آموزش مقدماتی رزبری پیکو و اینترنت اشیاء

با استفاده از کیت ProMake®

براساس میکروپایتون

تهیه کننده

شرکت دانش بنیان گیگا پرداز پارس

Contents

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 خوردکار از گیاهان شما مراقبت کرده و زمانی که آنها به آب نیاز دارند را اطلاع بدهد؟ و یا هزاران هزار ایده‌ی دیگر؟ رزبری پیکو ابزاری قدرتمند است که امکان طراحی و ساخت ایده‌های مارا به آسان‌ترین شکل ممکن، فراهم می‌کند. بسته اینترنت اشیا ProMake مبتنی بر میکرو پایتون، مجموعه‌ای از سخت افزارهای مورد نیاز را به صورت ماژول‌های تست شده در اختیار شما قرار داده و به شما امکان پیاده‌سازی گستره‌ی وسیعی از پروژه‌ها را در کوتاه‌ترین زمان ممکن و بدون نیاز به یادگیری طراحی برد، خرید جداگانه حسگرها برای کاربردهای خاص، سیم‌کشی میان قطعات و... می‌دهد.

خانواده رزبری پیکو، یک نرم‌افزار متن باز را برای برنامه‌نویسی بردهای خود معرفی می‌کند 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 رزبری پیکو 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.

این درسنامه برای چه کسانی مناسب است؟

کلیه افرادی که دارای دانش اولیه الکترونیک و نیز آشنایی به مفاهیم برنامه نویسی را دارند می توانند از این این درسنامه بهمراه سخت افزارهای مرتبط شرکت گیگاپرداز پارس استفاده نمایند. دانش پذیران گرامی با داشتن مهارت اولیه براحتی با دنبال کردن دروس و انجام پروژه ها به سطح مهارت مناسبی برای دوره های آتی و نهایتا انجام پروژه های واقعی در زمینه اینترنت اشیاء، رباتیک و هوشمندسازی خواهند رسید.

با توجه به اینکه طراحی سخت افزارها بصورت ماژولار انجام شده است و نیاز به سیم بندی و لحیم کاری نیست لذا خطرات مرتبط برای ارائه این بسته به گروه کودکان علاقه مند نیست و با توجه به پیش مقدمات اشاره شده و با استفاده از یک مربی امکان ارائه به گروههای سنی مختلف می باشد.

برنامه های اشاره شده در این درسنامه در این آدرس زیر قابل دسترس می باشد.

https://github.com/easyiot-official/ProMake-MicroPython/tree/main/Final%20Firmware_Basic%20Kit%20v1

محتویات بسته

ProMake PICO Carrier	بستر قرارگیری ماژول رزبری پیکو و ماژول‌های ProMake در کنار یکدیگر
ProMake Sensor TAG	ماژول دارای سنسور های 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

کریر بورد ProMake PICO Carrier

کریر برد توسعه مقدماتی ماژولار رزبری پیکو ProMake اولین و تنها برد توسعه‌ای Raspberry Pico طراحی و ساخته شده در کشور عزیزمان ایران است که می‌تواند میزبان ماژول رزبری پیکو (هر سه مدل معمولی، PICO W, PICO H) شما در کنار سه ماژول ProMake باشد. با داشتن " ProMake Raspberry Pico Basic Carrier" قادر خواهید بود با استفاده از ماژول‌های 100% تست شده‌ی ProMake (بدون نیاز به هیچ گونه لحیم کاری و یا طراحی مدارات الکترونیکی) سخت‌افزار 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 Raspberry Pico خود بارگذاری نمایید و شاهد نتایج آن باشید.

با حذف چالش‌های ساخت و راه‌اندازی سخت‌افزار، این کریر برد سرعت توسعه پروژه شما را تا 10 برابر افزایش می‌دهد. از این رو گزینه‌ای ایده‌آل برای نمونه‌سازی سریع ایده‌ها و انجام پروژه‌ها در حوزه اینترنت اشیاء و رباتیک می‌باشد.ماژول Raspberry Pico 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 راه‌اندازی مدار معطوف خواهد کرد. یک اتصال اشتباه می‌توان باعث سوخت و خراب شدن رزبری پیکو 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. 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 سخت‌افزار 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 مقدماتی ماژولار رزبری پیکو پرومیک" این مشکلات را حل کردیم و شما را از شر breadboard وسیم کشی‌ها درهم و برهم و هزینه‌های زیاد ساخت مدار چاپی نجات دادیم! تا با صرف کمترین هزینه و زمان قابل اتکاترین سخت‌افزار برای شروع کار و حتی تست میدانی(Field Test) در اختیار شما باشد.

با توجه به حذف پیچیدگی‌های طراحی سخت‌افزار، همه علاقه‌مندان به تکنولوژی با هر سطحی از دانش (از دانش آموزان گرفته تا دانشگاهیان و افراد حرفه‌ای) می‌توانند به راحتی از این کریر برد 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 برد مقدماتی ماژولار رزبری پیکو پرومیک" می‌توان قابلیت‌های زیادی از قبیل:

- بسترهای ارتباطی متنوع (مثل LoRa، WiFi، Ethernet،GSM و ...)
- سنسورهای متنوع (مثل دما، رطوبت، فشار، نور و ...)
- عملگرهای متنوع(مثل Relay، Servo Motor، DC Motor و ...) را به سرعت به برد Raspberry Pico افزود.

در طراحی "ProMake Raspberry Pico Basic Carrier" 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 فراهم شود. روی برد دو امکان تغذیه کریر برد در نظر گرفته شده است که از طریق خود ماژول پیکو و هم یک پورت USB-C بکه در پروژه های کاربردی امکان پیاده‌سازی حالتهای مختلف را فراهم می کند. البته در نسخه حرفه ای کریر برد پیکو امکان تغذیه آداپتوری 7 تا 24 ولت وجود دارد که برای استفاده در پروژه هایی مانند رهیاب خودرو با تغذیه 12 و 24 خودرو بسیار مناسب و کارا است. امکان دیگر وجود سیستم کلاک ساعت RTC برای کاربردهای خاص بهره‌گیری از زمان و ساعت دقیق بهمراه باتری پشتیبان در نسخه حرفه ای کریر پیکو است که قابلیت جذابی برای کارهایی مثل لاگ گیری می باشد.

