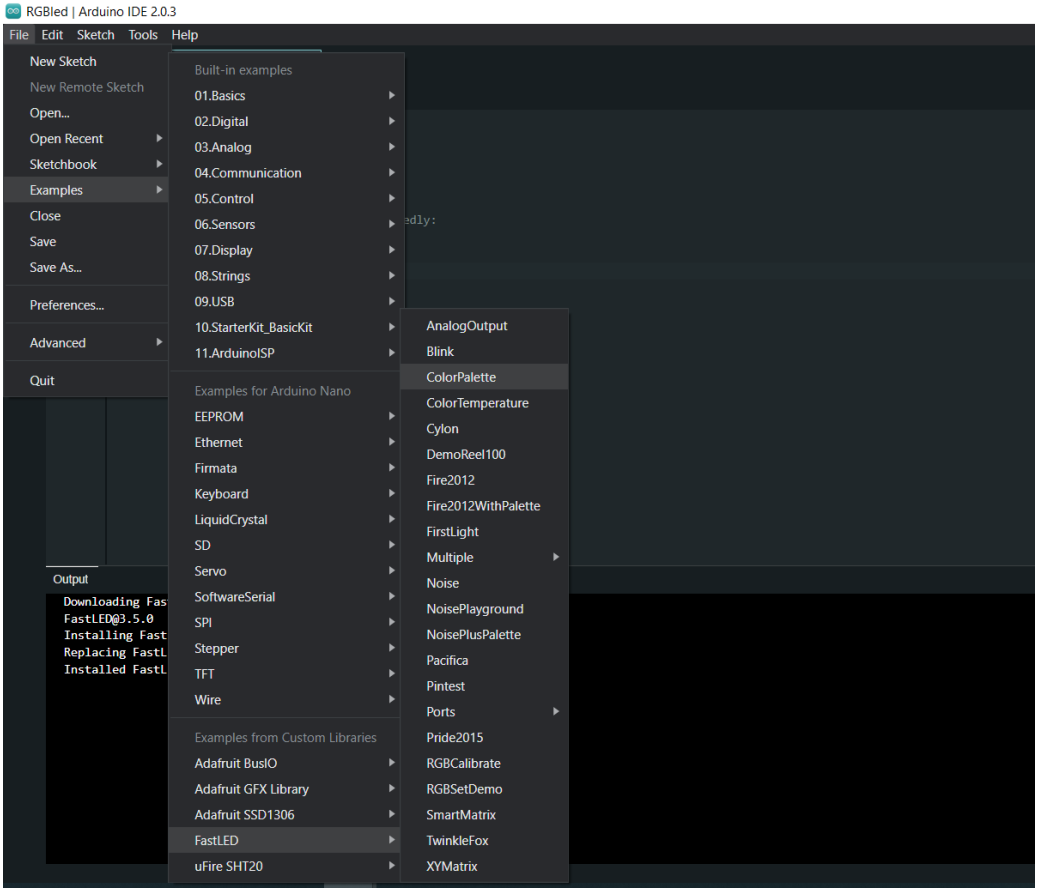
آماده‌سازی سخت افزار

ابتدا ماژول آردوینو نانو را با توجه به جهت صحیح بر روی کریر بورد قرار می‌دهیم و توسط کابل USB به کامپیوتر متصل می‌کنیم.



کدنویسی و شرح کد

در کتابخانه FastLED، مثال‌های جالبی برای آشنایی با عملکرد این کتابخانه در کار با RGB LED ها وجود دارد. که از منوی File ←Examples← FastLED قابل دسترس هستند.



برای نمونه ColorPalette که تنها نیاز به تغییر پایه متصل به led به ۱۶ که با نام LED_PIN ابتدای برنامه مشخص است، و تعداد LED ها به ۲ که به نام NUM_LEDS مشخص است، دارد و به راحتی بر روی برد اجرا می‌شود.

```
#define LED_PIN   16
#define NUM_LEDS  2
```

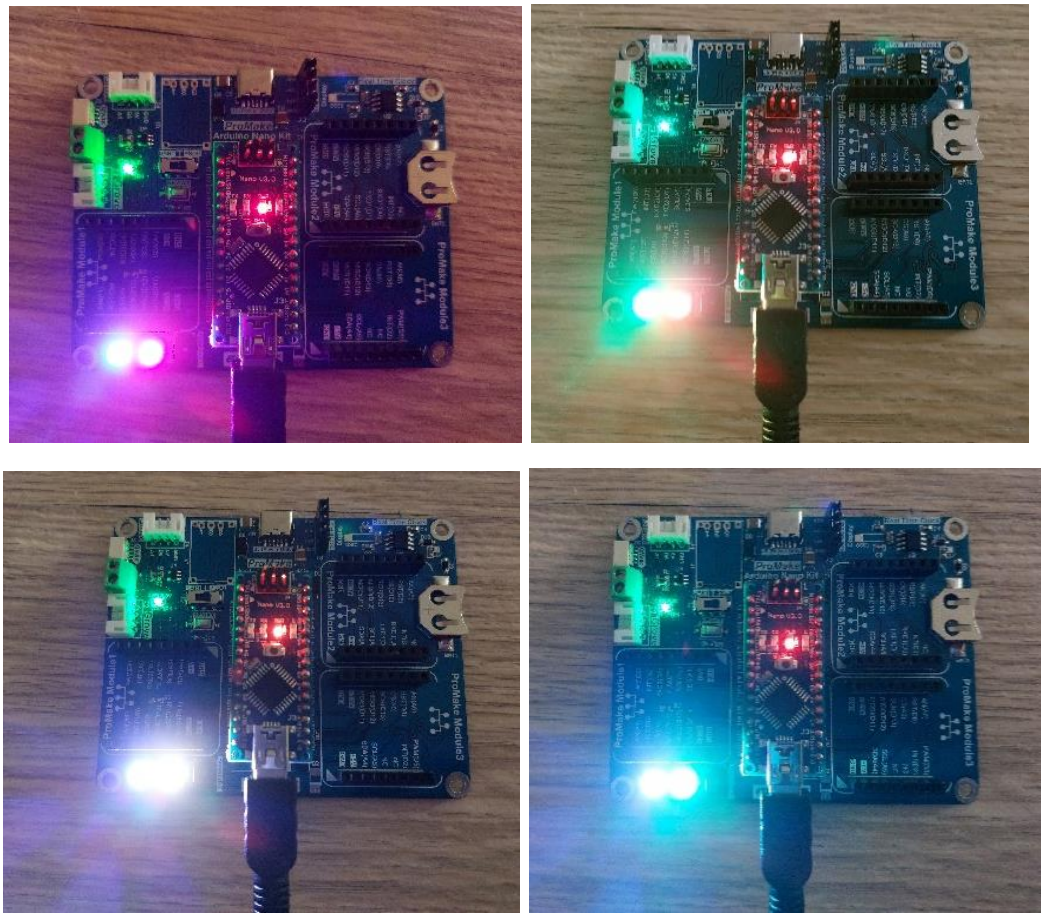
همچنین برنامه‌ای بسیار ساده با استفاده از این کتابخانه نیز می‌تواند به صورت زیر نوشته شود.

```
#include <FastLED.h>
#define LED_PIN 16
#define NUM_LEDS 2
CRGB leds[NUM_LEDS];
void setup() {
 FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
}
void loop() {
 leds[0] = CRGB(255, 0, 0);
 FastLED.show();
 delay(500);
 leds[1] = CRGB(0, 255, 0);
 FastLED.show();
 delay(500);
 leds[0] = CRGB(0, 0, 255);
 FastLED.show();
 delay(500);
 leds[1] = CRGB(150, 0, 255);
 FastLED.show();
 delay(500);
 leds[0] = CRGB(255, 200, 20);
 FastLED.show();
 delay(500);
 leds[1] = CRGB(85, 60, 180);
 FastLED.show();
 delay(500);
 leds[0] = CRGB(50, 255, 20);
 FastLED.show();
 delay(500);
 leds[1] = CRGB(10, 30, 220);
```

```
FastLED.show();
delay(500);
}
```

پس از اضافه کردن کتابخانه و تعریف پین متصل به RGBLED ها و تعیین کردن تعداد LED ها، آرایه‌ای از نوع CRGB نیز تعریف می‌کنیم. که این نوع آرایه شامل سه عضو داده یک بایتی برای هر یک از کانال‌های رنگ قرمز، سبز و آبی است.

در تابع setup باید FastLED را با پارامتر های تعریف شده در بالا مقدار دهی کنیم و سپس در حلقه ی اصلی با استفاده از تغییر اعداد در تابع CRGB که رنگ‌ها را به وجود می‌آورند، LED ها را با روش دلخواه کنترل کنیم. همچنین لازم به ذکر است برای اعمال تغییر در LEDها باید تابع FastLED.show(); را فراخوانی نماییم.

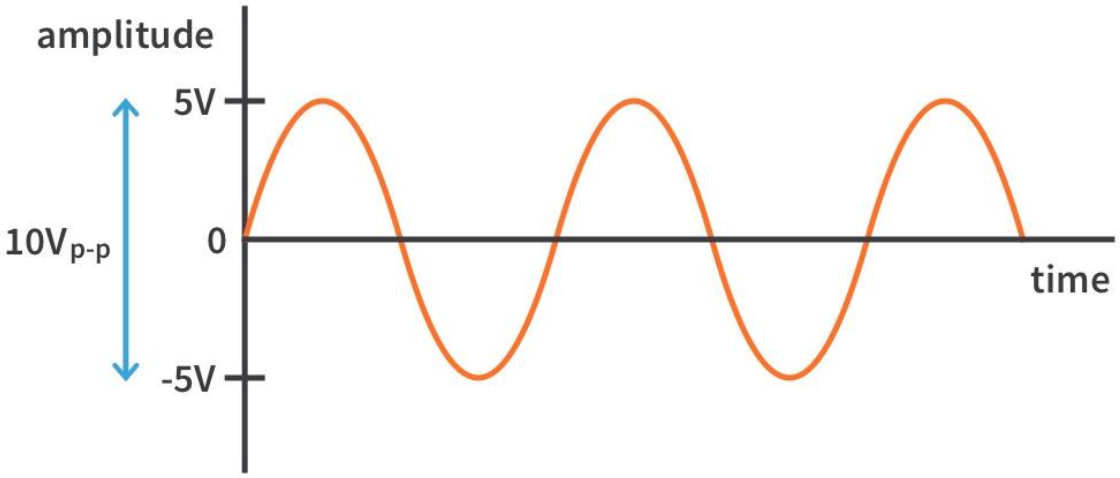


بخش چهارم: دریافت سیگنال از ورودی آنالوگ

درس ششم: ارتباط با سنسور MQ-2 جهت شناسایی گازهای خطرناک

پیش نیاز۱: سیگنال آنالوگ

سیگنال آنالوگ سیگنالی متغییر با زمان است که ویژگی آن این است که در هر بازه زمانی محدود و پیوسته، بی‌شمار نقطه وجود دارد که هرکدام مقدار متفاوت و مشخصی دارند. در یک سیگنال آنالوگ، مقدار ولتاژ، جریان یا فرکانس سیگنال می‌تواند بیانگر اطلاعات مورد نظر باشد. سیگنال‌های آنالوگ معمولا برای اندازه گیری میزان تغییرات در نور، صدا، دما، موقعیت، فشار یا سایر پدیده‌های فیزیکی استفاده می‌شوند و خروجی اکثر سنسورها، آنالوگ می‌باشد.



در این درس قصد داریم خروجی آنالوگ سنسور گاز MQ2 را اندازگیری کرده و رفتار آن را در اثر تحریک با گاز مشاهده کنیم.

پیش نیاز۲: سریال مانیتور

در نرم افزار آردوینو IDE ، پنجره‌ای به نام **سریال مانیتور** وجود دارد که برای مشاهده‌ی خروجی‌های متنی برنامه بسیار مفید است و می‌توان اطلاعاتی مانند مقادیر اندازگیری شده توسط سنسورها را به صورت متن از طریق پورت سریال آردوینو در رایانه شخصی مشاهده کرد. همچنین سریال مانیتور برای اشکال زدایی(Debugging) برنامه نیز می‌تواند سودمند باشد.

سریال مانیتور در محیط برنامه نویسی آردوینو از منو Tools --> Serial Monitor و همچنین به صورت مستقیم در برنامه(مطابق شکل فوق) قابل دسترس است.

دقت کنید که چون پورت سریال ماژول آردوینو هم برای آپلود برنامه و هم برای ارتباط سریال مانیتور مورد استفاده قرار می‌گیرد. امکان انجام همزمان این دو فرایند توسط یک یا چند برنامه آردوینو وجود ندارد و منجر به ایجاد خطا می‌شود.

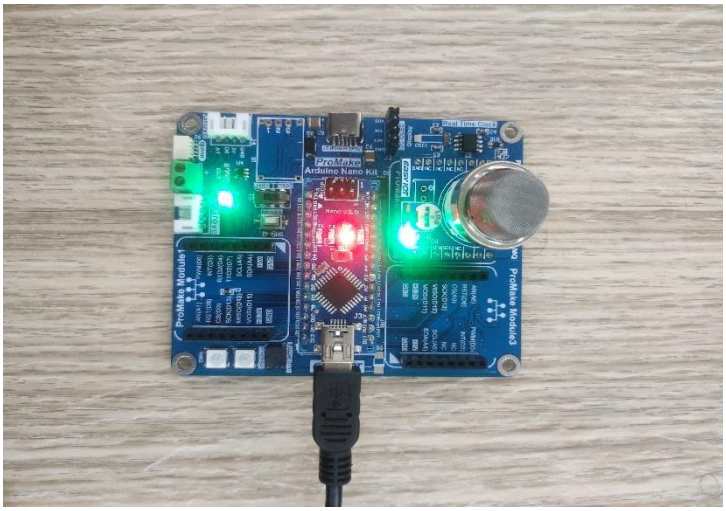
اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- ماژول آردوینو نانو + کابل USB
- ماژول گاز MQ-2 ProMake

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake گاز MQ-2 در محل ProMake Module 2 و ماژول آردوینو نانو در محل مربوطه بر روی کریر بورد قرار دهید و آردوینو نانو را توسط کابل USB به کامپیوتر متصل کنید.

نکته مهم در جایگذاری هر یک از ماژول‌ها، توجه به قرارگرفتن آن‌ها به صورت صحیح است. بدین منظور دقت شود که پایه های GND ماژول‌ها متناظر با پایه‌های GNDجایگاه آن‌ها بر روی کریر بورد، قرار بگیرد. در صورت صحیح قرارگرفتن ماژول ها led مربوط به تغذیه (PWR) آن‌ها روشن می‌شود.



کدنویسی و شرح کد

حال در محیط Arduino IDE کد با ساختار زیر را نوشته و اجرا می کنیم

```
int val = 0; // متغیر برای ذخیره‌سازی مقداری که خوانده می‌شود
int gasPin = A7; // متغیری که شماره پایه متصل به حسگر گاز را ذخیره می کند

void setup() {
 Serial.begin(9600);   // تنظیم ارتباط سریال
 pinMode(gasPin, INPUT); // تنظیم پایه متصل به حسگر گاز به عنوان ورودی
}

void loop() {
 val = analogRead(gasPin); // خواندن پایه حسگر گاز
 Serial.print("GAS: ");
 Serial.println(val); // چاپ مقدار خوانده شده
 delay(500);
}
```

تابع خواندن مقادیر آنالوگ پایه‌ها

analogRead() در آردوینو مقدار آنالوگ را از پین مشخص‌شده می‌خواند برد آردوینو nano حاوی یک مبدل آنالوگ به دیجیتال ۸ کاناله است. لذا که فقط پین‌های آنالوگ آردوینو می‌توانند به عنوان ورودی

ولتاژ آنالوگ استفاده شوند. مبدل آنالوگ به دیجیتال برد آردوینو دارای دقت ۱۰ بیتی (یعنی مقادیر عدد صحیح بین $[0, 2^{10} - 1]$) است. به این معنی که ولتاژهای ورودی بین ۰ تا ۵ ولت را به مقادیر عدد صحیح بین ۰ تا ۱۰۲۳ نگاشت می‌کند. بنابراین هر واحد $\frac{5}{1024} = 4.9\text{mV}$ است.

توابع ارتباط سریال

برای تعریف میزان سرعت و فعال کردن پورت سریال جهت ارسال داده برای سریال مانیتور از تابع Serial.begin(baudrate) استفاده می‌کنیم. baudrate بیانگر سرعت انتقال داده (بیت بر ثانیه) در ارتباط سریال است که باید در فرستنده و گیرنده به صورت مشابه تنظیم شود.

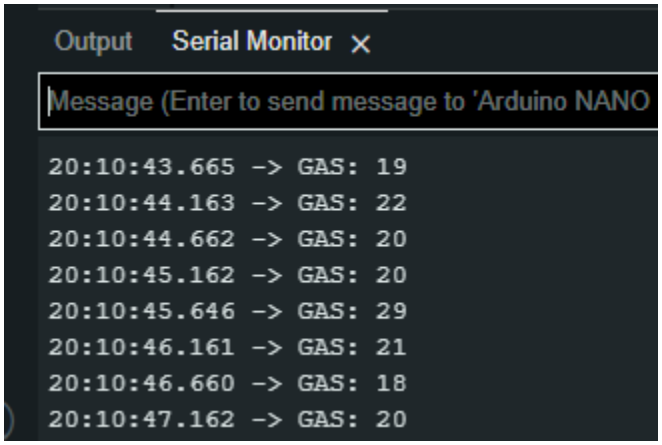
```
Serial.begin(9600);
//Serial.begin(baudrate);
```

تابع Serial.print(data, format) داده ها را به صورت رشته‌ای(String) به پورت سریال ارسال می‌کند. data داده‌ای است که می‌خواهیم آن را ارسال کنیم و format قالب نمایش و تعداد ممیز اعشار نمایش داده را مشخص می‌کند. که الزام وجود آن در صورتی است که بخواهیم در مبنای غیر از (DEC)۱۰ یعنی (BIN)۲ یا (OCT)۸ یا (HEX) 16 یا با تعداد اعشار دلخواه نمایش دهیم. برای مثال خروجی آردوینو برای عبارت Serial.print(25, BIN); مقدار 11001 می‌باشد.

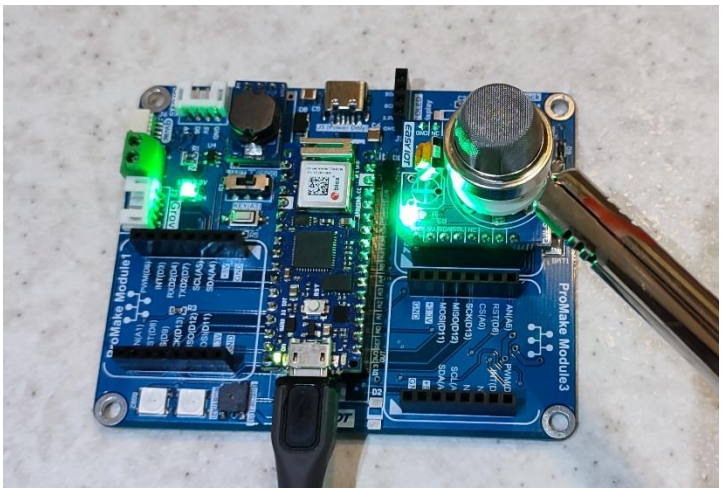
تابع Serial.println همانند تابع print عمل می‌کند با این تفاوت که با هر بار ارسال اطلاعات، دستور رفتن به خط بعد را نیز اجرا می‌کند. زمانی که می‌خواهیم عبارتی عیناً چاپ شود آن را درون " " قرار می‌دهیم.

```
Serial.print("Sensor Value: ");
Serial.print(sensorValue);
// Serial.print (data, format type (optional))
// Serial.println (data, format type (optional));
```

پس از اجرای برنامه فوق برروی برد نتایج در سریال مانیتور به شکل زیر است.



حال برای تست عملکرد سنسور گاز کافی است که از یک فندک استفاده کنیم. ابتدا باید فندک را روشن کرده و شعله آن را با فوت خاموش کنیم تا فقط گاز از آن متصاعد شود. حال با نزدیک کردن سر فندک خاموش به سنسور گاز شاهد تغییر در اعداد خوانده شده از حسگر خواهیم بود.



درس هفتم: تولید بوق هشدار هنگام بالا رفتن سطح گازهای خطرناک

پیش نیاز۱: آشنایی با سنسور گاز MQ

این سنسورها نسبت به طیف گسترده‌ای از گازها حساس اند و در خانه و دمای اتاق استفاده می‌شوند.

سنسور MQ-2 حساس به متان، بوتان، LPG و دود است. این سنسور نسبت به گازهای قابل اشتعال و گازهای تولید شده از فرایند احتراق حساس می‌باشد .کاربردهای متعددی برای این سنسور گاز وجود دارد به ویژه اینکه می‌توان با استفاده از آن جان انسان را از خطرات احتمالی نجات داد .از این رو حسگرهای گاز نقش مهمی در بخش‌های مختلف ایفا می‌کنند که شامل صنعت، پزشکی، کاربردهای زیست محیطی و کاربردهای خانگی برای نظارت بر گازهای سمی و قابل اشتعال می‌شود.

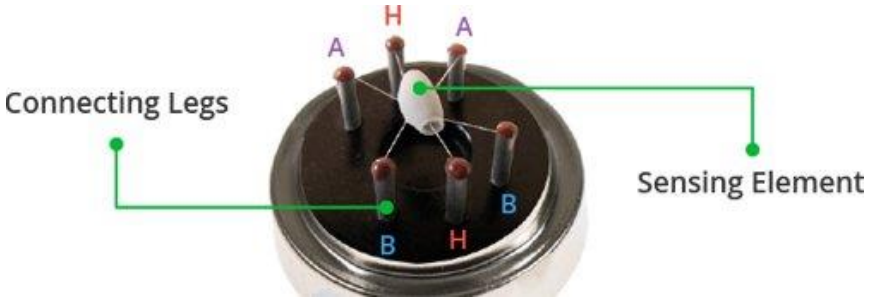
قابلیت‌های سنسورهای گاز MQ

- رنج تشخیص: ۱۰تا ۱۰۰۰ ppm
- ولتاژ کاری: ۵ ولت
- خروجی آنالوگ: ۰ تا ۵ ولت
- حساسیت بالا برای تشخیص گاز
- پروب‌های قابل استفاده برروی ماژول ProMake
  - MQ-2: گازهای قابل اشتعال ۳۰۰ تا ۱۰۰۰۰ ppm
  - MQ-3: گاز الکل ۲۵ تا ۵۰۰ ppm
  - MQ-4: گاز، متان ۳۰۰ تا ۱۰۰۰۰ ppm
  - MQ-5: گاز، متان، گاز زغال سنگ ۳۰۰ تا ۵۰۰۰ ppm
  - MQ-6: گاز LPG، ایزوبوتان، پروپان، ۱۰۰ تا ۱۰۰۰۰ ppm
  - MQ-7: مونوکسید کربن CO ۱۰ تا ۱۰۰۰ ppm

ساختار سنسورهای MQ

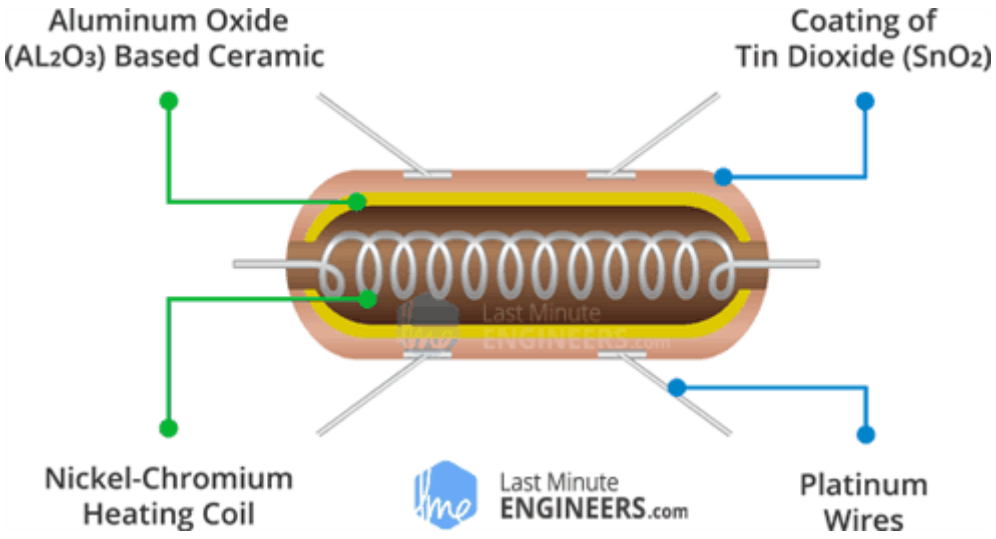
سنسورهای گاز سری MQ از هیتر داخلی کوچک استفاده می‌کنند لذا با دولایه توری از فولاد ضد زنگ که "شبکه ضد انفجار" نامید می شود، پوشانده شده است. بدین ترتیب می‌توان اطمینان حاصل نمود که حرارت داخلی پروب منجر به بروز انفجار گازهای قابل اشتعالی که قصد شناسایی آنها را داریم نمی شود.

این شبکه توری همچنین باعث در امان ماندن حسگر داخلی از ذرات معلق موجود در محیط می‌شود و وفقط امکان نفوذ عناصر گازی را به محفظه داخلی حسگر فراهم می کند.



اگر این لایه توری را برداریم حسگر داخلی را مشابه تصویر فوق خواهیم یافت. حساسه این حسگر داری ٦ پایه می باشد. دو عدد از این پایه ها که با حرف H نشان داده شده اند به سیم پیچی از جنس آلیاژ نیکل-کروم متصل هستند که وظیفه تولید حرارت را بر عهده دارند.

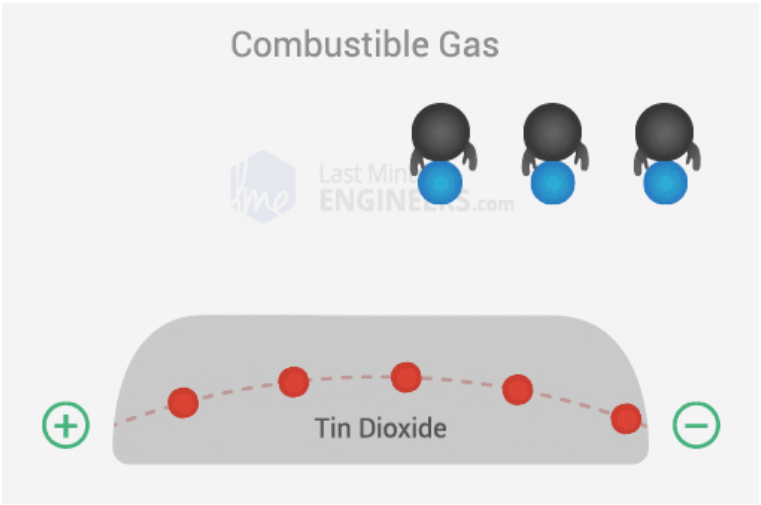
حساسه کپسول مانند سرامیکی دارای یک پوشش دی اکسید قلع(SnO2) می باشد که به گازهای قابل اشتعال حساس است. ساختار کپسولی از طرف دیگر منجر به بهبود تولید گرما می شود.



نحوه عملکرد سنسور MQ-2

همان طور که گفتیم حسگر MQ2 دارای حساسه‌ای است که از تکنولوژی نیمه هادی اکسید فلزی(MOS) بهره می‌گیرند. هنگامی که لایه دی اکسید قلع تا دمای بالایی گرم شود اکسیژن برروی سطح آن جذب می‌شود. در صورتی که هوای اطراف حسگر پاک باشد، الکترون‌های آزادی که موجب رسانایی دی اکسید قلع می‌شوند، جذب ملکول‌های اکسیژن می‌شوند. بدین ترتیب لایه دی اکسید قلع دچار کمبود الکترون شده و مقاومت الکتریکی بالایی در برابر عبور جریان پیدا می‌کند.

در صورت وجود گازهای کاهنده(متضاد اکساینده) چگالی اکسیژن جذب شده برروی سطح دی اکسید قلع در اثر واکنش با این گازها کم می‌شود. بدین ترتیب الکترون‌ها در لایه دی اکسید قلع آزاد شده و به جریان اجازه عبور آزادانه می‌دهند.



لذا با افزایش غلظت گازهای قابل احتراق ولتاژ خروجی این حسگر بالا رفته و با کاهش غلظت این گازها این ولتاژ پایین می‌آید.

نحوه کالیبره کردن سنسور MQ-2

در صورتی که این حسگر برای مدت طولانی در انبار یا بلا استفاده مانده باشد ممکن است کالیبراسیون آن به هم بخورد. برای اولین استفاده پیشنهاد می‌شود حسگر به مدت ۲۴ تا ۴۸ ساعت روشن و گرم بماند تا به حداکثر دقت برسد.

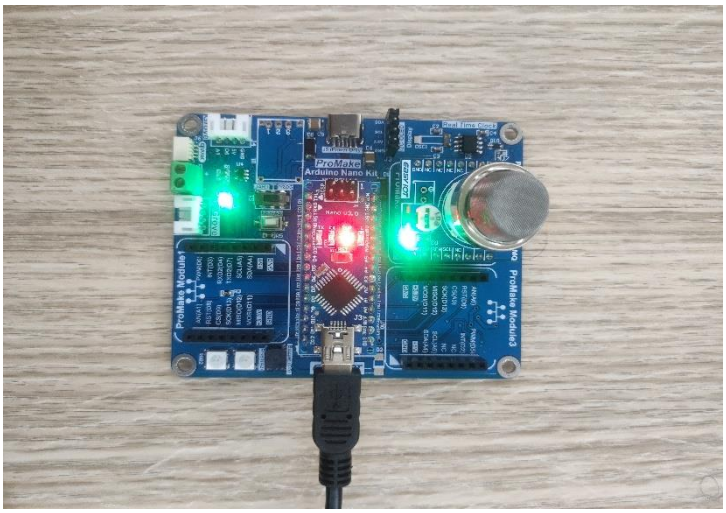
اگر حسگر به تازگی مورد استفاده قرار گرفته باشد زمان لازم برای گرم شدن کامل حدود ۵ تا ۱۰ دقیقه خواهد بود. در بازه زمانی گرم شده ابتدا مقادیر ولتاژ خوانده شده از حسگر اعداد بالایی خواهد بود و به مرور این اعداد کوچک می‌شوند تا به پایداری برسند.

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- ماژول آردوینو نانو + کابل USB
- ماژول گاز MQ-2 ProMake

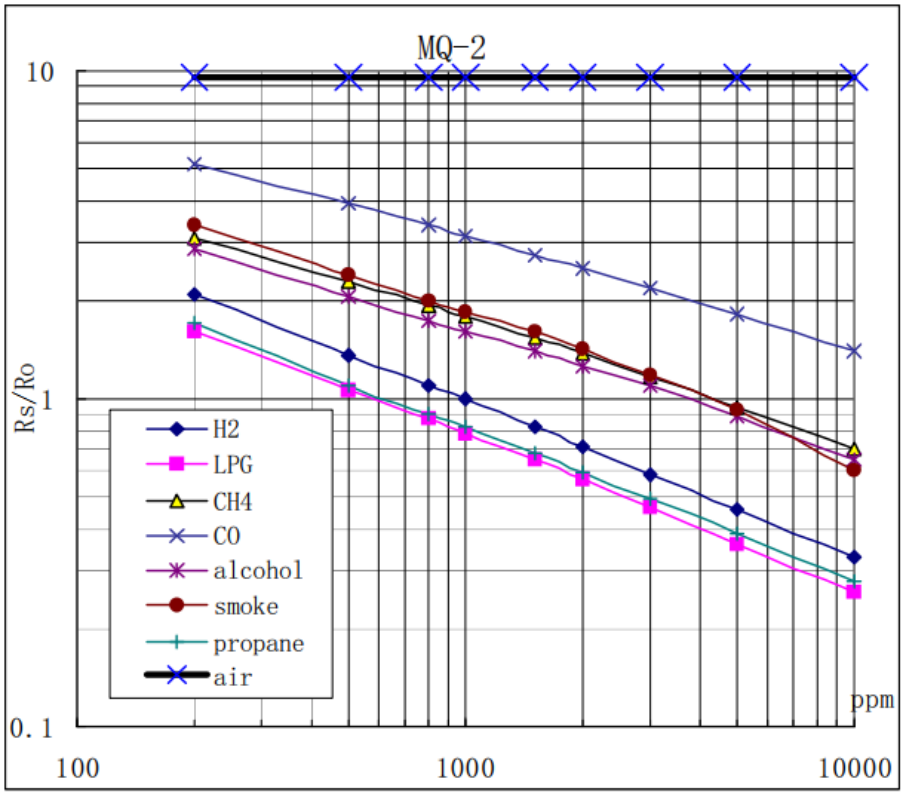
آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake گاز MQ-2 در محل ProMake Module 2 و ماژول آردوینو نانو در محل مربوطه بر روی کریر بورد قرار دهید و آردوینو نانو را توسط کابل USB به کامپیوتر متصل کنید.



کدنویسی و شرح کد

همان طور که توضیح داده شد سنسور گاز MQ-2 به حضور چندین گاز در محیط حساسیت نشان می‌دهد. همچنین دقت اندازگیری آن با تغییر دما و رطوبت نیز تغییر می‌کند. لذا هنگام استفاده از آن برای اندازه گیری هم می‌بایست هم کالیبراسیون به دقت انجام پذیرد و هم اثر دما و رطوبت لحاظ شود. همچنین جهت جلوگیری از تداخل اثر گازهای دیگری که بر حساسه اثر می گذارند باید در محیطی استفاده شود که فقط امکان حضور گاز هدف در آن وجود داشته باشد.



در نمودار فوق تغییرات نسبت مقاومت حسگر در حضور غلظت‌های مختلف گازهای موثر بر حساسه، بر مقاومت حسگر در حضور 1000ppm گاز هیدروژن در هوای پاک، نمایش داده شده است.

برای ساده سازی محاسبات مورد نیاز جهت یافتن غلظت گاز موجود در محیط براساس خروجی حسگر MQ-2 از کتابخانه MQ2 استفاده خواهیم کرد.