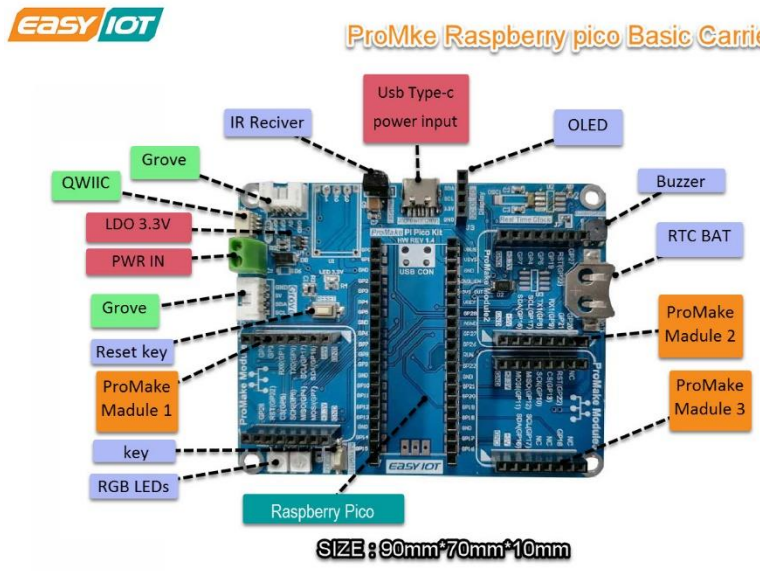
با استفاده از "کریر برد مقدماتی ماژولار رزبری پیکو پرومیک" می‌توانید ماژول Raspberry Pico خود را به یک برد توسعه اینترنت اشیاء تبدیل کنید و به راحتی یک Sensor Node یا یک ربات متصل به اینترنت بسازید و آن را از دور کنترل کنید.

مشخصات فنی

- پشتیبانی از انواع ماژول رزبری پیکو استاندارد
  - PICO
  - PICO H
  - PICO W
- امکان استفاده از سه ماژول ProMake به صورت هم‌زمان

- SPI، I2C، Serial و Analog برای ماژول اول
- SPI، I2C برای ماژول دوم
- Serial، I2C و Analog برای ماژول سوم

- دو عدد RGB LED برروی برد
- Buzzer برروی برد
- کلید فشاری
- کانکتورهای توسعه ای QWIIC و Grove
- پشتیبانی از صفحه نمایش I2C OLED
- یک عدد دیود گیرنده IR
- روش های تامین تغذیه :
  - کانکتور USB Type-C روی برد
  - کانکتور VIN روی برد(7 الی 24 ولت)
  - USB Port ماژول پیکو
- حداکثر جریان تغذیه 3.3V LDO روی برد: 600mA
- ابعاد: 90mm x 70mm x 10mm
- وزن بورد بدون ماژول: 32 گرم



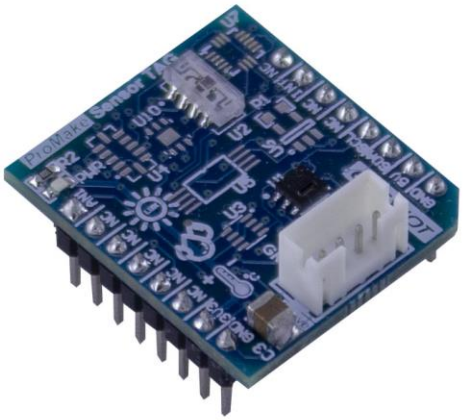
ماژول ProMake Sensor TAG

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 0 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 0 لوکس تا حدود 167K لوکس 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)
- ولتاژکاری: 3.3 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 کیت‌های 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, ... را دارد. بروی این ماژول سنسور MQ2بصورت 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 24 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 سریMQ
- دارای کانکتور جهت تعویض آسان سنسور
- دارای پتانسیومتر جهت کالیبراسیون
- دارای خروجی آنالوگ برای ارائه به برد توسعه میزبان
- دارای قطعه ADC با اینترفیسI2C
- دارای LED نشانگرPWR
- ولتاژکاری: 5ولت
- ابعاد: 25mm x 28mm x 30mm
- قابل استفاده برروی بردهای آموزشی(breadboard)

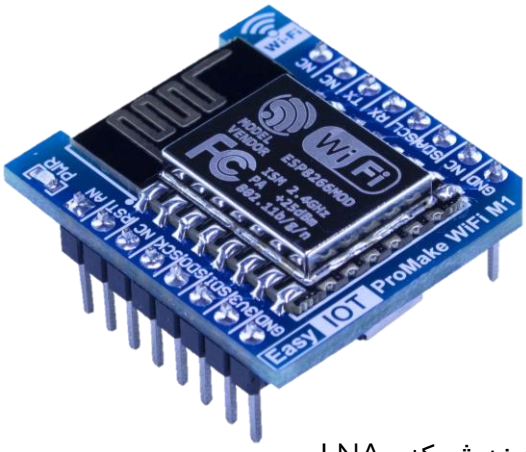
ماژول ProMake WiFi ESP12

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 3.3 ولت طراحی شده است و مصرف انرژی در حالت آماده به کار آن کمتر از 1.0 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: 3.3 ولت
- توان آماده بکار کمتر از 1 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 10 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 10 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 کیت‌های 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 ولتاژAC به رله، به برد دست نزنید.

مشخصات فنی

- امکان سوئیچ بارهای تا 250 VAC یا 30VDC
- حداکثر جریان قابل کنترل توسط رله تا A10
- دارای LED وضعیت رله و تغذیه ماژول
- ولتاژکنترل رله: 5 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)

ماژول رزبری پیکو Raspberry PICO

خانواده ماژولهای رزبری پیکو تا زمان نگارش این درسنامه دارای 4 عضو هستند که به لحاظ محتوای ارائه شده در این مستند تفاوتی در استفاده با کریر بوردهای پرومیک ایجاد نمی کنند و تنها سری PICO W بدلیل داشتن امکان وای فای داخلی مزیتهای بیشتری در نوع ارتباط و کاربری دارد. لذا اینجا ما به معرفی اجمالی سری اولیه PICO می پردازیم و سری پیشرفته از PICO W هم استفاده خواهد شد.

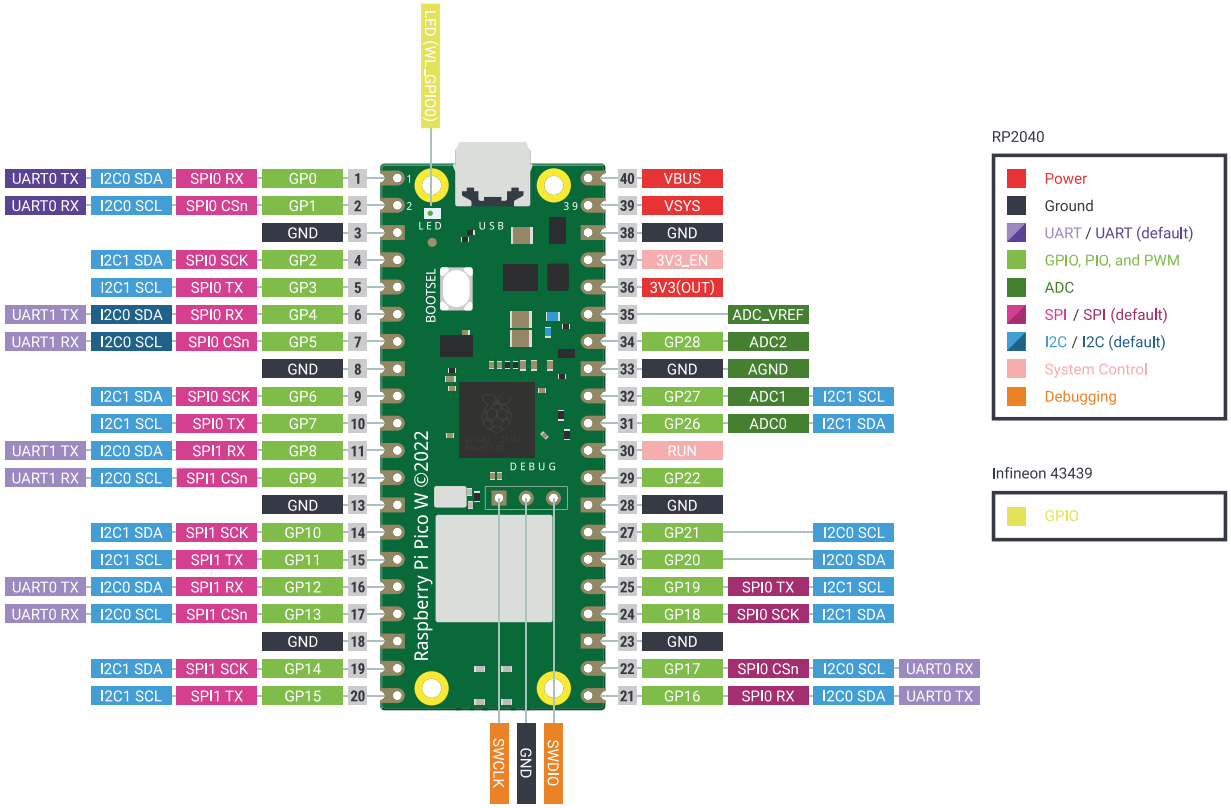
رزبری پیکو یک میکروکنترلر با قابلیتهای فراوان بر پایه هسته Dual-core Arm Cortex M0+ می باشد و دارای واسطهای دیجیتال متعددی هست که سبب انعطاف بالای آن شده است. برخی از ویژگیهای اصلی بصورت زیر می باشد:

- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-drop programming using mass storage over USB
- 26 x multi-function GPIO pins
- 2 x SPI, 2 x I2C, 2 x UART, 3 x 12-bit ADC, 16 x controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Accelerated floating-point libraries on-chip
- 8 x Programmable I/O (PIO) state machines for custom peripheral support

ماژول Pico از طریق یک سری پین در هر دو لبه خود با سخت افزارهای دیگر صحبت می کند. اکثر این پین ها به عنوان یک پایه ورودی/خروجی (GPIO) همه منظوره کار می کنند، به این معنی که می توان آنها را

طوری برنامه ریزی کرد که به عنوان یک ورودی یا یک خروجی عمل کنند. برخی از پینها ویژگیهای اضافه تری دارند و برای ارتباط با سخت افزارهای پیچیده تر استفاده می شوند و که در درسهای آتی بیشتر معرفی می شوند. یکسری از این پینها هم برای تغذیه پاور و عملکرد ثابتی دارند. در ضمن ماژول دارای 3 پین برای ورودی آنالوگ است که البته می توانند بصورت ورودی و خروجی دیجتال هم پیکربندی شوند.

40 پین Raspberry Pi Pico در شکل زیر برچسب گذاری شده اند و عملکردهای اضافی پین ها نیز اشاره شده است.



برنامه نویسی میکرو پایتون

محیط برنامه نویسی Thonny

زبان میکرو پایتون

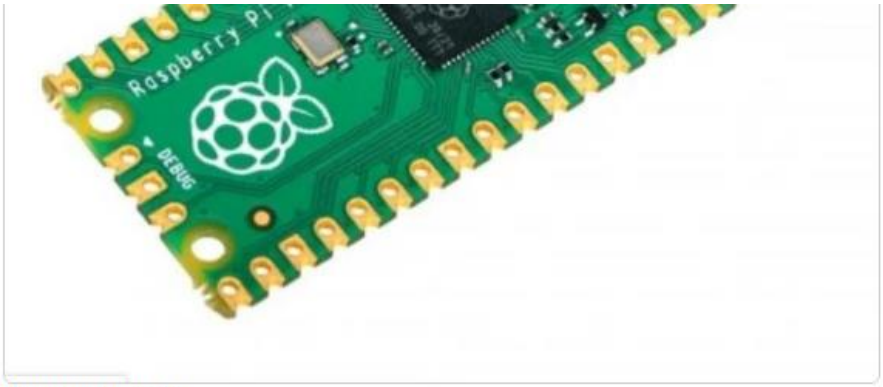
با توجه به محبوبیت جهانی زبان پایتون و گسترش استفاده آن در کاربردهای مختلف و سهولت و امکاناتی که ارائه می کند بهره گیری از آن در سخت افزارهایی مانند میکروکنترلرها نیز مورد توجه قرار گرفت.

به واقع زبان برنامه نویسی پایتون برای سیستمهایی کامپیوتری مانند دسکتاپ، لپ تاپ و سرور توسعه یافته است. بورد های میکروکنترلر مانند رزبری پیکو کوچکتر، ساده تر و حافظه بسیار کمتری دارند و این بدین معنی است که میکروکنترلرها نمی توانند همان زبان پایتون را مشابه کامپیوتر ها اجرا کنند.

ایجاست که میکروپایتون وارد می شود. در ابتدا توسط Damien George توسعه یافت و در سال 2014 اولین بار منتشر شد. میکروپایتون با زبان برنامه نویسی پایتون سازگار است و بطور اختصاصی برای میکروکنترلرها توسعه داده شده است. میکروپایتون دارای اکثر ویژگی های اصلی زبان پایتون است و افزون بر آن دارای ویژگیهای خاصی برای استفاده از میکروکنترلرهایی مانند رزبری پیکو و ESP8266 و سری های ESP32 و سایر بوردهای میکروکنترلر هست. اگر قبلا با پایتون برنامه نویسی کرده اید شما بسرعت با میکروپایتون آشنا خواهید شد. و اگر نه، نگران نباشید، میکروپایتون زبان دوستانه ای برای یادگیری است. برای برنامه نویسی میکرو پایتون از نرم افزار های IDE مختلفی می توان استفاده کرد اما در این درسنامه از نرم افزار Thonny استفاده می شود که در مستندات شرکت رزبری هم این ابزار استفاده شده است.

نصب نرم افزار میکروپایتون روی رزبری پیکو

رزبری پیکو بصورت کارخانه ای دارای Firmware میکرو پایتون نیست و برای شروع کار با آن لازم است نسخه مناسب میکروپایتون را از وبسات micropython.org دریافت کنید. دقت نمائید که نوع میکروکنترلر و سری آنرا بدرستی انتخاب نمائید. مطابق شکل زیر آخرین نسخه پایدار ارائه شده را دانلود نمائید.