حال در محیط Arduino IDE کد با ساختار زیر را نوشته و اجرا می‌کنیم. در این کد در صورت رسید غلظت گاز مونواکسیدکربن (CO) به آستانه دلخواه، هشدار صوتی جهت جلوگیری از مسمویت و مرگ احتمالی در اثر گاز گرفتگی، تولید می‌شود.

```
int BuzzerPin = 5; // متغیری که پایه متصل به بازر را ذخیره می کند
int MQpin = A7;   // متغیری که شماره پایه متصل به حسگر گاز را ذخیره می کند

#include <ProMake_MQ2.h>
ProMake_MQ2 mq2(MQpin);

void setup() {
 Serial.begin(9600);
 mq2.begin(); // انجام فرایند کالیبراسیون
}
void loop() {
 float co = mq2.readCO(); // اندازگیری مقدار مونواکسید کربن

 if (co > 10) {  // بررسی در برابر آستانه هشدار مد نظر
  for (int hz = 440; hz < 1000; hz += 25) {
   tone(BuzzerPin, hz, 50);
   delay(30);
  }
  for (int hz = 1000; hz > 440; hz -= 25) {
   tone(BuzzerPin, hz, 50);
   delay(30);
  }
  delay(100);
 } else {
  delay(1000);
 }
}
```

در ساختار شرطی if امکان تغییر مقدار آستانه براساس نیاز وجود دارد.

حلقه ها

یک عبارت حلقه‌ای به ما اجازه می‌دهد تا یک یا چند عبارت را چندین بار اجرا کنیم. در ادامه متداول ترین ساختارهای حلقه‌ای را معرفی می‌کنیم.

ساختار حلقه for

حلقه for برای تکرار اجرا کردن بخشی از کد که نیاز داریم به تعداد بار مشخص اجرا و تکرار شوند به کار می‌رود. دستوراتی که می‌خواهیم تکرار شوند، بین آکولاد ({}) قرار می‌گیرند. معمولاً یک شمارنده افزایشی، برای کنترل اجرای حلقه و درنهایت خاتمه‌ی آن مورد استفاده قرار می‌گیرد.

سر حلقه[1]ی For سه قسمت دارد: (از چپ به راست به ترتیب) مقداردهی اولیه، شرط حلقه، مقدار افزایشی حلقه

```
for (initialization; condition; increment) { // for(مقدار افزایشی حلقه; شرط حلقه; مقداردهی اولیه)
// عملیات ها;
}
```

مقداردهی فقط یک بار و در شروع حلقه رخ می‌دهد. در هر بار تکرار حلقه، شرط چک می‌شود و اگر نتیجه True بود، کد درون حلقه اجرا می‌شود و سپس عمل افزایش متغیر حلقه انجام می‌شود. دوباره شرط چک می‌شود و عملیات توضیح داده شده در خط قبل، تا زمان True بودن شرط تکرار خواهد شد. هنگامی‌که نتیجه‌ی شرط False شود، حلقه پایان می‌پذیرد. برای مثال

```
for (int x; x<10; x++) {
        int y;
        y=x+y;
}
```

ابتدا x برابر ۰ است و عملیات جمع با x=0 انجام می‌گیرد، سپس x با ۱ جمع شده و برابر ۱ می‌شود و عملیات انجام می‌گیرد و تا جایی پیش می‌رود که x برابر ۹ شود و عملیات انجام گیرد. حال زمانی که x با ۱ جمع شود برابر ۱۰ شده و دیگر شرط x<10 درست نیست و برنامه از حلقه خارج می‌شود.

ساختار حلقه while

حلقه while به‌طور مداوم و بدون انتها، دور می‌زند تا زمانی که عبارت منطقی درون پرانتز()، مقدار False بگیرد. کدهای داخل حلقه می‌بایست مورد عبارت منطقی بررسی را عوض کند و گرنه حلقه while هیچ‌گاه تمام نخواهد شد.

```
while(expression){
 // عملیات ها
}
```

برای مثال

```
int i;
while (i <= 5)
{
        int j;
        j=i*i;
        i++;
```

[1] Header

```
}
```

ابتدا i برابر صفر است و عبارت i<=5 صحیح است و برنامه وارد حلقه می‌شود و عملیات به توان رساندن را انجام می‌دهد و در نهایت i با ۱ جمع می‌شود و برابر با ۱ می‌شود حال باز شرط i<=5 صحیح است و حلقه مجددا اجرا می‌شود تا جایی که i برابر ۵ شود و با ۱ جمع شده و برابر ۶ شود حال دیگر شرط حلقه صحیح نیست و ادامه برنامه از آن خارج می‌شود.

بخش پنجم: ارتباط از طریق I2C

درس هشتم: ارتباط با حسگر دما و رطوبت روی ماژول Sensor Tag

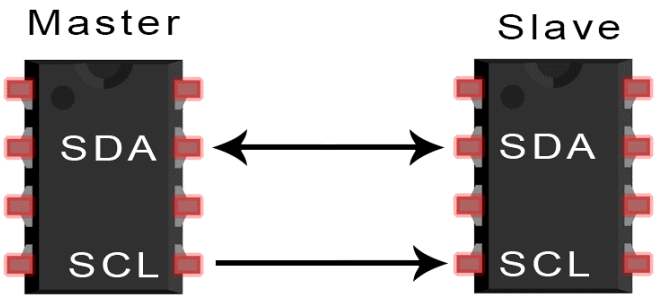
پیش نیاز: ارتباط I2C

ارتباط I2C یک پروتکل ارتباطی سریال یک طرفه از Master به Slave است و داده‌ها به ترتیب در امتداد خط SDA منتقل می‌شوند. ارتباط I2C همگام نیز هست، بدین معنی که خروج بیت‌ها از Master و نمونه‌برداری از بیت‌ها در Slave توسط یک سیگنال کلاک مشترک بین Master و Slave هماهنگ می‌شود و سیگنال کلاک همیشه توسط Master کنترل می‌شود.

در ارتباط I2C فقط از دو سیم برای انتقال داده استفاده می‌شود که به شرح زیر است:

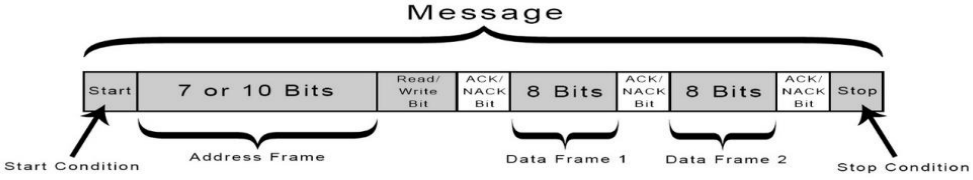
SDA (Serial Data) = خطی که برای ارسال و دریافت داده بین master و slave مورد استفاده قرار می‌گیرد.

SCL (Serial Clock) = خطی که حامل سیگنال کلاک می‌باشد.



نحوه عملکرد I2C

در I2C داده‌ها به صورت بسته‌هایی شامل چندین بخش فرستاده می‌شوند. هر پیام ارسالی شامل شرط شروع،بخش آدرس (که همان آدرس باینری مربوط به slave است)، بیت read/write، یک یا دو بخش مربوط به داده، بیت‌های ACK/NACK بین هربخش از داده‌های ارسالی و در نهایت شرط خاتمه است.



معرفی اجزای یک پیام I2C

به طور پیش‌فرض هر دو خط SDA و SCL در وضعیت 1 یا High قرار دارند و در اصطلاح فنی pullup هستند.

شرط شروع: خط SDA از حالت High به حالت Low تغییر می‌کند قبل از اینکه خط SCL از حالت High به حالت Low تغییر کند.

شرط پایان: خط SDA بعد از اینکه خط SCL از حالت Low به حالت High تغییر می‌کند از از حالت Low به حالت High تبدیل می‌شود.

بخش آدرس: توالی ۷ یا ۱۰ بیتی منحصر به فرد برای هر Slave که وقتی که Master می‌خواهد به آن Slave پیام بفرستند آن را با آدرس آن Slave پر می‌کند.

بیت read/write: یک بیت واحد که مشخص می‌کند Master قصد ارسال داده برای Slave را دارد (سطح ولتاژ پایین) یا می‌خواهد از آن داده دریافت کند (سطح ولتاژ بالا).

بیت ACK/NACK: به دنبال ارسال هر بخش از پیام یک بیت تصدیق کننده[7] / عدم تصدیق کننده[8] توسط Master شنود می‌شود. در صورتی که بخش آدرس پیام یا بخش داده‌ی پیام به درستی توسط Slave دریافت شود، یک بیت ACK از طرف Slave با Low کردن خط SDA ارسال می‌گردد. در صورت عدم حضور Slave با آدرس خواسته شده و عدم ارسال بیت ACK، وضعیت موجود خط SDA که High است دال بر بیت NACK خواهد بود.

مراحل انتقال داده از طریق پروتکل I2C

۱- شرایط شروع از طرف Master به تمامی Slave هایی که به I2C Bus متصل هستند، ارسال می‌گردد. به این صورت که ابتدا خط SDA و سپس خط SCL از حالت High به Low تغییر می‌کنند.

۲- سپس آدرس ۷ یا ۱۰ بیتی و بیت read/write توسط Master برروی Bus فرستاده می‌شود و Master منتظر دریافت تصدیق می‌ماند.

۳- هر کدام از Slave آدرس فرستاده شده را با آدرس خود مقایسه می‌کنند. در صورت همخوانی آدرس، Slave مورد نظر بیت ACK را از طریق قرار دادن خط SDA در حالت Low، ارسال می‌کند. و در صورتی که آدرس همخوانی نداشته باشد، خط SDA در حالت High ثابت باقی خواهد ماند.

[7] Ack
[8] Nack

٤-Master در صورت دریافت ACK بسته داده را ارسال و یا دریافت می‌کند.

٥- بعد از ارسال هر کدام از بسته‌های داده، دستگاه دریافت کننده یک بیت ACK به فرستنده ارسال می‌کند تا از دریافت موفق داده اطمینان حاصل کند.

٦- برای توقف انتقال داده‌ها، Master شرایط توقف را به Slave ارسال می‌کند، به این صورت که ابتدا خط SCL و سپس خط SDA را از حالت Low به حالت High تغییر می‌دهد.

برای استفاده از ارتباط I2C در آردوینو کتابخانه Wire مورد استفاده قرار می‌گیرد.

```
#include <Wire.h>
```

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- ماژول ProMake Sensor TAG
- آردوینو نانو + کابل USB

آماده‌سازی سخت افزار

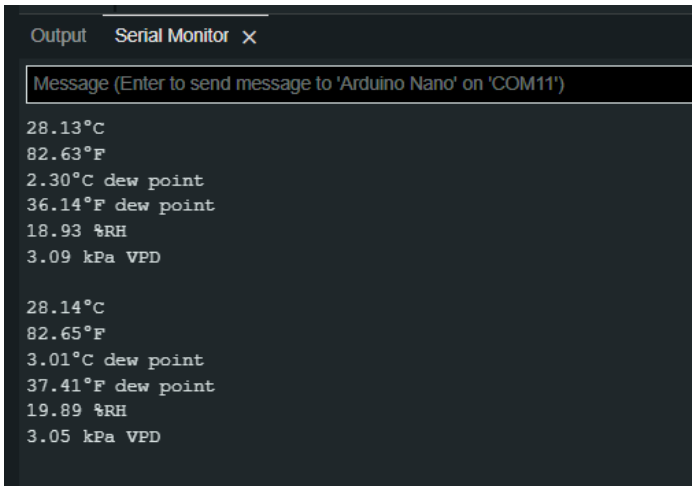
با توجه به نشانگرهای جهت ماژول ProMake Sensor TAG در جای ProMake Module 2 و ماژول آردوینو نانو بر روی کریر بورد قرار دهید و آردوینو را توسط کابل USB به کامپیوتر متصل کنید.

کدنویسی و شرح کد

برای استفاده راحت تر از حسگر SHT20 از کتابخانه توسعه داده شده توسط شرکت سازنده کیت استفاده خواهیم کرد. حال در محیط Arduino IDE کد زیر را نوشته و اجرا نمایید.

```
#include "ProMake_SHT20.h"
ProMake_SHT20 sht20; //تعریف متغیر استفاده از سنسور
void setup() {
 Serial.begin(9600); //تنظیم ارتباط سریال برای نمایش
 sht20.checkSHT20(); //شروع به کار سنسور دما و رطوبت
}
void loop() {
 Serial.println((String)sht20.readTemperatureC() + "°C");   // چاپ دما برحسب سانتی گراد
 Serial.println((String)sht20.readTemperatureF() + "°F");   //چاپ دما برحسب فارنهایت
 Serial.println((String)sht20.dew_pointC() + "°C dew point"); // دمایی که هوا با بخار آب اشباع می شود
 Serial.println((String)sht20.dew_pointF() + "°F dew point");
 Serial.println((String)sht20.readHumidity() + " %RH"); // مقدار رطوبت
 Serial.println((String)sht20.vpd() + " kPa VPD");    //معیاری برای اختلاف رطوبت با میزان رطوبت اشباع در هوا
 Serial.println();
 delay(5000);
}
```

نتایج در سریال مانیتور به شکل زیر است.



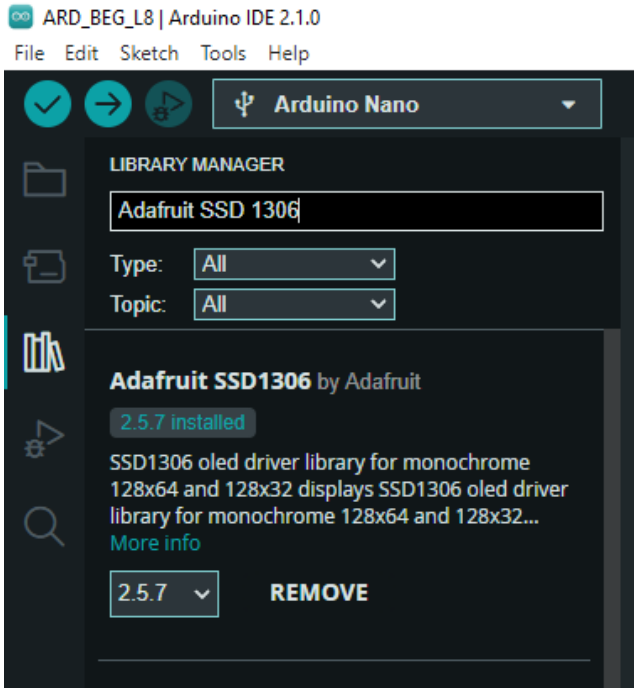
درس نهم: نمایش اطلاعات بر روی نمایشگرOLED

پیش نیاز: نمایشگرهای OLED

نمایشگرهای OLED، نمایشگرهایی با کنتراست و رزولوشن بالا می‌باشند، از این رو قابلیت خوانایی زیادی را برای کاربر فراهم می‌کنند. این نمایشگرها نیاز به نور پس زمینه (Backlight) ندارند و پیکسل‌ها خودشان نور افشانی می‌کنند. چیپ درایور این ماژول SSD1306 است که توانایی ارتباط I2C را برای این ماژول فراهم می‌آورد. نمایشگر۰٫۹۱ اینچی OLED دارای ٤ پایه به شرح زیر است:

- VCC: تغذیه نمایشگر – ٥ ولت
- GND: زمین
- SCL: کلاک پروتکل I2C
- SDA: خط داده پروتکل I2C

برای استفاده از آن دو کتابخانه Adafruit GFX و Adafruit SSD1306 مورد نیاز می‌باشد. که می‌توانید از بخش کتابخانه‌های برنامه آردوینو آنها را نصب و استفاده نمایید.



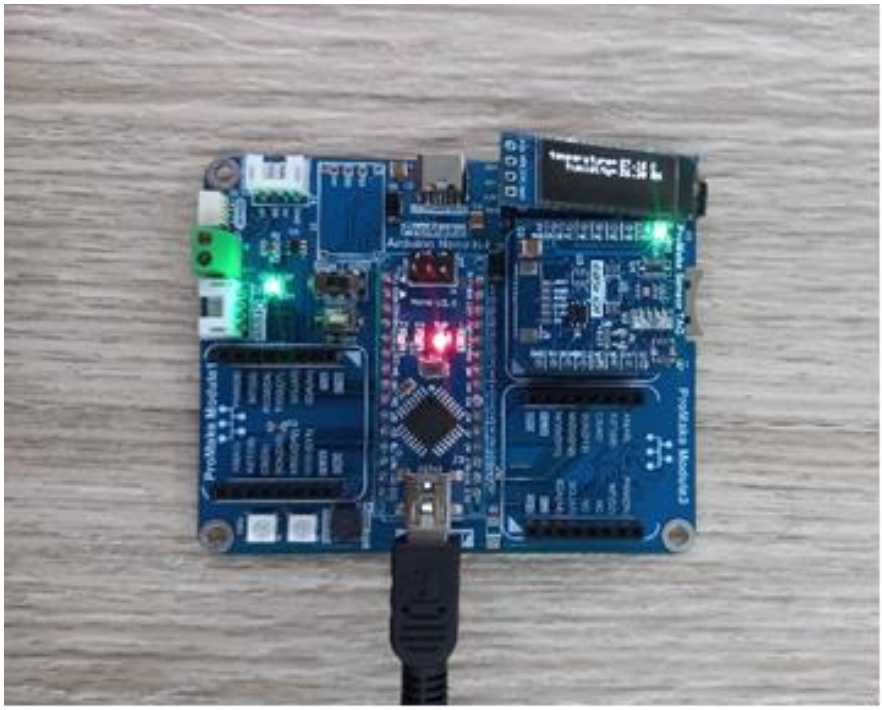

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- آردوییِنو نانو + کابل USB
- ماژول ProMake Sensor TAG
- نمایشگر OLED

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake Sensor TAG در جای ProMake Module 2 و ماژول آردوینو نانو را بر روی کریر بورد قرار دهید و آردوینو را توسط کابل USB به کامپیوتر متصل کنید. نمایشگر OLED را نیز با توجه به مارکاژ پایه‌ها در جای تعبیه شده روی کریر بورد قرار دهید.