Vendor: Raspberry Pi
Features: Breadboard friendly, Castellated Pads, Micro USB
Source on GitHub: rp2/PICO
More info: Website

Installation instructions

Flashing via UF2 bootloader

To get the board in bootloader mode ready for the firmware update, execute `machine.bootloader()` at the MicroPython REPL. Alternatively, hold down the BOOTSEL button while plugging the board into USB. The uf2 file below should then be copied to the USB mass storage device that appears. Once programming of the new firmware is complete the device will automatically reset and be ready for use.

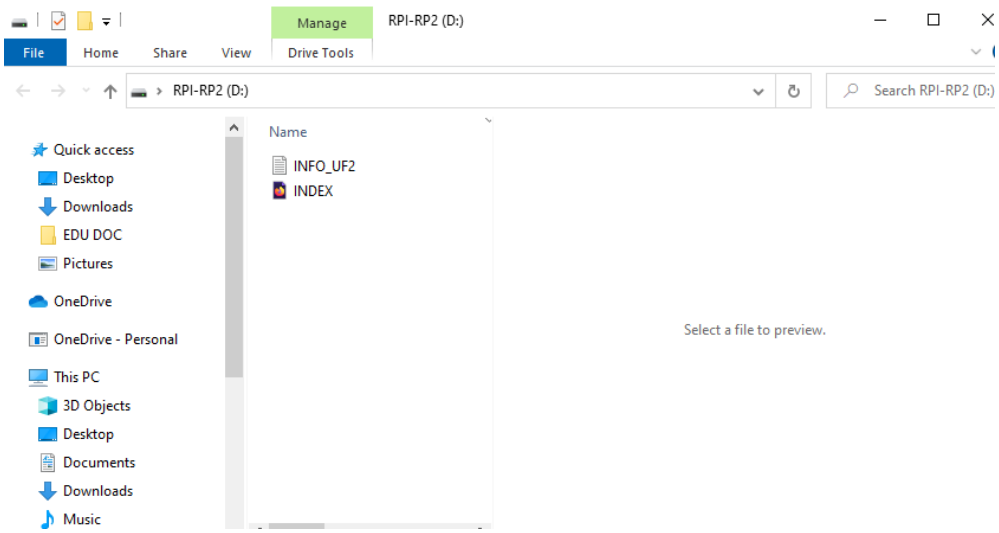
Firmware

Releases

v1.20.0 (2023-04-26) .uf2 [Release notes] (latest)
v1.19.1 (2022-06-18) .uf2 [Release notes]
v1.18 (2022-01-17) .uf2 [Release notes]
v1.17 (2021-09-02) .uf2 [Release notes]
v1.16 (2021-06-18) .uf2 [Release notes]

برای نصب نرم افزار میکروپایتون رزبری پیکو بسادگی مراحل زیر را انجام دهید:

1. کابل میکرو USB را به ماژول پیکو متصل نمائید.
2. دکمه BOOT SEL بالای ماژول را به پائین فشار دهید و همزمان سر دیگر کابل را به کامپیوتر متصل نمائید. دقت نمائید که هنگام اتصال کلید بوت فشرده باشد.
3. حدود 3 ثانیه کلید بوت را نگه دارید و بعد رها کنید. چند لحظه بعد پیکو بعنوان یک درایو جدا شدنی به سیستم اضافه می شود و بصورت اتوماتیک این درایو باز می شود. فایل میکروپایتون دانلود شده(دارای پسوند uf2) را به پنجره باز شده بیاندازید.
4. بعد از چند لحظه برنامه روی رزبری پیکو اجرا و پنجره بسته می شود و کامپیوتر رزبری پیکو را خواهد شناخت. در بخش بعدی نحوه اتصال به آن و شروع برنامه نویسی بیان می شود.

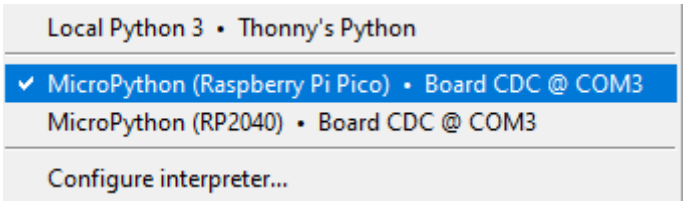


نصب نرم افزار Thonny در ویندوز

برای نصب نرم افزار Thonny می توانید به وبسایت شرکت مراجعه نمائید و متناسب با سیستم عامل کامپیوترتان آنرا دانلود و نصب نمائید.

محیط نرم افزار بشکل زیر می باشد:

1. منوی ابزار: شامل منوهای نرم افزار و آیکون های دسترسی سریع مانند ذخیره، اجرا و توقف برنامه است.
2. بخش دسترسی به فایلهای درون میکروکنترلر و کامپیوتر میزبان
3. محیط برنامه نویسی که بصورت یک فایل قابلیت ذخیره بروی میکروکنترلر و کامپیوتر میزبان را دارد
4. محیط پایتون شل که خروجی برنامه در حال اجرا را نمایش می دهد و همچنین می توان در این بخش نیز دستورات مستقلی را نوشت و اجرا نمود.
5. در این بخش به مفسر برنامه دسترسی دارید و امکان انتخاب ارتباط با میکروکنترلر متصل به کامپیوتر را دارد. از این بخش با انتخاب زبان میکرو پایتون و رزبری پیکو و پورت سریال نرم افزار به میکروکنترلر متصل می شود.



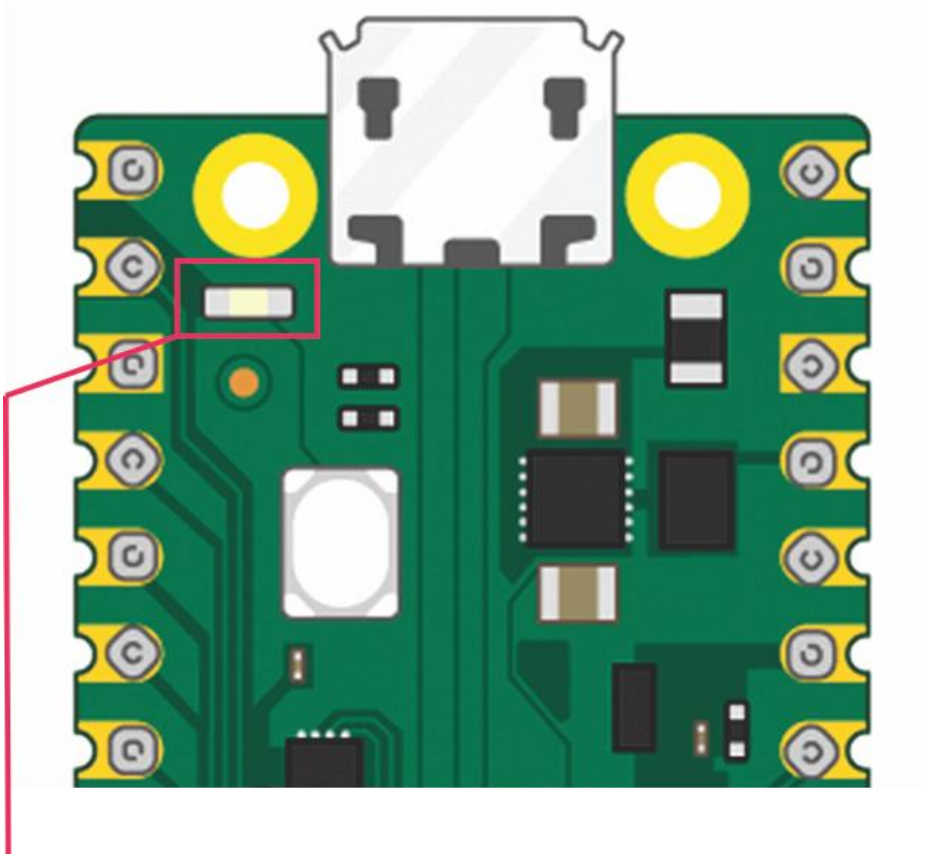
بخش اول: کنترل خروجی‌های دیجیتال

درس اول: چشمک زدن LED روی ماژول رزبری پیکو

معرفی:

در این درس قصد داریم با استفاده از سیگنال دیجیتال، LED روی برد ماژول پیکو را روشن و خاموش و یا به اصطلاح به حالت چشمک زدن درآوریم. با استفاده از اقلام اشاره شده در ادامه متن با ماژول رزبری پیکو ارتباط می گیریم و توسط پایه GP25 که به LED روی برد ماژول(On Board) متصل است، بصورت یک ثانیه درمیان روشن و خاموش می کنیم.

دقت نمائید که منظور از LED On Board المان روی خود ماژول پیکو هست و البته که روی بورد کریر پیکو هم RGB LED سه رنگ وجود دارد که درسهای آتی در مورد کنترل آنها صحبت خواهیم کرد.

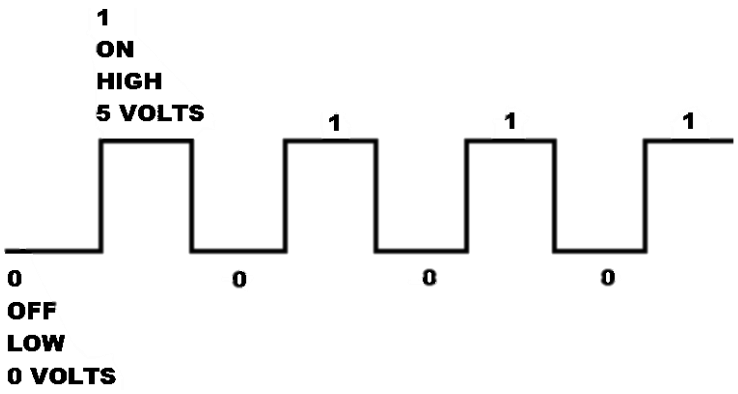


قطعه On Board LED سمت چپ کانکتور میکرو USB قرار دارد

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. در ماژول رزبری پیکو سطح سیگنال دیجیتال بین 0 تا 3.3 ولت می باشد.



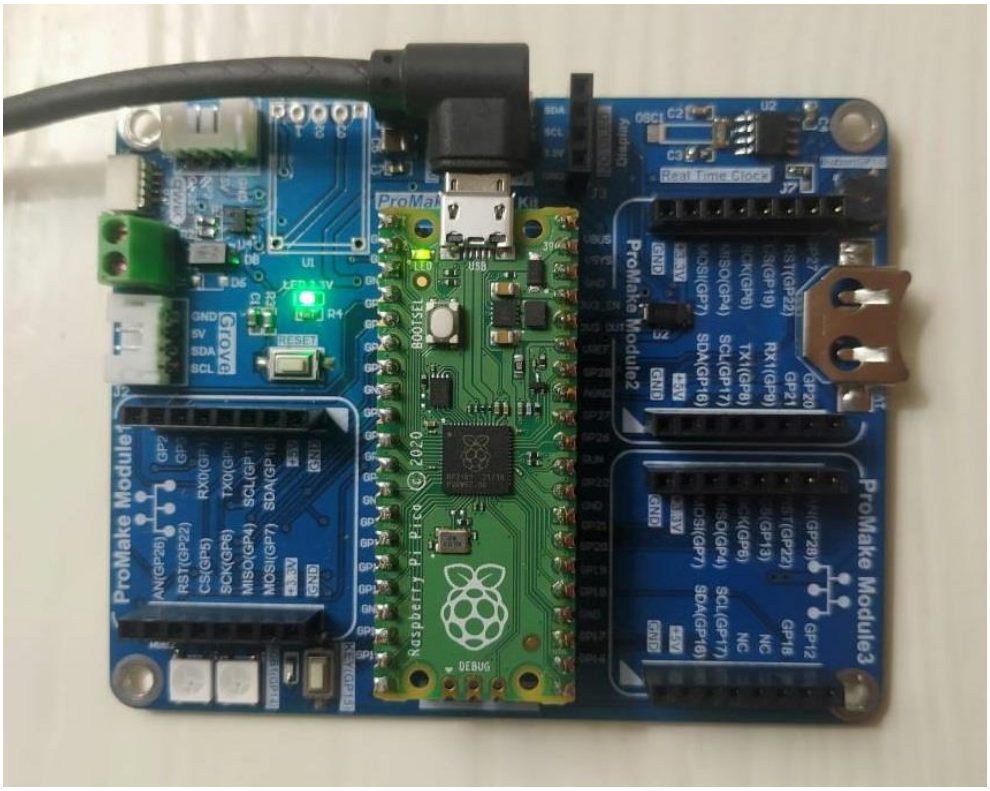
شکل 1: سیگنال دیجیتال باینری

اقلام مورد نیاز

ماژول رزبری پیکو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول رزبری پیکو را توسط کابل میکرو USB به کامپیوتر متصل کنید. توجه نمائید که همانطور که در مطلع درسنامه اشاره شد که کریر بورد رزبری پیکو از سه طریق امکان تغذیه پاور دارد. یکی از این راهها خروجی تغذیه 5 ولت ماژول رزبری پیکو است. البته دقت شود که میزان جریان دهی ماژول محدود است و برای برخی از ماژولهای پر مصرف مانند GSM حتما از تغذیه از طریق کانکتور USB Type-C و یا ورودی Vin استفاده نمائید.



کدنویسی و شرح کد

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

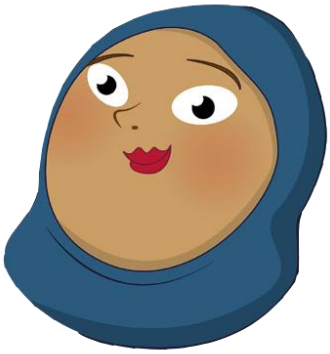
```
#RPI PICO ProMake Carrier Board
#OnBoard PICO LED Blink
#The LED will turn on for one second and then turn off for one second

from machine import Pin
from time import sleep

LED = Pin(25,Pin.OUT)

while True:
    LED.on()
    sleep(1)
    LED.off()
    sleep(1)
```

کتابخانه ها در میکرو پایتون

```
from machine import Pin
from time import sleep
```

این دو خط کوتاه کد کلیدی برای کار با میکرو پایتون در Pico است: با این دو خط ، مجموعه ای از کدها را که به عنوان کتابخانه شناخته می شود، در این مورد، کتابخانه ماشین به برنامه وارد یا اضافه می کند. کتابخانه ماشین شامل تمام دستورالعمل هایی است که میکرو پایتون برای برقراری ارتباط با Pico و سایر دستگاه‌های سازگار با میکروپایتون، نیاز دارد و البته برنامه نویسی را برای محاسبات فیزیکی و سخت افزاری گسترش می‌دهند.