کدنویسی و شرح کد

هدف برنامه نوشته شده نمایش دما و رطوبت اندازه‌گیری شده توسط ماژول ProMake Sensor TAG بر روی نمایشگر OLED می‌باشد. پس مطابق درس هشتم کتابخانه‌ی سنسور نیز اضافه و از آن استفاده شده است.

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
#define SCREEN_WIDTH 128 // تعداد پیکسل های عرض نمایشگر
#define SCREEN_HEIGHT 32 // تعداد پیکسل های ارتفاع نمایشگر
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#include "ProMake_SHT20.h"
ProMake_SHT20 sht20;

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

 if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  while (true)
   ;
 }

 Wire.begin();
 sht20.checkSHT20();
 display.clearDisplay();    //پاک کردن صفحه نمایشگر
 display.setTextColor(WHITE); //تنظیم رنگ نوشته روی نمایشگر
 display.setTextSize(1);    //تنظیم سایز نوشته روی نمایشگر
}
void loop() {
 float temp, hum;
 temp = sht20.readTemperatureC();
 hum = sht20.readHumidity();
 display.clearDisplay();
 display.setCursor(0, 1);     //تنظیم مکان چاپ روی نمایشگر
 display.print("temperature= "); // "" چاپ عبارت متنی داخل
 display.setCursor(75, 1);    // انتقال مکان چاپ به جلو عبارت
 display.print(temp);       //چاپ مقدار داخل متغیر
 display.setCursor(110, 1);
 display.print("C");
 display.setCursor(0, 10);
 display.print("humidity= ");
 display.setCursor(75, 10);
 display.print(hum);
 display.setCursor(110, 10);
 display.print("RH");
```

```
 display.display(); //نمایش بر روی نمایشگر
 delay(1000);
}
```

توابع کار با نمایشگر

تعریف طول و عرض نمایشگر برای OLED 128x32 و تعریف نمایشگر متصل به I2C

```
#define SCREEN_WIDTH 128 //تعداد پیکسل های عرض نمایشگر
#define SCREEN_HEIGHT 32 //تعداد پیکسل های ارتفاع نمایشگر
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

پاک کردن نمایشگر و به عبارتی خاموش کردن تمامی پیکسل‌ها

```
display.clearDisplay();
```

تنظیم رنگ نوشته روی نمایشگر

```
display.setTextColor(WHITE);
```

تنظیم سایز نوشته روی نمایشگر (از سایز ۱ تا ۸ می توان قرار داد.)

```
display.setTextSize(1);
```

تنظیم موقعیت نشانگر و شروع نمایش

```
display.setCursor( x, y );
```

نمایش متن

```
display.print("نوشته");
```

نمایش اعداد درون متغیر

```
display.print(temp);
```

اعمال دستورات و تغییرات نوشته شده در خطوط قبل و نمایش برای نمایشگر

```
display.display(); //نمایش بر روی نمایشگر
```

درس دهم: اندازگیری نور محیطی و کنترل رله برای روشن کردن چراغ‌ها در شب

پیش نیاز: سنسور نور VEML7700

سنسور نور محیطی VEML7700 دارای دقت بالا (ALS) و با رابط I2C است. این سنسور از چندین فناوری اختصاصی برای اطمینان از اندازه گیری دقیق شدت نور با پاسخ طیفی بسیار نزدیک به چشم انسان استفاده می‌کند. این سنسور با استفاده از یک دیود نوری حساس، تقویت کننده کم نویز و یک مبدل A/D با دقت ۱۶ بیتی، می‌تواند داده‌ها را مستقیماً بدون نیاز به محاسبات پیچیده ارائه دهد. محدوده دینامیکی برای سنسور نور محیط بسیار وسیع است، از 0 لوکس تا حدود 167K لوکس شامل می‌شود. محدوده دینامیکی بالا همراه با پاسخ خطی به منابع مختلف نور، به این سنسور اجازه می‌دهد تا در پشت شیشه تیره یا پانل‌های ساخته شده از مواد نیمه شفاف دیگر قرار گیرد.

کتابخانه سنسور نور

برای استفاده از سنسور نور محیطی VEML7700 موجود در ماژول ProMake Sensor Tag از کتابخانه Adafruit_VEML7700 استفاده می‌شود که از اینجا[9] قابل دانلود است.

```
#include "Adafruit_VEML7700.h"
```

ابتدا عبارت زیر برای استفاده از سنسور مورد نظر نوشته می‌شود.

```
Adafruit_VEML7700 veml = Adafruit_VEML7700();
```

دو پارامتر بهره و زمان استفاده از این سنسور با استفاده از دو دستور زیر قابل تنظیم است.در برنامه نوشته شده در switch case ها گزینه های مقادیر این دو پارامتر آورده شده است. در صورت تنظیم نکردن مقادیر پیش فرض سنسور در نظر گرفته می شود.

```
veml.setGain(VEML7700_GAIN_1_8);
veml.setIntegrationTime(VEML7700_IT_100MS);
```

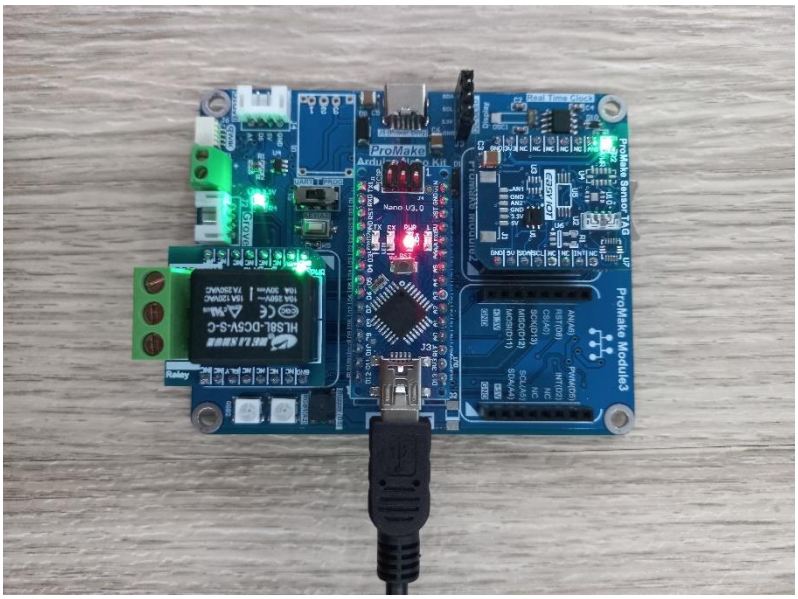
اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- آردوینو نانو + کابل USB
- ماژول ProMake Sensor TAG
- ماژول ProMake رله تک کانال

[9] https://github.com/adafruit/Adafruit_VEML7700/archive/refs/heads/master.zip

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake Sensor TAG و ماژول ProMake رله تک کانال را به ترتیب در جای ProMake Module 2 و ProMake Module 1 و ماژول آردوینو نانو بر روی کریر بورد قرار دهید و آردوینو توسط کابل USB به کامپیوتر متصل می‌شود.



کدنویسی و شرح کد

کد نوشته شده برای راه اندازی و خواندن مقادیر سنسور نور و کنترل رله به شکل زیر است.

```
#include "Adafruit_VEML7700.h"
Adafruit_VEML7700 veml = Adafruit_VEML7700();
int RELAY_PIN = 9;  // the Arduino pin, which connects to the IN pin of relay
void setup() {
        pinMode(RELAY_PIN, OUTPUT);
        Serial.begin(9600);
        while (!Serial) { delay(10); }
        Serial.println("Adafruit VEML7700 Test");

        if (!veml.begin()) {
                Serial.println("Sensor not found");
                while (1);
        }
        Serial.println("Sensor found");
```

```
        Serial.print(F("Gain: "));
        switch (veml.getGain()) {
                case VEML7700_GAIN_1: Serial.println("1"); break;
                case VEML7700_GAIN_2: Serial.println("2"); break;
                case VEML7700_GAIN_1_4: Serial.println("1/4"); break;
                case VEML7700_GAIN_1_8: Serial.println("1/8"); break;
        }

        Serial.print(F("Integration Time (ms): "));
        switch (veml.getIntegrationTime()) {
                case VEML7700_IT_25MS: Serial.println("25"); break;
                case VEML7700_IT_50MS: Serial.println("50"); break;
                case VEML7700_IT_100MS: Serial.println("100"); break;
                case VEML7700_IT_200MS: Serial.println("200"); break;
                case VEML7700_IT_400MS: Serial.println("400"); break;
                case VEML7700_IT_800MS: Serial.println("800"); break;
        }
        veml.setLowThreshold(200); //مقدار کمینه مد نظر
        veml.setHighThreshold(500); //مقدار بیشینه مدنظر
        veml.interruptEnable(true); // راه اندازی وقفه
}

void loop() {
        Serial.print("raw ALS: "); Serial.println(veml.readALS());
        Serial.print("raw white: "); Serial.println(veml.readWhite());
        Serial.print("lux: "); Serial.println(veml.readLux());
        uint16_t irq = veml.interruptStatus();
        if (irq & VEML7700_INTERRUPT_LOW) {
                Serial.println("** Low threshold");
                digitalWrite(RELAY_PIN, HIGH);
        }
        if (irq & VEML7700_INTERRUPT_HIGH) {
                Serial.println("** High threshold");
                digitalWrite(RELAY_PIN, LOW);
        }
        delay(2000);
}
```

دستور switch case

دستور switch case همانند ساختار if، جریان برنامه را کنترل می‌کند. ساختار سوییچ کدهای متفاوتی که باید در شرایط مختلف اجرا شوند را نظم دهی می‌کند. به‌طور خاص، ساختار سوییچ، مقدار متغیر را با مقداری که در قسمت case مشخص شده، مقایسه می‌کند. وقتی که مقدار بررسی شده یک case با مقدار متغیر برابر شد؛ کد آن case اجرا می‌شود.

کلمه **break** نیز در این ساختار وجود دارد که معمولاً در انتهای هر case استفاده می‌شود. بدون break، سوییچ تا زمانی که به یک break دیگر یا به انتهای ساختار برسد، کدهای caseهای دیگر را نیز اجرا می‌کند . درواقع break برای خاتمه دادن به case استفاده می‌شود و هرگاه به این کلمه برسیم، برنامه، از خط بعد از بلوک سوییچ اجرا خواهد شد.

حالت اختیاری **default** نیز وجود دارد که اگر هیچ‌کدام از caseها با متغیر برابر نشدند، کدهای این قسمت در صورت وجود داشتن، اجرا می‌شوند.

```
switch (var) {
        case 1://case label:
                // در صورتی که متغیرvarبرابر با ۱ است دستورات نوشته شده در این بخش انجام می شود
        break;
        case 2://case label:
                // در صورتی که متغیرvar برابر با۲ است دستورات نوشته شده در این بخش انجام می شود
        break;
        default:
                // اگر متغیر با هیچکدوم از case ها برابر نبود دستورات قسمت default اجرا می شود
                // قسمت Default اختیاری است.
        break;
}
```

var متغیری است که مقدار آن با کیس‌های مختلف مقایسه می‌شود و label مقداری برای مقایسه با متغیر است.

درس یازدهم: ارسال AT command به ماژول WiFi ESP8266 و شناسایی شبکه‌های Wi-Fi موجود جهت اتصال

پیش نیاز: AT command

برای ارتباط با چیپ ESP8266 روی ماژول ProMake Wi-Fi می‌بایست دستوراتی به نام AT command استفاده کنیم. این دستورات پیام‌های متنی هستند که از طریق ارتباط سریال(UART) ارسال و دریافت می‌گردند.

برخی از دستورات پرکاربرد به شرح زیر است:

دستور AT

دستور AT حضور و سلامت ماژول ESP8266 را تست می‌کند و در صورت موفقیت پاسخ OK دریافت می‌شود.

دستور AT+CWMODE=mode

با دستور AT+CWMODE=mode مدکاری ماژول ESP8266 تعیین می‌گردد.

مقداری قابل ارسال برای پارامتر Mode:

۱ = با درج عدد یک به جای کلمه Mode در دستور بالا، ماژول در حالت کلاینت (client) یا Station قرار می‌گیرد.

۲ = با درج عدد ۲ به جای کلمه Mode در دستور بالا، ماژول در حالت Access point قرار می‌گیرد. (حالت پیش فرض ماژول)

۳ =با درج عدد ۳ به جای کلمه Mode در دستور بالا، ماژول در دو حالت Access point و Station قرار می‌گیرد.

دستور AT+CWLAP

با دستور AT+CWLAP تمامی اکسس پوینت‌های Wi-Fi در دسترس ماژول ESP8266 را به فرمت زیر نمایش می‌دهد.

```
+CWLAP:ecn,ssid,rssi,mac
OK
```

پارامترهای دریافتی:

- **ecn:** روش احراز هویت و رمزنگاری مورد استفاده که شامل موارد زیر می‌باشد:
  - open = ۰
  - WEP = ۱
  - WPA_PSK = ۲
  - WPA2_PSK = ۳
  - WPA_WPA2_PSK = ۴
- **ssid:** نام اکسس پوینت می‌باشد
- **rssi:** قدرت سیگنال را مشخص می‌کند
- **mac:** مک آدرس اکسس پوینت را مشخص می‌کند.

دستور AT+CWJAP="ssid","pwd"

دستور AT+CWJAP="ssid","pwd" اتصال ماژول ESP8266 به شبکه Wi-Fi را با توجه به نام و کلمه عبور مربوطه برقرار می‌کنند.

پارامترهای ارسالی:

- ssid= نام اکسس پوینتی که قصد اتصال به آنرا داریم.
- pwd= پسورد اکسس پوینتی که قصد اتصال به آنرا داریم.

دستور AT+CIPMUX=mode

با ارسال دستور AT+CIPMUX=mode تعیین می شود امکان برقراری چند اتصال همزمان وجود داشته باشد یا خیر.

مقادیر قابل ارسال برای پارامتر mode:

- ۰= فعال کردن تنها یک اتصال
- ۱= فعال کردن چند اتصال همزمان

دستور AT+CIPSTART=id,type,addr,port

با دستور AT+CIPSTART یک اتصال TCP ایجاد می‌شود.

پارامترهای ورودی دستور:

- **id=** این پارامتر نشان دهنده شناسه یا همان شماره اتصال IP می‌باشد که می‌تواند عددی بین ۰ الی ٤ باشد. همچنین در صورتیکه قصد دارید تنها یک اتصال ایجاد کنید می‌توانید از وارد کردن این پارامتر صرف نظر کنید.
- **type=** به جای این کلمه باید نوع پروتکل مورد استفاده را که می‌تواند TCP یا UDP باشد، را وارد کنید.
- **addr=** آدرس IP مقصد را مشخص کنید.

- **port=** شماره پورت مقصد را وارد کنید.

دستور AT+CIPSEND=id,length

با دستور AT+CIPSEND=id,length می‌توان داده‌ی مورد نظر را به صورت یک بسته با حداکثر حجم ۲۰۴۸ بایت در هر بار اجرای دستور ارسال کرد.

پارامترهای ورودی دستور

- **id=** با استفاده از این پارامتر شناسه اتصالی که قرار است برروی آن داده‌ها ارسال شود مشخص می‌کنیم. لازم به ذکر است وارد کردن این پارامتر در صورتی که در وضعیت تک اتصالی هشتیم لازم نیست.
- **length=** این پارامتر مشخص کننده طول رشته داده‌ای است که قصد ارسال آن را داریم. به عنوان مثال برای ارسال کلمه Hello باید عدد ۵ را به جای این پارامتر وارد کنید، چراکه طول رشته Hello پنج است.

پس از ارسال دستور فوق ماژول ESP8266 به ما کاراکتر ">" را برمی‌گرداند که به این معناست که ماژول آماده دریافت داده‌ها جهت ارسال است.

دستور AT+CIPCLOSE=id

با دستور AT+CIPCLOSE=id برای بستن اتصال‌های فعال مورد استفاده قرار می‌گیرد.

پارامترهای ورودی دستور

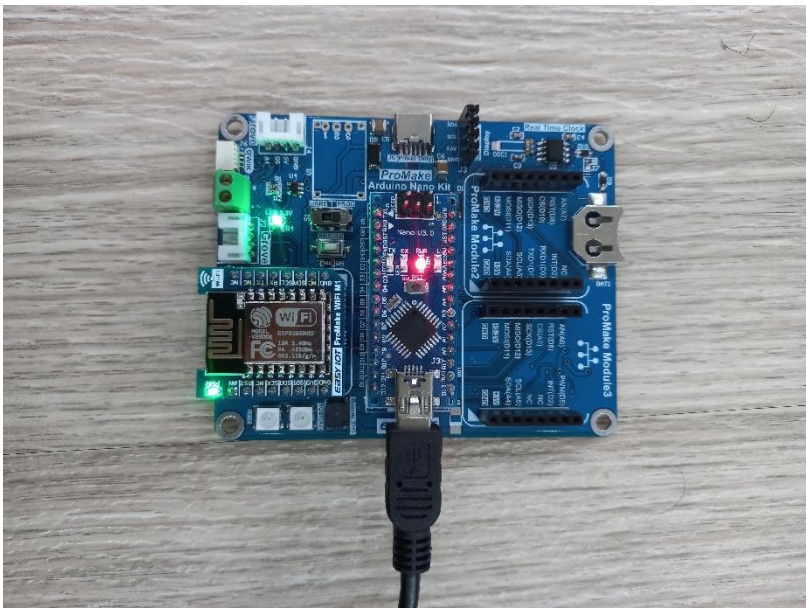
- **id=** این پارامتر تعیین کننده شناسه اتصالی است که قصد قطع ارتباط آنرا داریم و می‌بایست عددی بین ۰ الی ٤ به جای آن وارد کنیم. همچنین در صورتیکه تنها یک کانکشن دارید می‌توانید از وارد کردن این پارامتر صرف نظر کنید.