بدون خط اول(کتابخانه ماشین)، هیچ یک از پین‌های GPIO Pico قابل کنترل نیستند و نمی توان LED روی برد را روشن و خاموش کرد.



وارد کردن بخشی از کتابخانه

در پایتون و میکروپایتون این امکان وجود دارد که بخشی از یک کتابخانه را بجای کل آن وارد برنامه کنید. این امکان کمک می کند که برنامه حافظه کمتری استفاده کند و البته توابع مختلف را از کتابخانه های مختلف ترکیب نمائید.

در این درسنامه در برخی برنامه ها کل کتابخانه و در جاهائی بخشی از کتابخانه مانند from machine import Pin که فقط تابع Pin را از کتابخانه ماشین وارد برنامه می کند.

تابع تعیین جهت پین‌ها

```
LED = Pin(25,Pin.OUT)
```

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 Pin انجام می شود. در آرگومان وضعیت می‌توان OUT برای خروجی بودن و INPUT برای ورودی بودن را قرار داد. همچنین اگر وضعیت Pin.PULL_UP 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، 1 منطقی باشد. در برنامه فوق کنترل روشن و خاموش بودن LED با تنظیم پین مربوطه به صورت خروجی انجام 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++;
}
```

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 1 جمع می‌شود و برابر با 1 می‌شود حال باز شرط i<=5 صحیح است و حلقه مجددا اجرا می‌شود تا جایی که i برابر 5 شود و با 1 جمع شده و برابر 6 شود حال دیگر شرط حلقه صحیح نیست و ادامه برنامه از آن خارج می‌شود.

تابع مقداردهی به پین‌ها

```
LED.on()
```

با استفاده از این تابع مقدار دهی سطح بالا برای یک پایه انجام می‌گیرد. همچنین می توان از تابع LED.value(1) برای تغییر سطح منطقی خروجی به یک پین استفاده کرد. به هر شکل با استفاده از تابع اشاره شده مقدار خروجی پین 25 به یک منطقی و 3.3 ولت تغییر می یابد و LED روشن می شود.

```
LED.off()
```

با استفاده از این تابع مقدار دهی سطح پائین برای یک پایه انجام می‌گیرد. همچنین می توان از تابع LED.value(0) برای تغییر سطح منطقی خروجی به صفر پین استفاده کرد. به هر شکل با استفاده از تابع اشاره شده مقدار خروجی پین 25 به صفر منطقی و حدود صفر ولت تغییر می یابد و LED خاموش می شود.

تابع تولید تاخیر

```
sleep(1)
```

این تابع برای ایجاد تاخیر زمانی استفاده می‌شود و پیکو 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 ثانیه 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.

چالش و تمرین

- بااستفاده از تابع Toggle برنامه را بازنویسی کنید.
- آیا می توان زمانهای تاخیر طولانی تری را برای روشن و خاموش کردن LED با برنامه ایجاد کرد؟
- کوتاهترین زمان تاخیری که روشن و خاموش شدن LED قابل رویت است را محاسبه کنید.
- تفاوت کتابخانه time و utime چیست؟

بخش دوم: خواندن ورودی‌های دیجیتال

درس دوم: روشن کردن LED با فشرده شدن کلید فشاری

معرفی:

در این درس برای اولین بار با کریر بورد پیکو ارتباط می گیریم و با اساتفاده از کلید فشاری روی برد کنترل روشن و خاموش کردن LED روی بورد پیکو را انجام می دهیم. برای این کار از کلید KEY گوشه پائین برد مطابق عکس زیر استفاده می کنیم که به پین GP15 پیکو متصل است. کلید مطابق اطلاعات سایت مستندات با یک مقامت روی برد به VCC متصل است و اصطلاحا Pull up شده است. دقت نمائید که یک کلید فشاری دیگر روی برد با نام RESET وجود دارد که برای ریست کردن بورد پیکو استفاده می شود و با کلید KEY اشتباه گرفته نشود.

چاپ روی بورد های پرومیک

برای سهولت استفاده و برنامه نویسی محصولات پرومیک روی همه بورد ها اطلاعات مهم چاپ شده است. برای مثال اینکه هر پین روی کریر بورد به کدام پین ماژول میکروکنترلر متصل شده است. در اینجا پین کلید فشاری KEY به پایه GP15 روی بورد رزبری متصل شده است.

شما می توانید سایر اتصالات کانکتورهای ماژولهای پرومیک را بررسی کنید و متوجه شوید به کدام پین GPIO بورد پیکو متصل شده است.

پیش نیاز

کلید فشاری امکان تولید دو حالت منطقی 1یا 0 را فراهم می کند. مقدار 0 یا 1 بستگی به نقشه بورد و نحوه اتصال کلید دارد. در اینجا با توجه به بلوک دیاگرام و نقشه کریر بورد در حالت غیر فشرده مقدار پایه GP15 یک و سطح ولتاژ3.3 است و در حالت فشرده شدن کلید مقدار پین به سطح صفر منطقی و ولتاژ صفر می رسد. 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.

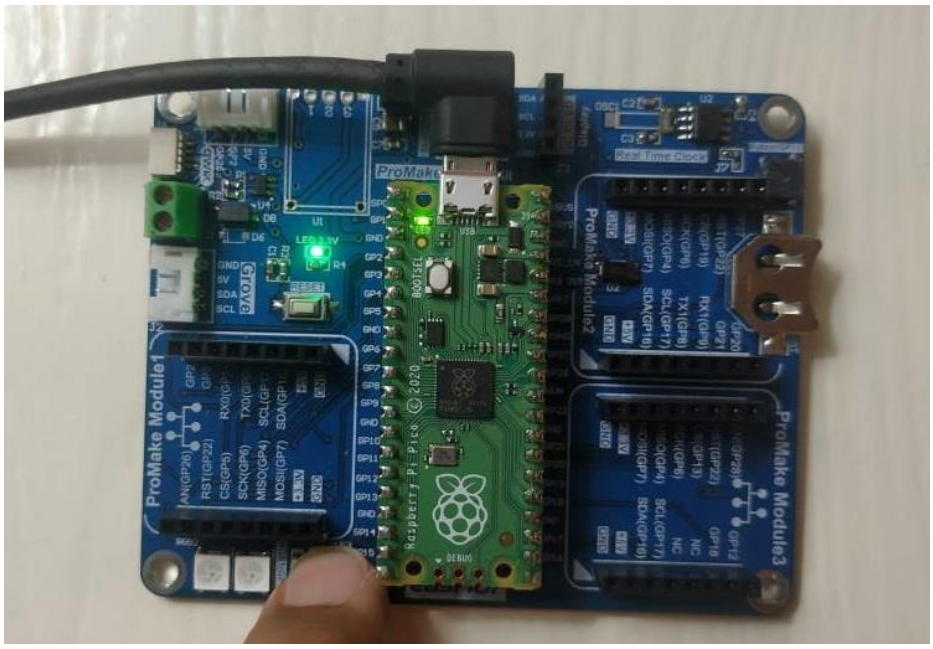
اقلام مورد نیاز

کیت رزبری پیکو ProMake(جهت استفاده از دکمه فشاری روی آن)

بورد رزبری پیکو + کابل USB میکرو

آماده‌سازی سخت افزار

ماژول رزبری پیکو 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 کیت منطبق نموده و ماژول پیکو را وارد کانکتور مادگی 40 پین نمایید. حال ماژول پیکو را توسط کابل میکرو USB به کامپیوتر متصل کنید.



کدنویسی و شرح کد

در محیط Thonny IDE کد زیر را وارد نمایید.