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- آردوینو نانو + کابل USB
- ماژول ProMake WiFi M1

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake WiFi M1 در جای ProMake Module 1 و ماژول آردوینو نانو بر روی کریر بورد قرار دهید و توسط کابل USB به کامپیوتر متصل کنید.



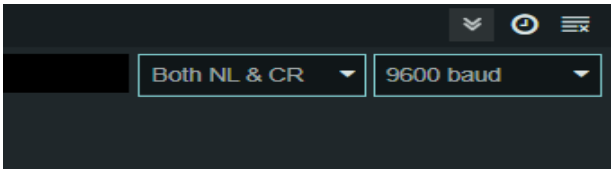
کدنویسی و شرح کد

در محیط Arduino IDE کد زیر را نوشته و اجرا نمایید.

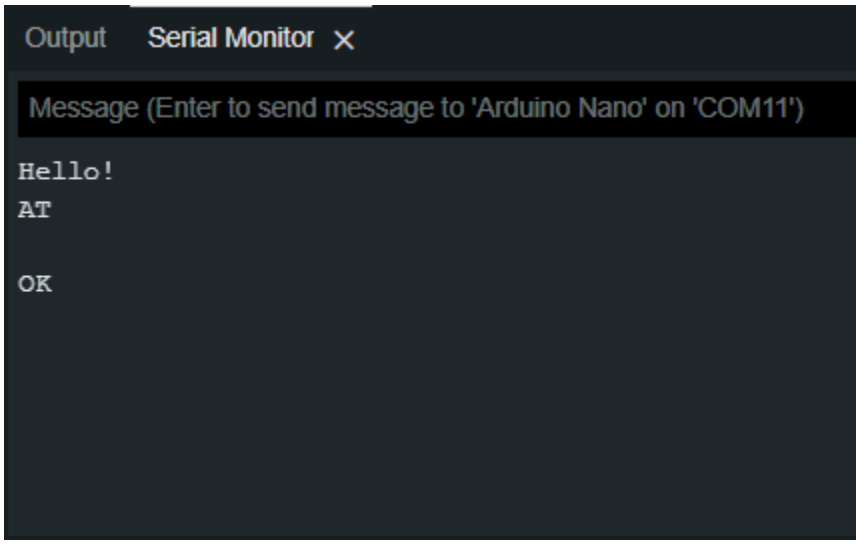
```
#include <SoftwareSerial.h>
#define RX 4
#define TX 7

SoftwareSerial esp8266(RX, TX);

void setup() {
 Serial.begin(9600);
 Serial.println("Hello!");
 esp8266.begin(9600);
}

void loop() {
 if (esp8266.available()) {
  Serial.write(esp8266.read());
 }
 if (Serial.available()) {
  esp8266.write(Serial.read());
 }
}
```

این برنامه ارتباط سریال مانیتور با ماژول Wi-Fi را از طریق UART برقرار می‌کند تا بتوانیم ارسال و دریافت AT command ها را به صورت دستی امتحان کنیم. حال با ارسال هر کدام از دستورات در سریال مانیتور نتایج را مشاهده می‌کنیم.

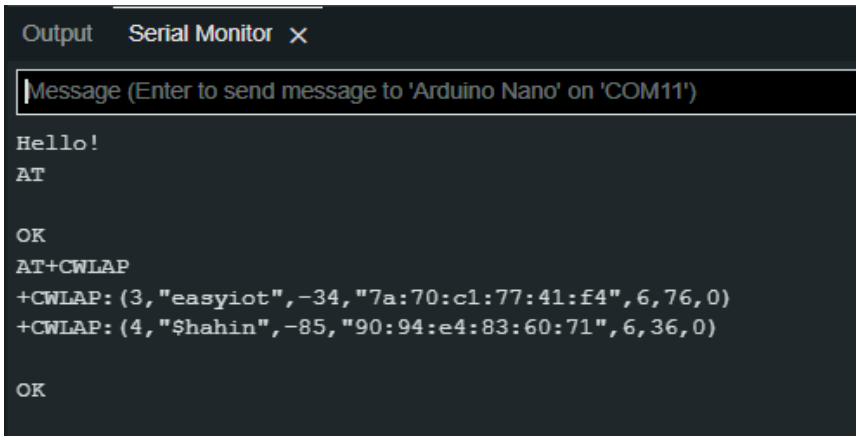
دقت شود تنظیمات سریال مانیتور با باودریت ۹۶۰۰ و مود Both NL & CR به شکل زیر باشد.



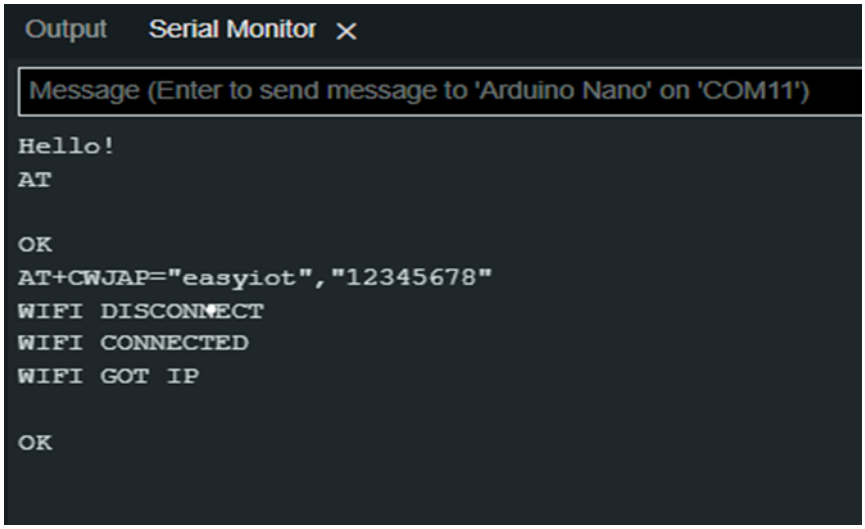
با ارسال دستور AT پاسخ OK را مطابق تصویر زیر دریافت می‌نماییم.



با ارسال دستور AT+CWLAP اکسس پوینت‌های Wi-Fi در دسترس و شناسایی شده، نمایش داده می‌شوند.



و با ارسال دستور "AT+CWJAP="easyiot","12345678 که حاوی نام و کلمه عبور اکسس پوینتی است که قصد اتصال به آن را داریم، ماژول Wi-Fi به آن متصل می‌شود.

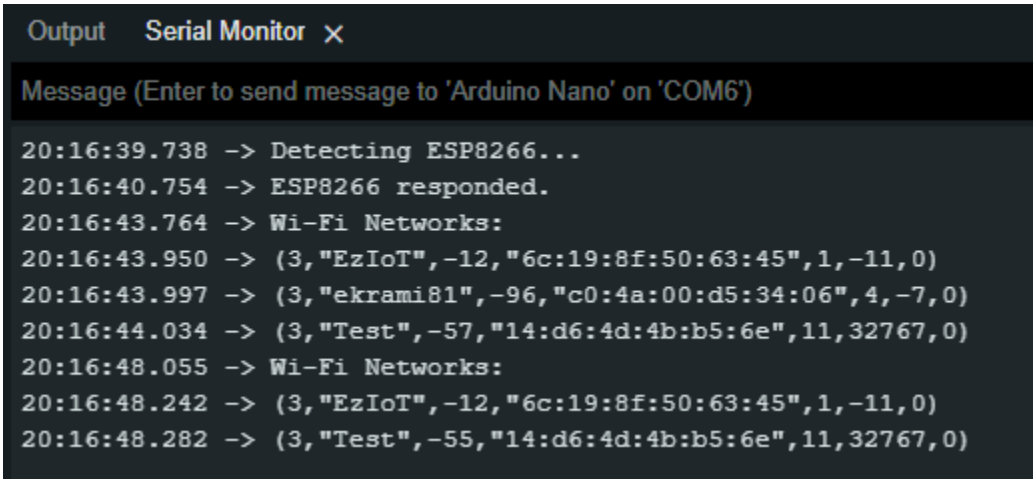


برنامه زیر به صورت خودکار دستورات بالا را اجرا می‌کند.

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266Serial(4, 7); // ایجاد ارتباط سریال نرم افزاری با ماژول
void setup() {
 Serial.begin(9600);     //تنظیم سرعت سریال مانیتور
 esp8266Serial.begin(9600); // تنظیم سرعت ارتباط سریال با ماژول
 delay(1000);
 Serial.println("Detecting ESP8266...");
 esp8266Serial.println("AT"); // ارسال دستور AT جهت اطمینان از سلامت ماژول
 delay(1000);
 if (esp8266Serial.find("OK")) { // بررسی دریافت پاسخ صحیح
  Serial.println("ESP8266 responded.");
 } else {
  Serial.println("ESP8266 NOT responded!");
  while (1);
 }
}
void loop() {
 delay(1000);
 esp8266Serial.println("AT+CWLAP"); // ارسال دستور دریافت لیست اکسس پوینت های در دسترس
 delay(2000);
 if (esp8266Serial.available()) { // بررسی دریافت پاسخ
  Serial.println("Wi-Fi Networks:");
```

```
  while (esp8266Serial.find("+CWLAP:")) { // شناسایی عبارت ابتدای هر شبکه
   String line = esp8266Serial.readStringUntil('\n'); // خواندن اطلاعات دریافتی برای هر شبکه
   Serial.println(line);
  }
 }
}
```

نتیجه در سریال مانیتور به صورت زیر خواهد بود.



پروژه اول: تولید موسیقی و رقص نور

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake(جهت استفاده از RGB LED ها و بازر روی آن)
- آردوینو نانو + کابل USB

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول آردوینو نانو را بر روی کریر بورد در جای خود قرار دهید و توسط کابل USB به کامپیوتر متصل نمایید.

کدنویسی و شرح کد

با استناد و استفاده از توضیحات و کدهای درس چهارم و پنجم در محیط Arduino IDE کد زیر را نوشته و اجرا نمایید.

```
//Music Dynamic Rhythm Lamp
#define NTD0 -1
#define NTD1 294
#define NTD2 330
#define NTD3 350
#define NTD4 393
#define NTD5 441
#define NTD6 495
#define NTD7 556

#define NTDL1 147
#define NTDL2 165
#define NTDL3 175
#define NTDL4 196
#define NTDL5 221
#define NTDL6 248
#define NTDL7 278

#define NTDH1 589
#define NTDH2 661
#define NTDH3 700
#define NTDH4 786
#define NTDH5 882
#define NTDH6 990
#define NTDH7 112
```

```
#define WHOLE 1
#define HALF 0.5
#define QUARTER 0.25
#define EIGHTH 0.25
#define SIXTEENTH 0.625

int tune[] = {
  NTD3, NTD3, NTD4, NTD5,
  NTD5, NTD4, NTD3, NTD2,
  NTD1, NTD1, NTD2, NTD3,
  NTD3, NTD2, NTD2,
  NTD3, NTD3, NTD4, NTD5,
  NTD5, NTD4, NTD3, NTD2,
  NTD1, NTD1, NTD2, NTD3,
  NTD2, NTD1, NTD1,
  NTD2, NTD2, NTD3, NTD1,
  NTD2, NTD3, NTD4, NTD3, NTD1,
  NTD2, NTD3, NTD4, NTD3, NTD2,
  NTD1, NTD2, NTDL5, NTD0,
  NTD3, NTD3, NTD4, NTD5,
  NTD5, NTD4, NTD3, NTD4, NTD2,
  NTD1, NTD1, NTD2, NTD3,
  NTD2, NTD1, NTD1
};

float durt[] = {
  1,1,1,1,
  1,1,1,1,
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,0.5,0.5,1,1,
  1,0.5,0.5,1,1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,0.5,0.5,
  1,1,1,1,
  1+0.5,0.5,1+1,
};

#include <FastLED.h>
```

```
#define LED_PIN 16
#define NUM_LEDS 2
CRGB leds[NUM_LEDS];

int length;
int tonepin = 5;

void setup() {
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
  pinMode(tonepin, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  length = sizeof(tune) / sizeof(tune[0]);
}

void loop() {
  for (int x = 0; x < length; x++) {
    tone(tonepin, tune[x]);
    leds[0] = CRGB(255, 0, 0);
    FastLED.show();
    leds[1] = CRGB(0, 0, 0);
    FastLED.show();
    delay(400 * durt[x]);
    leds[0] = CRGB(0, 0, 0);
    FastLED.show();
    leds[1] = CRGB(0, 255, 0);
    FastLED.show();
    delay(100 * durt[x]);
    noTone(tonepin);
  }
  delay(4000);
}
```

کد بر مبنای تعریف پارامترهای مربوط به موسیقی نظیر فرکانس نت، ضرب موسیقی و تنظیم روشن و خاموش شدن LED ها با پخش و سکوت موسیقی نوشته شده است.

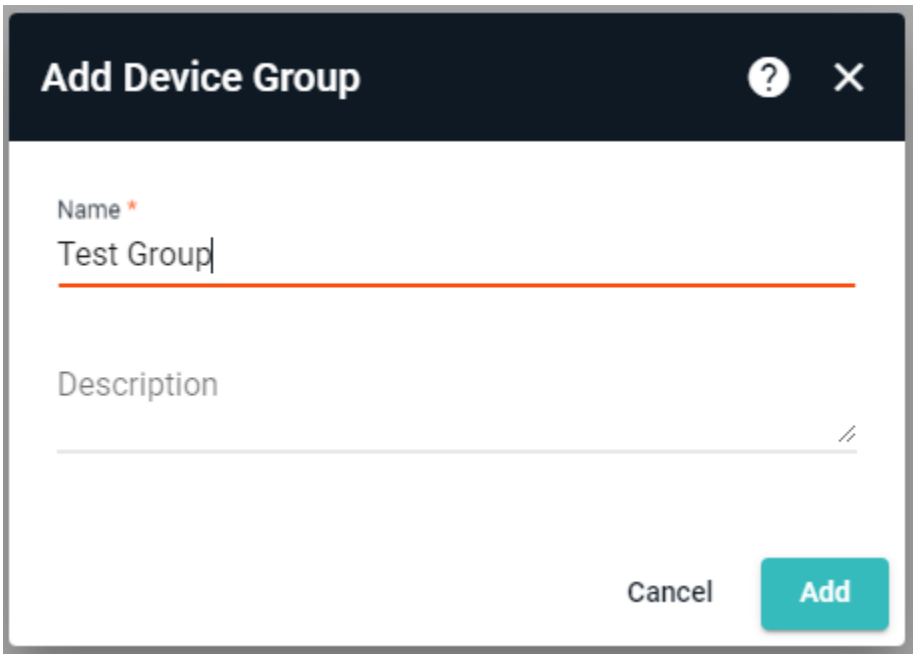
پروژه دوم: ارسال اطلاعات دما و رطوبت، نور محیطی و سطح گازها به داشبرد اینترنتی

پیش نیاز

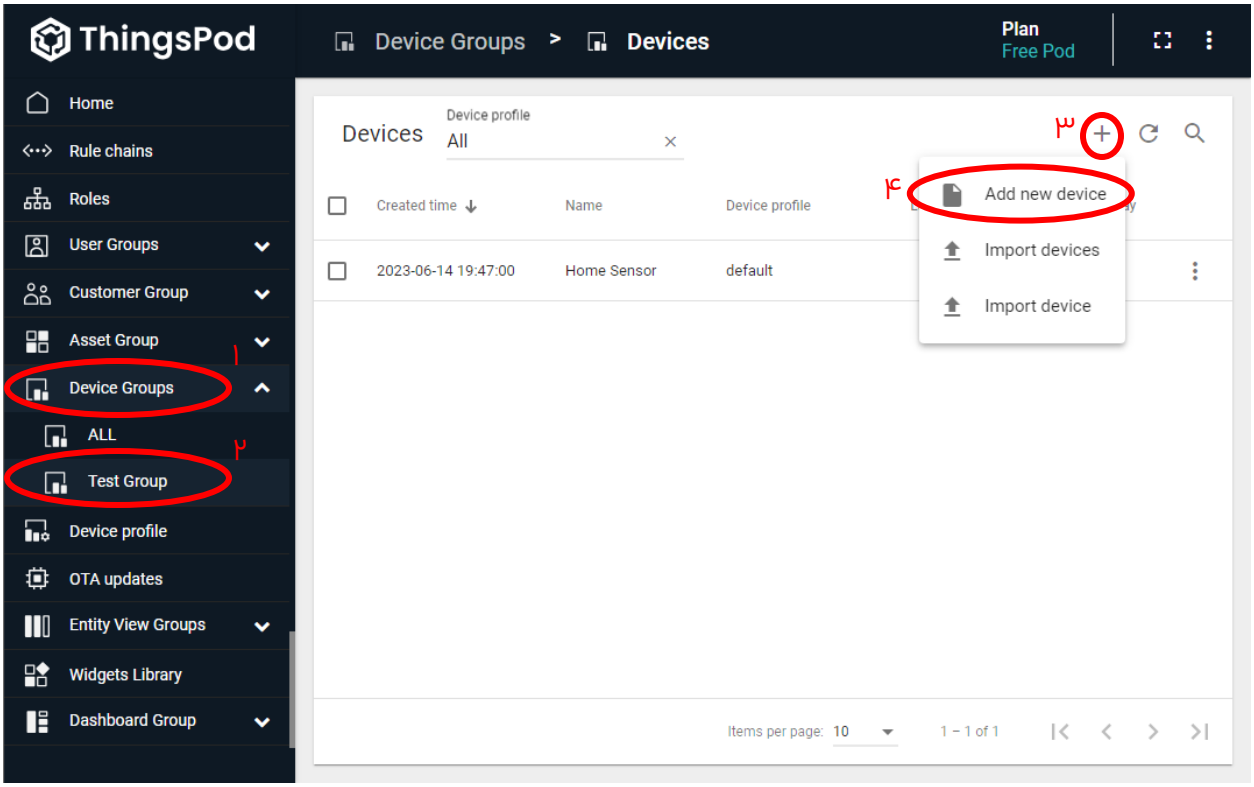
پلتفورم‌های ابری اینترنت اشیاء، زیرساخت انتقال، ذخیره‌سازی و نمایش داده‌ها بین دستگاه‌ها از طریق اینترنت را فراهم می‌کنند. با کمی جستجو پلتفرم‌های ابری اینترنت اشیا متعددی را می‌توانید پیدا کنید. ما در این پروژه قصد داریم با استفاده thingspod که یک پلتفرم ابری اینترنت اشیا ایرانی است که اشیا یا دستگاه‌های فیزیکی را قادر می‌سازد به ابر متصل شوند و به کاربران اجازه می‌دهد تا داده های حسگر خود را تجزیه و تحلیل و تجسم کنند، بیشتر آشنا شویم.

برای استفاده از thingspod ابتدا به سایت https://thingspod.com مراجعه کرده و یک حساب کاربری ایجاد می‌نماییم.

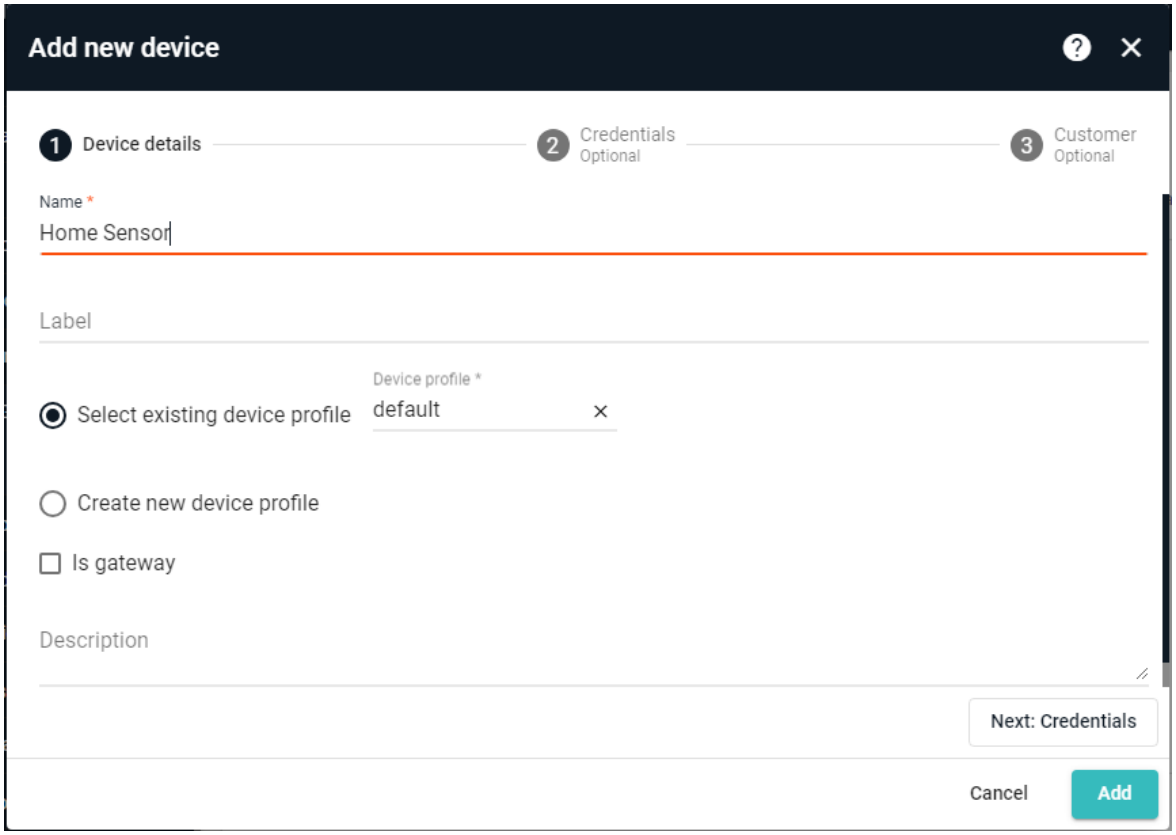
سپس یک گروه جدید به نام Test Group را طبق شکل زیر در بخش Device Groups با زدن دکمه + ایجاد می‌کنیم.



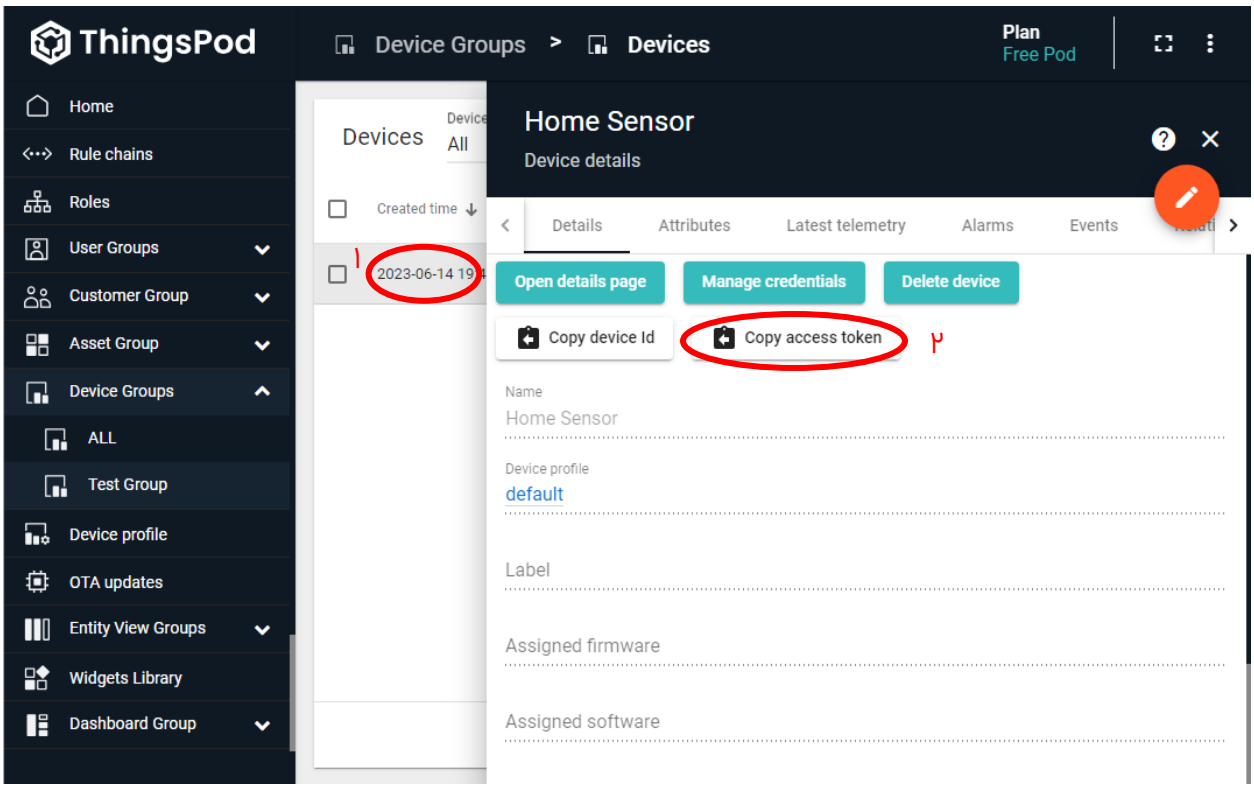
سپس دستگاه جدید را طبق شکل زیر در بخش Device Groups-> Test Group با زدن دکمه + و سپس انتخاب آیتم Add new device ایجاد می‌کنیم.



در پنجره ای که باز می شود نام دستگاه را Home Sensor انتخاب می کنیم و دکمه Add را می فشاریم



حال با کلیک بروی دستگاه ایجاد شده پنجره به صورتی که در شکل زیر مشاهده می کنید نمایان می شود. با زدن برروی دکمه Copy access token می‌توانید کلید امنیتی مورد نیاز برای ارسال داده های دستگاه به پلتفورم را به دست آورید و برای استفاده در کدنویسی نگهداری کنید.



برای ارسال داده‌های حسگرهای دستگاه به پلتفورم می‌بایست آنها با فرمت JSON را به URL زیر ارسال کنیم.

https://thingspod.com/api/v1/ACCESS_TOKEN/telemetry

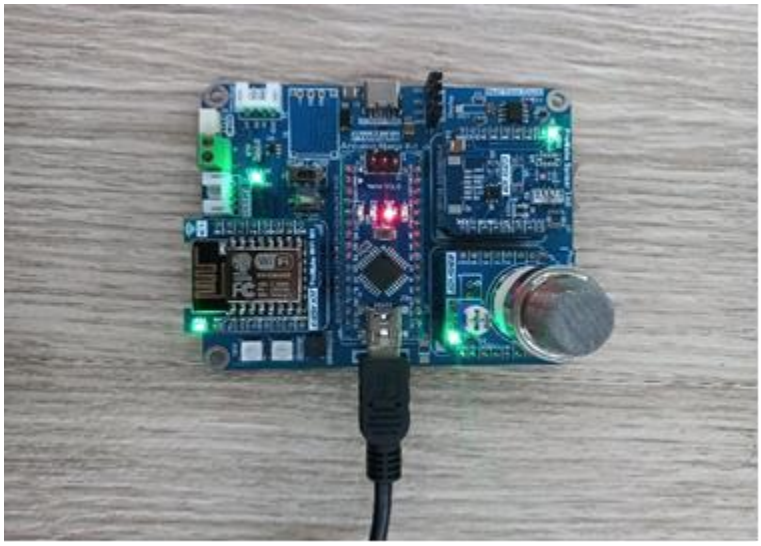
توکن دریافتی در مرحله قبل را باید به جای عبارت ACCESS_TOKEN در URL قراردهیم تا پلتفورم دستگاه ما را شناسایی کند.

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- آردوینو نانو + کابل USB
- ماژول ProMake WiFi M1 در جای ProMake Module 1
- ماژول ProMake Sensor TAG در جای ProMake Module 2
- ماژول ProMake گاز MQ-2 در جای ProMake Module 3

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول‌های فوق و آردوینو نانو را بر روی کریر بورد قرار دهید و توسط کابل USB به کامپیوتر متصل نمایید.



کدنویسی و شرح کد

حال کد زیر را در نرم افزار آردوینو نوشته و جایگزاری های زیر را انجام دهید:

- نام (SSID) شبکه WiFi که قصد اتصال به آن را دارید به جای عبارت YOUR_SSID وارد نمایید.
- کلمه عبور شبکه WiFi که قصد اتصال به آن را دارید به جای عبارت YOUR_PASS وارد نمایید.
- توکن دریافتی از سایت را به جای عبارت YOUR_TOKEN وارد نمایید.

حال کد را برروی آردوینو بارگزاری نمایید.

```
//Sensors ========================================
#include <ProMake_SHT20.h>
ProMake_SHT20 sht20;

#include <ProMake_MQ2.h>
int MQpin = A6; // متغیری که شماره پایه متصل به حسگر گاز را ذخیره می کند
ProMake_MQ2 mq2(MQpin);

#include "ProMake_VEML7700.h"
ProMake_VEML7700 veml = ProMake_VEML7700();

// WiFi UART Connection ===============================
```

```
#include <SoftwareSerial.h>
SoftwareSerial espUART(4, 7); // شی تبادل داده سریال با ماژول وای فای

#define ESP_AT_BAUD 9600
#define ESP_RST_PIN 8

#include <ProMake_AtCommand.h>
ProMake_AtCommand espAT(&espUART); // AT شی ارسال دستورات

#define SSID "YOUR_SSID" // نام WiFi برای اتصال
#define PASS "YOUR_PASS" // رمزWiFi برای اتصال

unsigned long lastConnectionTime = 0;      // زمان آخرین ارتباط موفق با سرور
const unsigned long postingInterval = 10000L; // فاصله زمانی بین دو ارسال داده برای سرور به میلی ثانیه

// IOT Platform ==================================
#define SERVER_NAME "thingspod.com"
#define SERVER_PORT (443)

#define TOKEN "YOUR_TOKEN" // ساخته شده در سایت Write API key

float TempSensor;
float HumidSensor;
float lightSensor;
float gasSensor;

void setup() {
 Serial.begin(9600); //تنظیم ارتباط سریال برای نمایش

 pinMode(MQpin, INPUT);
 pinMode(ESP_RST_PIN, OUTPUT);
 //ESP RESET
 digitalWrite(ESP_RST_PIN, LOW);
 delay(50);
 digitalWrite(ESP_RST_PIN, HIGH);
 delay(2000); // to boot

 // راه اندازی ارتباط سریال با ماژول وای فای
 espUART.begin(ESP_AT_BAUD);
```

```
bool initOK = false;

for (int i = 0; i < 5; i++) {
 if (espAT.sendCmd(F("AT")) == TAG_OK) {
  initOK = true;
  break;
 }
 delay(1000);
}

if (!initOK) {
 Serial.println(F("Cannot initialize ESP module"));
 delay(5000);
 return;
}

// disable echo of commands
espAT.sendCmd(F("ATE0"));

// set station mode
espAT.sendCmd(F("AT+CWMODE=1"));
delay(200);

// set single connections mode
espAT.sendCmd(F("AT+CIPMUX=0"));

// Disable autoconnect
// Automatic connection can create problems during initialization phase at next boot
espAT.sendCmd(F("AT+CWAUTOCONN=0"));

// enable DHCP
espAT.sendCmd(F("AT+CWDHCP=1,1"));
delay(200);

Serial.println(F("Connecting to WiFi..."));
int ret = espAT.sendCmd(F("AT+CWJAP_CUR=\"%s\",\"%s\""), 20000, SSID, PASS);
if (ret == TAG_OK) {
 Serial.println(F("WiFi Connected"));
```

```
 } else {
  Serial.println(F("Failed connecting WiFi"));
 }

 Wire.begin();
 sht20.checkSHT20(); // شروع به کار سنسور دما و رطوبت
 mq2.begin();      // انجام فرایند کالیبراسیون
 veml.begin();     // شروع به کار سنسور نور محیطی
}

char request[256] = "";
char payload[80] = "";

void loop() {

 TempSensor = sht20.readTemperatureC();
 char tempStr[8];
 dtostrf(TempSensor, 4, 2, tempStr);
 HumidSensor = sht20.readHumidity();
 char humidStr[8];
 dtostrf(HumidSensor, 4, 2, humidStr);
 veml.getALSLux(lightSensor);
 char lightStr[8];
 dtostrf(lightSensor, 4, 2, lightStr);
 gasSensor = mq2.readCO();
 char gasStr[8];
 dtostrf(gasSensor, 4, 2, gasStr);

 // بررسی توالی زمانی بین دو ارسال موفق
 if (millis() – lastConnectionTime > postingInterval) {
  memset(payload, '\0', sizeof(payload));
  sprintf_P(payload, PSTR("{\"Temp\":%s, \"Humid\":%s, \"Light\":%s, \"Gas\":%s}"), tempStr, humidStr,
lightStr, gasStr);

  memset(request, '\0', sizeof(request));
  sprintf_P(request, PSTR("POST /api/v1/%s/telemetry HTTP/1.1\r\nHost: thingspod.com\r\nUser–Agent:
ESP8266/1.0\r\nContent–Type: application/json\r\nContent–Length: %d\r\nConnection:
close\r\n\r\n%s\r\n"), TOKEN, strlen(payload), payload);
```

```
  send_tcp(request);
 }
 delay(1000);
}

void send_tcp(const char *request) {
 int ret = espAT.sendCmd(F("AT+CIPSSLSIZE=4096"));

 Serial.println(F("Connecting ..."));
 // بازکردن اتصال
 ret = espAT.sendCmd(F("AT+CIPSTART=\"SSL\",\"%s\",%u"), 10000, SERVER_NAME, SERVER_PORT);
 if (ret == TAG_CONNECT) {
  // ارسال دستور شروع ارسال داده
  char cmdBuf[20];
  sprintf_P(cmdBuf, PSTR("AT+CIPSEND=%u"), strlen(request));
  espUART.println(cmdBuf);

  // دریافت تاییدا از برای شروع ارسال داده
  int idx = espAT.readUntil(2000, (char *)">", false);
  if (idx != NUMESPTAGS) {
   Serial.println(F("Data packet send error (1)"));
   ret = espAT.sendCmd(F("AT+CIPCLOSE"), 4000);
   return false;
  }
  Serial.println(F("Sending ..."));
  // ارسال داده
  espUART.print(request);