```
#RPI PICO ProMake Carrier Board
#LED Control By Key
#The LED will turn on and off by pressing KEY(GP15) on ProMake PICO Carrier

from machine import Pin
from utime import sleep

KEY = machine.Pin(15, machine.Pin.IN)
```

```
LED = machine.Pin(25,machine.Pin.OUT)

while True:

    if KEY.value() == 0: # KEY Pressed

        LED.on()

    else:

        LED.off()
```

مطابق توضیحات درس اول، برای استفاده از ماژول پیکو و دسترسی به پینهای GPIO از تابع Pin در کتابخانه ماشین استفاده می کنیم و همچنین برای ایجاد تاخیر زمانی نیز از تابع sleep در کتابخانه utime بهره می بریم.

تعریف LED و KEY

```
KEY = machine.Pin(15, machine.Pin.IN)
LED = machine.Pin(25,machine.Pin.OUT)
```

این کار دو متغیر به پینهای ماژول پیکو مرتبط می کند. دقت نمائید که چون کلید فشاری روی کریر بورد دارای مقاومت pull up هست نیاز نیست در برنامه و تعریف KEY آنرا pull up نمائیم. این یک مزیت استفاده از کریر بورد می باشد ولی اگر از بورد آموزشی استفاده می شد حتما نیاز بود یک مقاومت بیرونی و یا مقاومت درونی مازول پیکو را استفاده کنیم.

ساختارهای شرطی

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 (شرط1) { // در صورتی که جواب عبارت شرط صحیح یا به عبارتی 1 شود سراغ انجام دستورات ذیل آن می رود
        // انجام دستورات الف
}
elif (شرط2) {
        // 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 elif 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 عمگر های 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

برای کنترل LED لازم است که مقدار کلید خوانده شود و براساس آن شرط وضعیت روشن و یا خاموش بودن LED مشخص شود. برای خواندن مقدار کلید از تابع value استفاده می کنیم و این مقدار زا ملاک شرط قرار می دهیم. همانطور که قبلا گفتیم با توجه به طراحی مدار کریر بورد حالت پیش فرض کلید 1 منطقی است و در زمان فشردن کلید مقدارش به صفر تغییر می کند. لذا طبق دستور if زیر زمانی که کلید فشرده نشده LED خاموش و در حالت فشردن کلید LED روشن می شود.

```
if KEY.value() == 0: # KEY Pressed
    LED.on()
else:
    LED.off()
```

چالش و تمرین

- طوری برنامه را بازنویسی نمائید که در زمان روشن شدن LED وضعیت آن در Shell نمایش داده شود.
- چه تغییری در برنامه باید انجام شد که در زمان فشردن کلید LED خاموش و در حالت غیر فشردن روشن شود؟
- برنامه ای بنویسید که وقتی کلید فشرده شد 2 مرتبه LED چشمک زدن با فاصله زمانی 3 ثانیه درمیان عمل کند و سپس خاموش شود.

بخش سوم: تولید فرکانس و ولتاژهای مختلف در خروجی دیجیتال

درس سوم: کنترل شدت نور LED روی ماژول رزبری پیکو

معرفی

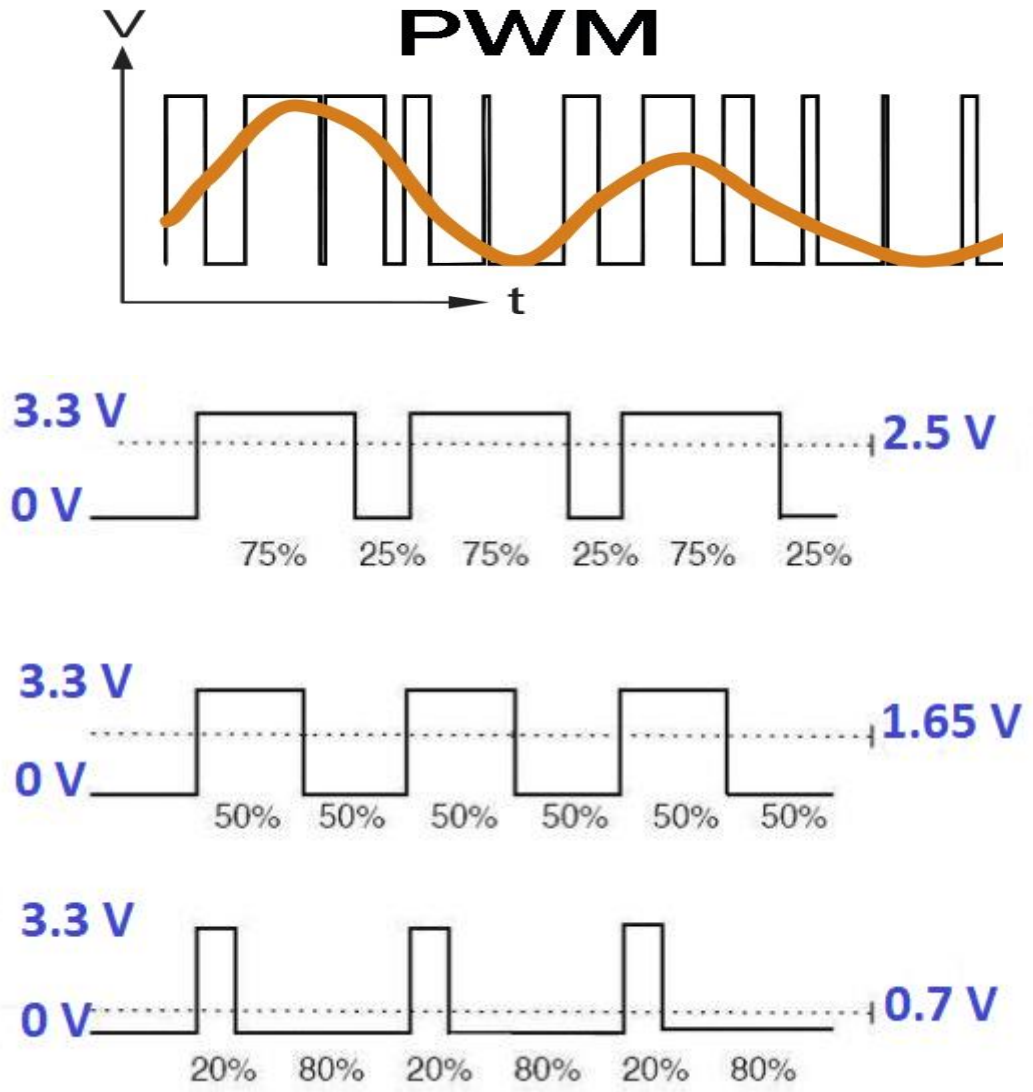
در این درس مفهوم سیگنال PWM را با کنترل میزان روشنایی LED روی برد ماژول پیکو تشریح می کنیم. میزان روشنایی LED به میزان جریان عبوری از آن بستگی دارد که در یک محدوده مشخص برای LED قابل تغییر است. تغییرات جریان نیز با میزان ولتاژ دوسر مقاومت سری (در حالت فعلی این مقاومت داخل ماژول پیکو قرار دارد) رابطه دارد و برای کنترل آن در دنیای آنالوگ بسادگی مقدار ولتاژ با المانهایی مثل پتانسیومتر قابل تغییر و کنترل است اما در دنیای دیجیتال که فقط صفر و یک هست چطور؟ برای این کار یک راه استفاده از یک و صفر های متوالی با دوره تناوب و زمانهای متفاوت است. به این ترتیب که زمانهای روشن، یک منطقی و یا در اینجا 3.3 ولت را تغییر دهیم تا به میزان مورد نظر جریان مقاومت و متناظر با آن LED تغییر کند. در ادامه به مفهوم سیگنال PWM می پردازیم تا موضوع روشن شود.

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 پایه‌ها `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 3.3 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 3.3ولت ON و در صفر ولت OFF باشد ایجاد کنیم، برای این کار می‌توان یک PWM که برای60 درصد زمان‌ها خروجی 3.3 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 3.3V * 0.6 = 2V



سه پارامتر اصلی PWM به صورت زیر است.

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 در مدت زمان مشخص باشد.

همه پین های GPIO تراشه سری RP2024 و از جمله ماژول پیکو قابلیت انتصاب بعنوان کانال PWM دارند و لذا مطابق جدول زیر براحتی می توان از پینهای مختلف استفاده کرد.

All 30 GPIO pins on RP2040 can be used for PWM:

GPIO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PWM Channel	0A	0B	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	6A	6B	7A	7B
GPIO	16	17	18	19	20	21	22	23	24	25	26	27	28	29		
PWM Channel	0A	0B	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	6A	6B		

اقلام مورد نیاز

ماژول رزبری پیکو + کابل USB مناسب

آماده‌سازی سخت افزار

ماژول رزبری پیکو را توسط کابل USB به کامپیوتر متصل می‌کنیم.

کدنویسی و شرح کد

ایجاد سیگنال PWM

در میکرو پایتون برای تولید یک سیگنال PWM نیاز به انجام سه کار زیر است:

1- وارد کردن تابع PWM از کتابخانه ماشین