  // دریافت تایید ارسال داده از ESP8266
  idx = espAT.readUntil(4000);
  if (idx != TAG_SENDOK) {
   Serial.println(F("Data packet send error (2)"));
   ret = espAT.sendCmd(F("AT+CIPCLOSE"), 4000);
   return false;
  }

  //دریافت پاسخ از سرور
  idx = espAT.readUntil(2000, "CLOSED\r\n", false);
  if (idx != NUMESPTAGS) {
```

```
    Serial.println(F("Tag CLOSED not found"));
    ret = espAT.sendCmd(F("AT+CIPCLOSE"), 4000);
    return false;
  }
  lastConnectionTime = millis();
  Serial.println(F("Successful"));
 } else {
  Serial.println(F("Failed connecting "));
 }
}
```

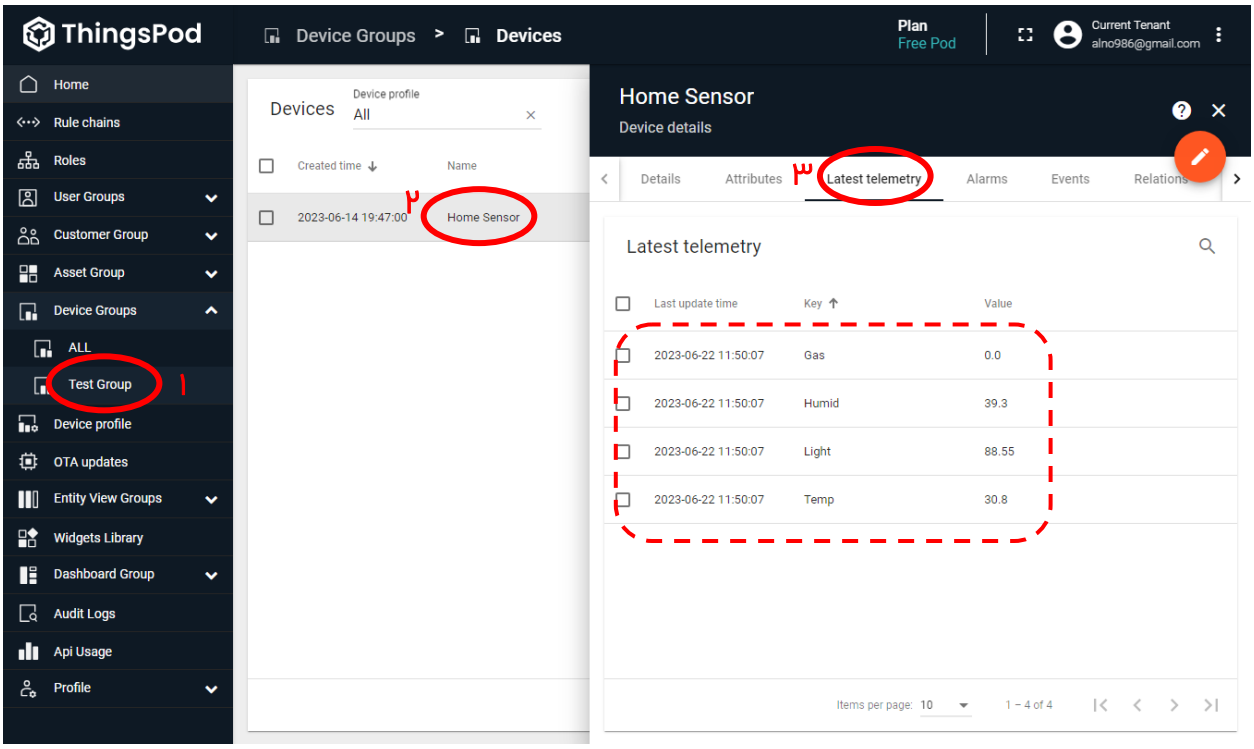
در کد نوشته شده از کتابخانه شرکت سازنده کیت و کدهایی که قبلاً برای سنسورهای مختلف شرح داده شد استفاده شده است.

برای راه اندازی ماژول ProMake WiFi M1 با ارسال AT Command ها ابتدا ماژول را در حالت Station قرار دادیم تا امکان اتصال به Access Point را داشته باشد. در دستور بعدی ماژول را در حالت تک اتصالی تنظیم کردیم چرا که در آن واحد به یک اتصال بیشتر نیاز نداریم. سپس دریافت خودکار تنظیمات شبکه از طریق DHCP را فعال نمودیم و در نهایت دستور اتصال به Access Point مورد نظر را ارسال نمودیم.

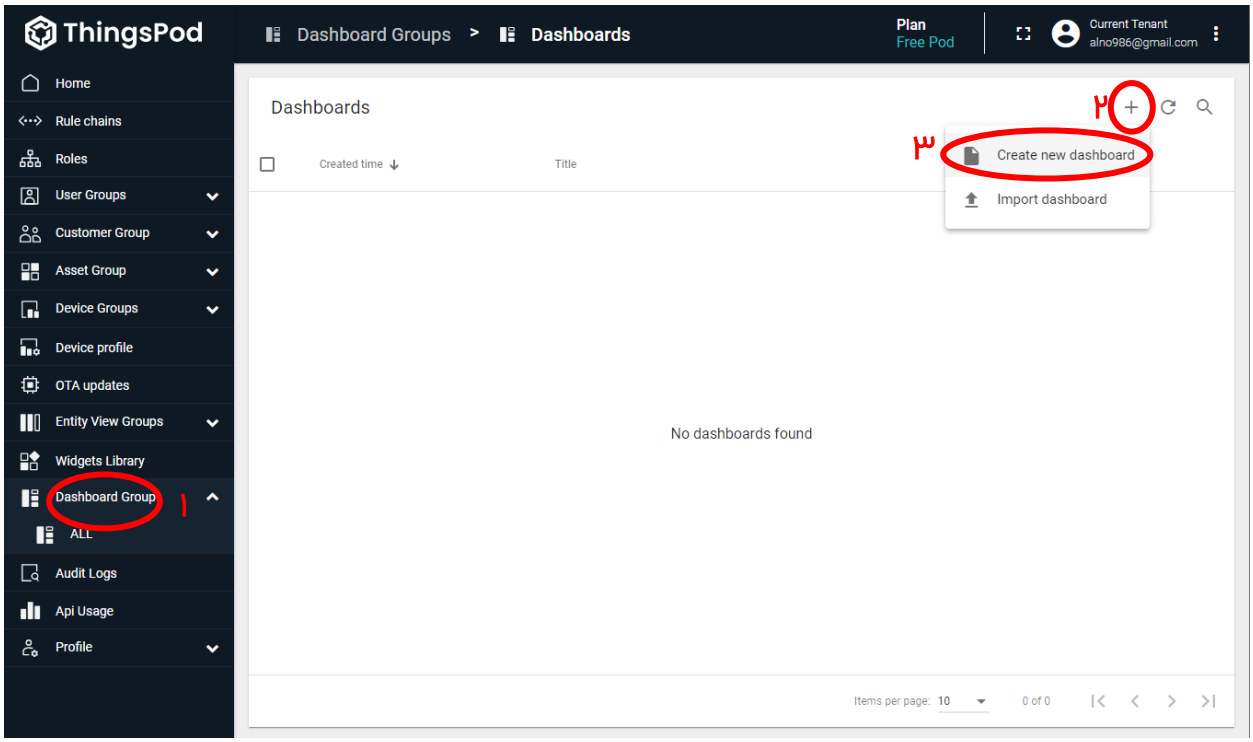
در تابع loop ابتدا مقادیر حسگرها دریافت کرده و سپس بررسی می‌کنیم آیا زمان سپری شده از ارسال قبلی از فاصله زمانی مدنظر برای ارسال بعدی بیشتر شده یا نه. در صورتی که فاصله زمانی ارسال گذشته بود ابتدا داده‌های دریافتی از حسگرها را در قالب یک عبارت JSON در آرایه رشته‌ای payload تنظیم می‌کنیم. سپس ساختار یک درخواست HTTP برای ارسال دادها به سرور را در آرایه رشته‌ای request تنظیم می‌کنیم. تا در نهایت توسط تابع send_tcp درخواست را برای سرور ارسال می‌کنیم.

تابع send_tcp با ارسال دستورات AT مورد نیاز یک اتصال TCP با سرور برقرار کرده و داده ها را ارسال کرده و پاسخ سرور را دریافت می‌کند.

با اجرای کد فوق برروی آردوینو، پس از اتصال موفق به شبکه داده ها حدوداً هر ۱۰ ثانیه یک بار برای پلتفورم ارسال شده و نتیجه قابل مشاهده در سایت در بخش Latest Telemetry مشابه تصویر زیر است. می‌توانید مقادیر ارسال شده توسط حسگرها برای سرور را مشاهده کرده و به روز شدن آنها را با زمان ثبتی مشاهده کنید.

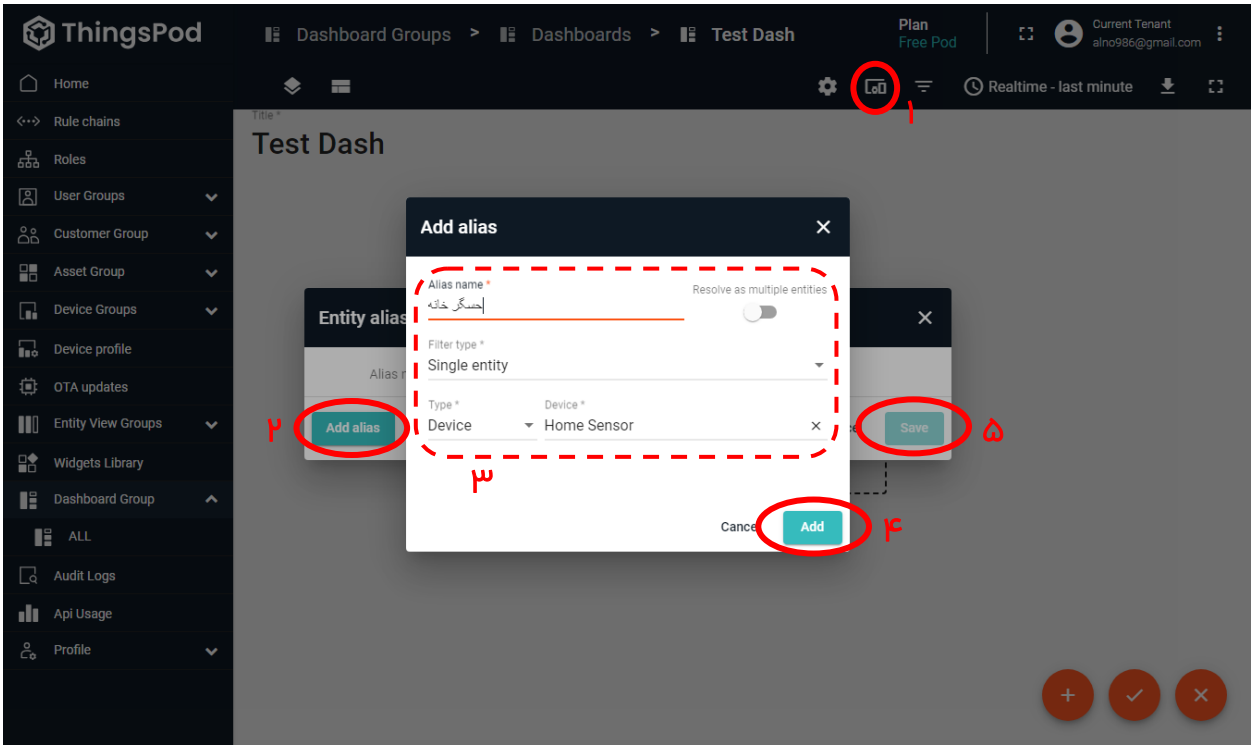


حال به بخش Dashboard Group بروید و در گروه ALL یک داشبرد جدید با نام دلخواه بسازید.

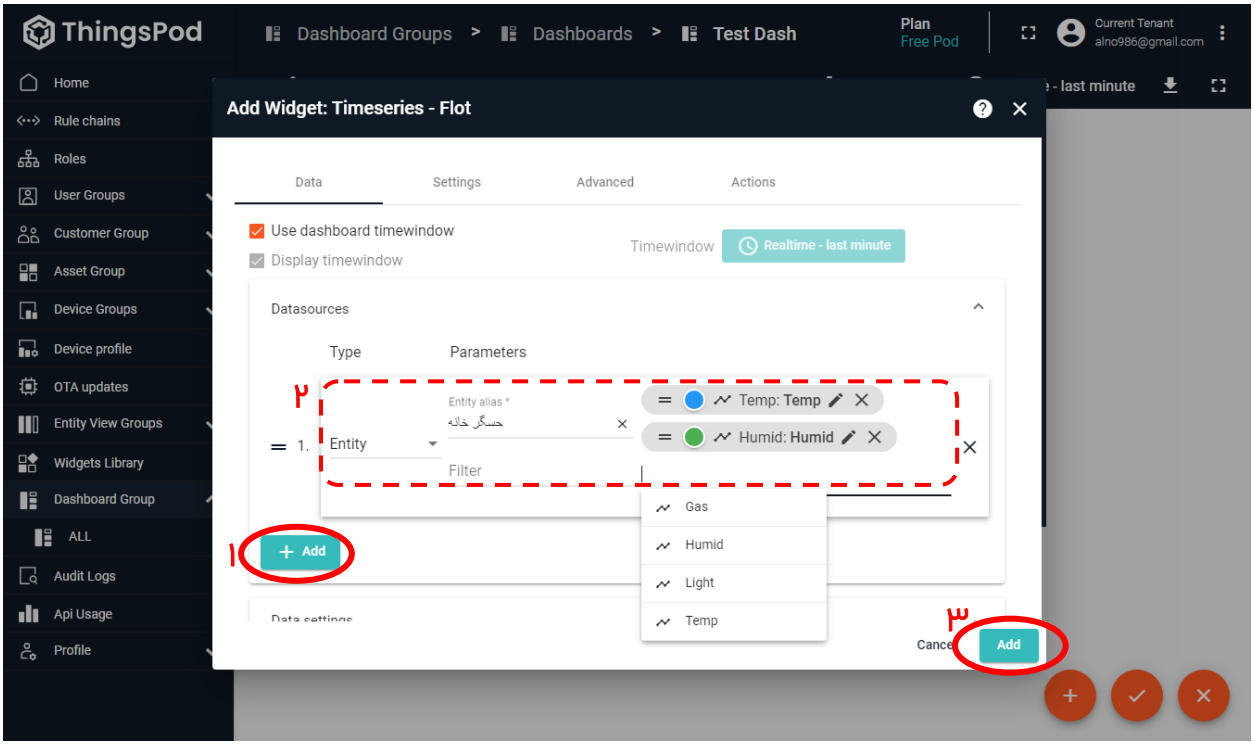


سپس داشبردی را که ایجاد نمودید باز کرده و برروی دکمه ویرایش کلیک نمایید، سپس مطابق تصویر برروی دکمه Entity Alias کلیک کرده و در پنجره باز شده برروی دکمه Add alias کلیک کرده و حسگری

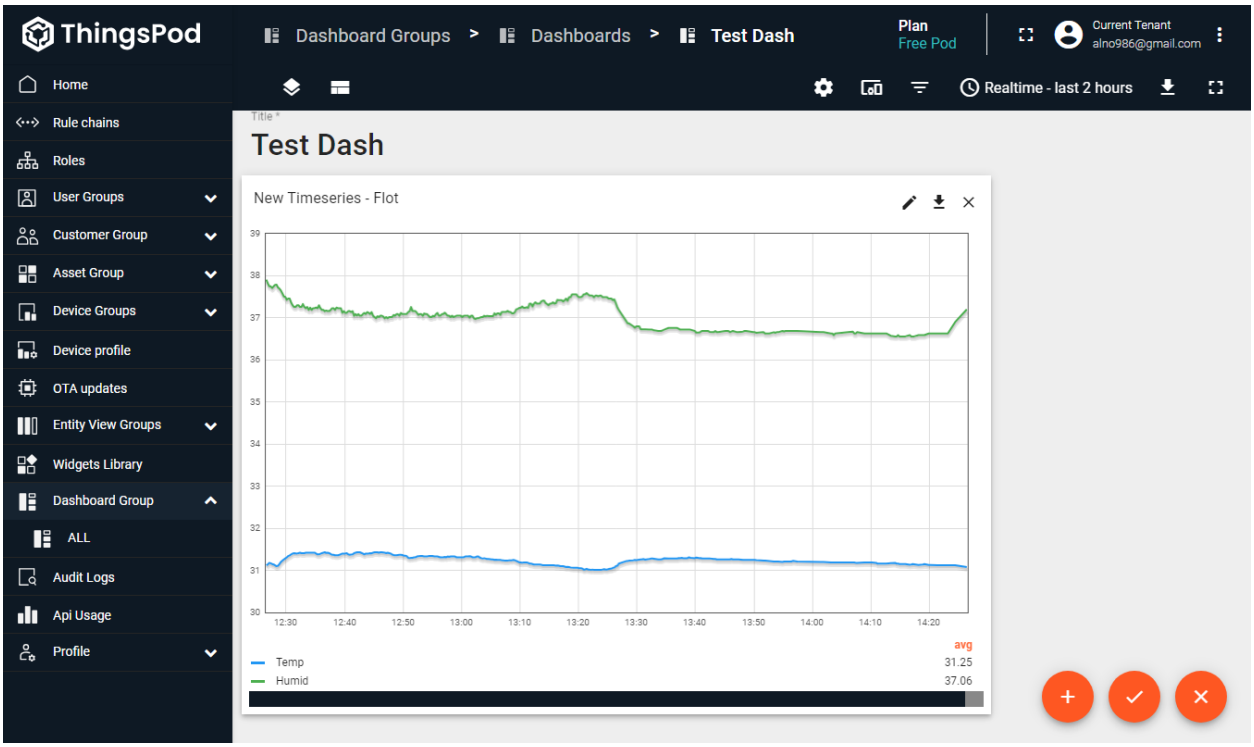
که در مرحله قبل ساخته بودیم را به عنوان یکSingle entity از نوع Device با نام "حسگر خانه" اضافه کنید. سپس برروی دکه Save کلیک نمایید.



حال برروی دکمه Add new widget کلیک کرده و ازگروه Charts یک Timeseries – Plot اضافه نمایید.



حال مطابق تصویر موجودیت "حسگر خانه" را انتخاب کرده و متغییرهای دما و رطوبت را جهت نمایش انتخاب کنید. با زدن دکمه Add ویجت نمودار برروی داشبرد نمایش داده می شود، حال می توانید اندازه آن را به صورت دلخواه در آورید و وضعیت تغییرات متغیرهای فوق را در بازه زمانی دلخواه مشاهده نمایید.



می‌توانید به این داشبرد ویجت‌های متنوعی به تناسب نیاز و سلیقه خود اضافه نمایید. همچنین این پلتفورم ابری اینترنت اشیاء امکانات متعددی دارد که می‌توانید با مطالعه مستندات آن با آنها آشنا شوید و مورد استفاده قرار دهید.

پروژه سوم: کنترل رله به صورت هوشمند(براساس دما و رطوبت یا نور محیط) و توسط گوشی

اقلام مورد نیاز

- کریر بورد آردوینو نانو ProMake
- آردوینو نانو + کابل USB
- ماژول ProMake WiFi M1 در جای ProMake Module 1
- ماژول ProMake Sensor TAG در جای ProMake Module 2
- ماژول ماژول ProMake رله تک کانال در جای ProMake Module 3

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول‌های فوق و آردوینو نانو را بر روی کریر بورد قرار دهید و توسط کابل USB به کامپیوتر متصل نمایید.

کدنویسی و شرح کد

کد نوشته شده برای کنترل هوشمند رله به صورت زیر است. دقت کنید که سطح آستانه مدنظر برای هرکدام از سنسورها قابل تنظیم است.

```
#include <ProMake_SHT20.h>
ProMake_SHT20 sht20;

#include "ProMake_VEML7700.h"
ProMake_VEML7700 veml = ProMake_VEML7700();

int RELAY_PIN = 14;  // پایه متصل به رله

float TempSensor;
float HumidSensor;
float lightSensor;

float TempSensorThreshold = 50;
float HumidSensorThreshold = 50;
float lightSensorThreshold = 100;

void setup() {
 Serial.begin(9600);  //تنظیم ارتباط سریال برای نمایش

 pinMode(RELAY_PIN, OUTPUT);

 Wire.begin();
 sht20.checkSHT20();  // شروع به کار سنسور دما و رطوبت
 veml.begin();      // شروع به کار سنسور نور محیطی
}

void loop() {

 TempSensor = sht20.readTemperatureC();
 HumidSensor = sht20.readHumidity();
 veml.getALSLux(lightSensor);

 /*
 Serial.println((String)TempSensor + "°C");
 if (TempSensor > TempSensorThreshold) {
```

```
  digitalWrite(RELAY_PIN, HIGH);
 }
 if (TempSensor < TempSensorThreshold) {
  digitalWrite(RELAY_PIN, LOW);
 }
 */
 /*
 Serial.println((String)HumidSensor + " %RH");
 if (HumidSensor > HumidSensorThreshold) {
  digitalWrite(RELAY_PIN, HIGH);
 }
 if (HumidSensor < HumidSensorThreshold) {
  digitalWrite(RELAY_PIN, LOW);
 }
 */

 Serial.println((String)lightSensor + " Lux");
 if (lightSensor < lightSensorThreshold) {
  digitalWrite(RELAY_PIN, HIGH);
 }
 if (lightSensor > lightSensorThreshold) {
  digitalWrite(RELAY_PIN, LOW);
 }
}
```

همان طور که مشاهده می‌نمایید در کد بخش‌هایی برای کنترل رله براساس اطلاعات هر سنسور قرار گرفته که کامنت شده است و فقط بخش کنترل با نور فعال است. با کامنت کردن یا کامنت نکردن کدهای هر قسمت رله با توجه به سنسور مد نظر شما کنترل می‌شود. البته می‌توانید ترکیب شرطی دلخواه خود را برای کنترل رله براساس ترکیب داده‌های حسگرها نیز بنویسید.

برای کنترل رله از طریق گوشی با استفاده از Wi-Fi ، ماژول رله، ماژول Wi-Fi و آردوینو نانو را بر روی کریر بورد قرارداده و به کامپیوتر متصل می‌کنیم. حال کد زیر را برروی آردوینو نانو بارگزاری نمایید.

```
// WiFi UART Connection ================================
#include <SoftwareSerial.h>
SoftwareSerial espUART(4, 7);  // شی تبادل داده سریال با ماژول وای فای

#define ESP_AT_BAUD 9600
#define ESP_RST_PIN 8
```

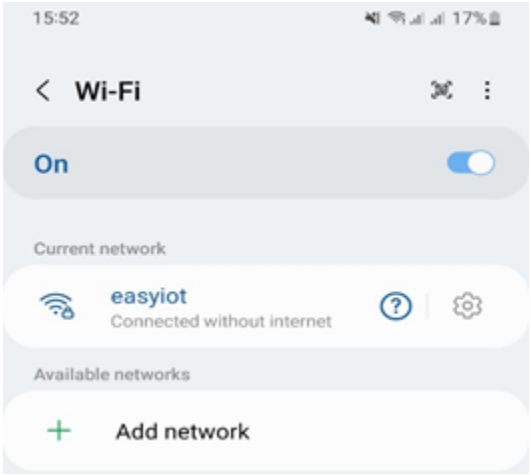
```
#include "ProMake_AtCommand.h"
ProMake_AtCommand espAT(&espUART); // شی ارسال دستورات AT

#define SSID "easyiot"  // نام WiFi برای اتصال
#define PASS "12345678" // رمز WiFi برای اتصال
#define SERVER_IP "192.168.4.1"

int RELAY_PIN = 14; // the Arduino pin, which connects to the IN pin of relay

void setup() {
 Serial.begin(9600); //تنظیم ارتباط سریال برای نمایش

 pinMode(RELAY_PIN, OUTPUT);
 pinMode(ESP_RST_PIN, OUTPUT);
 //ESP RESET
 digitalWrite(ESP_RST_PIN, LOW);
 delay(50);
 digitalWrite(ESP_RST_PIN, HIGH);
 delay(2000); // to boot

 // راه اندازی ارتباط سریال با ماژول وای فای
 espUART.begin(ESP_AT_BAUD);

 bool initOK = false;

 for (int i = 0; i < 5; i++) {
  if (espAT.sendCmd(F("AT")) == TAG_OK) {
   initOK = true;
   break;
  }
  delay(1000);
 }

 if (!initOK) {
  Serial.println(F("Cannot initialize ESP module"));
  delay(5000);
  return;
 }
```

```
 // disable echo of commands
 espAT.sendCmd(F("ATE0"));

 // set softAP mode
 espAT.sendCmd(F("AT+CWMODE=2"));
 delay(200);

 // set Multiple connections mode
 espAT.sendCmd(F("AT+CIPMUX=1"));

 // تنظیم پورت سرور
 espAT.sendCmd(F("AT+CIPSERVER=1,8888"));

 // enable DHCP
 espAT.sendCmd(F("AT+CIPAP=\"%s\""), 20000, SERVER_IP);
 delay(200);

 Serial.print(F("Starting WiFi AP ... "));
 int ret = espAT.sendCmd(F("AT+CWSAP=\"%s\",\"%s\",1,3"), 20000, SSID, PASS);
 if (ret == TAG_OK) {
  Serial.println(F("Successfull"));
 } else {
  Serial.println(F("Failed"));
 }
}

void loop() {
 if (espUART.available()) {
  read_data();
 }
}

int start_search = 0;
String input_data;
String Message;
// تابع read_data طبق کلمه ی ارسال شده از طریق wifi دستور مورد نظر ما را اجرا می کند
void read_data() {
 input_data = espUART.readString();
```

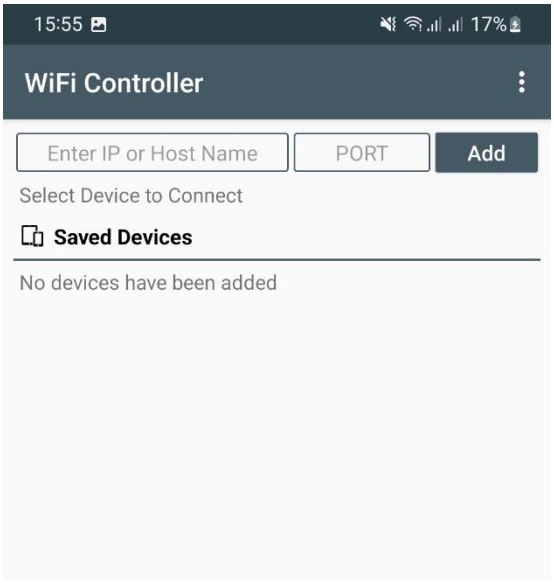
```
 Serial.println(input_data);
 input_data.trim();
 start_search = input_data.indexOf(":");
 start_search++;
 input_data = input_data.substring(start_search);
 Message = "";
 if (input_data == "on") {
  digitalWrite(RELAY_PIN, HIGH);
  Message = "Relay ON";
 } else if (input_data == "off") {
  digitalWrite(RELAY_PIN, LOW);
  Message = "Relay OFF";
 }
 if (Message != "") {
  Message = Message + " | ";
  char cmdBuf[20];
  sprintf_P(cmdBuf, PSTR("AT+CIPSEND=0,%u"), Message.length());
  espUART.println(cmdBuf);

  // دریافت تاییدا زبرای شروع ارسال داده
  int idx = espAT.readUntil(2000, (char *)">", false);
  if (idx != NUMESPTAGS) {
   Serial.println(F("Data packet send error"));
   return;
  }
  // ارسال داده
  espUART.println(Message);
 }
}
```

سپس ابتدا Wi-Fi گوشی را روشن می‌کنیم. در بین شبکه‌های کشف شده می‌بایست شبکه‌ای با نام easyiot شناسایی شده باشد. حال مطابق تصویر زیر به شبکه easyiot با رمز 12345678 متصل می‌شویم. حال که به شبکه Access Point ساخته شده توسط ماژول متصل شده ایم نیاز داریم با استفاده از اپلیکیشن WiFi Controller ESp8266 [۱۰] به دستگاه با یک اتصال TCP متصل شویم تا فرامین لازم را ارسال و پاسخ‌های مربوطه را دریافت کنیم.

[۱۰] https://cafebazaar.ir/app/com.mightyit.gops.wificontroller



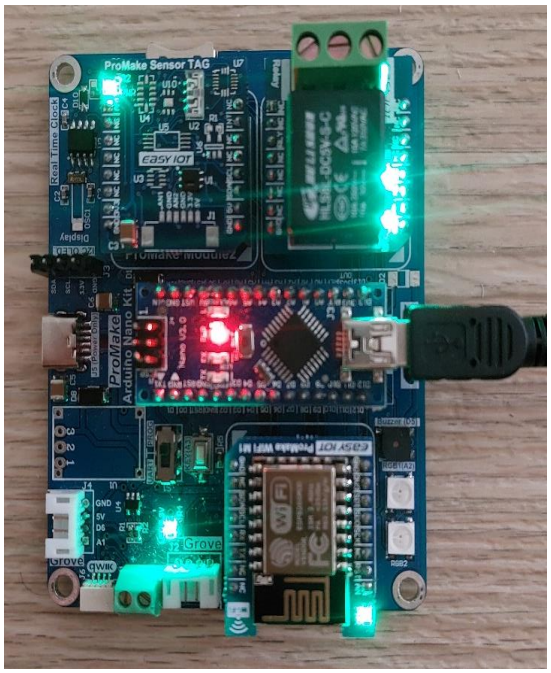
پس اپلیکیشن WiFi Controller ESP8266 [1] را بر روی گوشی نصب و اجرا می‌کنیم. محیط برنامه به شکل زیر است.

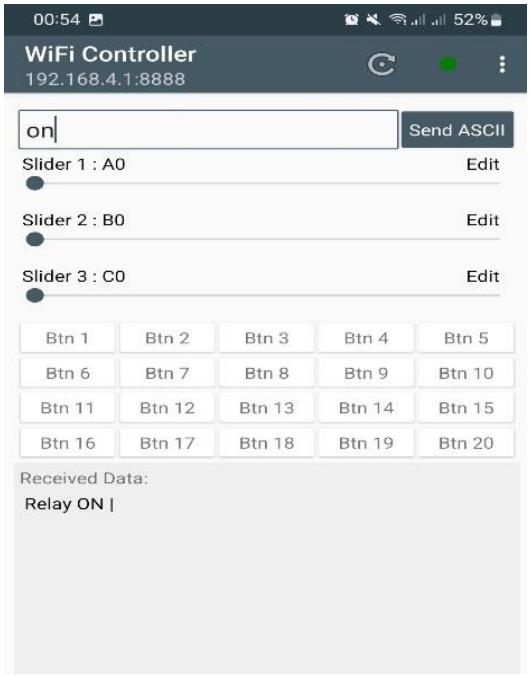


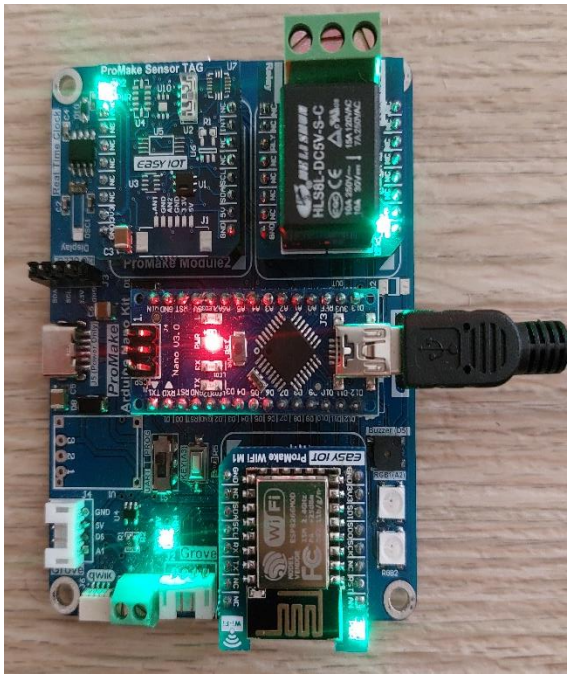
ابتدا مطابق آدرسی که در برنامه آردوینو تعریف کردیم IP را برابر 192.168.4.1 و پورت را 8888 وارد کرده و دستگاه جدید اضافه می کنیم.

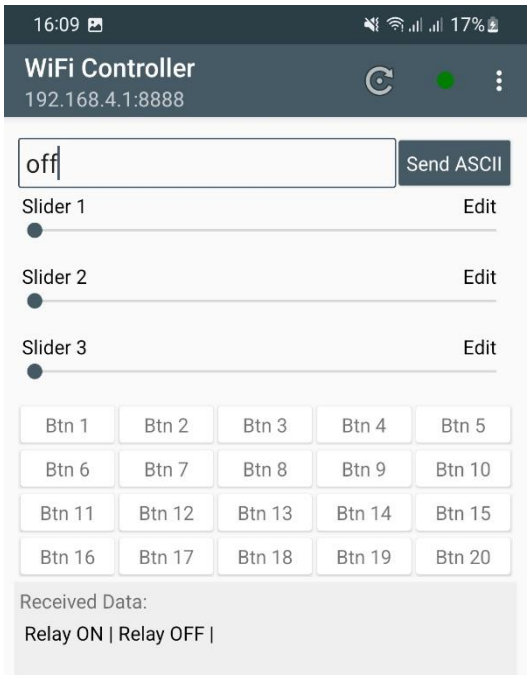
[1] https://cafebazaar.ir/app/com.mightyit.gops.wificontroller

با کلیک بر روی دستگاه ایجاد شده امکان ارسال فرامین به صورت‌های مختلف مهیا است و در برنامه فوق مطابق آنچه در کد آردینو نوشتیم با فرستادن پیام on رله روشن و با فرستادن off خاموش می‌شود.






ProMake Sensor TAG
ProMake Module2
ProMake Wifi M1


16:09
17%
WiFi Controller
192.168.4.1:8888
off
Send ASCII
Slider 1
Edit
Slider 2
Edit
Slider 3
Edit
Btn 1
Btn 2
Btn 3
Btn 4
Btn 5
Btn 6
Btn 7
Btn 8
Btn 9
Btn 10
Btn 11
Btn 12
Btn 13
Btn 14
Btn 15
Btn 16
Btn 17
Btn 18
Btn 19
Btn 20
Received Data:
Relay ON | Relay OFF |