```
from machine import PWM
```

2- انتصاب یک متغیر به تابع(کلاس) PWM توسط تابع PIN

```
LED = PWM(Pin(25))
```

3- انتصاب فرکانس به کانال PWM ایجاد شده

```
LED.freq(1000)
```

در این ایجا برای فرکانس مقدار 1000 هرتز پیشنهاد می شود که میزان روشن و خاموش شدن در واحد زمان می باشد.

4- تنظیم پارامتر Duty Cycle

```
LED.duty_u16(duty)
```

با توجه به اینکه در میکروپایتون ماژول رزبری پیکو برای تنظیم Duty Cycle از یک متغیر 16بیتی استفاده می شود که مقدار 0 تا 65535 را شامل می شود. واضح است که مقدار 0 در این مثال LED را روشن نمی کند و مقدار 100 با حدکثر روشنائی روشن خواهد شد و مقادیر مابین سطح روشنایی بین حداقل و حداکثر را خواهند داشت.

حلقه تکرار For

برای کنترل روشنایی LED و تغییر میزانDuty Cycle نیاز به یک حلقه تکرار شونده داریم تا میزان روشنایی با استفاده از یک تاخیر مناسب طوری تغییر دهد که روشنایی LED بطور پیوسته و به آرامی از صفر به حداکثر مقدار و روشن کامل و از حداکثر مقدار مجددا به صفر و خاموش تغییر نماید. برای این کار از حلقه For استفاده می کنیم .

در پایتون برای تکرارهای متناهی و مشخص از حلقه For استفاده می کنند. تعریف حلقه به شکلهای مختلفی استفاده می شود. در اینجا یک روش ساده استفاده از تابع range می باشد که بسادگی یک حلقه را ایجاد می کنیم. شکل ساده حلقه For بشکل زیر است. در این ساختار var متغیر تکرار شوند و در بازه تعریفی تکرار <iterable> است و قسمت دستورات در هر تکرار در بخش <statement(s)> قرار می گیرد.

```
for <var> in <iterable>:
    <statement(s)>
```

در اینجا برای تغییر Duty Cycle در بازه 16 بیتی عنوان شده بالا بشکل زیر از تابع range استفاده می کنیم. برای مثال `for duty in range(65535)` بدین معنا است که متغیر duty مقداری بین 0 تا 65535 در هر تکرار با افزایش 1 واحد خواهد داشت و این حلقه 65356 مرتبه تکرار خواهد شد.

در بخش دوم برای خاموش کردن از `for duty in range(65535, 0, -1)` استفاده شده است که برعکس حالت قبل از مقدار 65353 بطور معکوس تا 0 یک واحد کاهش می یابد.

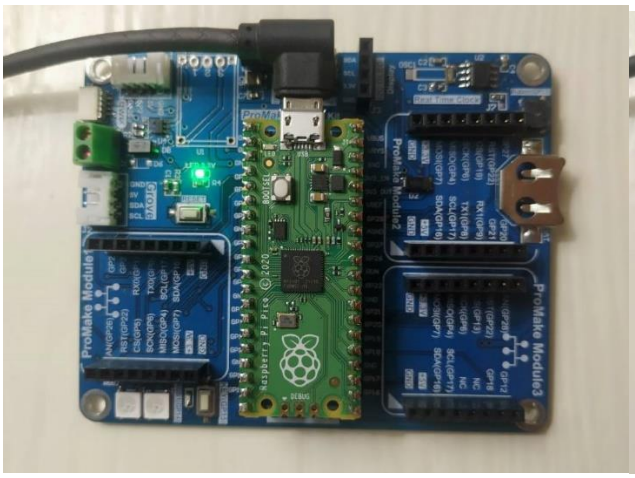
چرا 65535؟

عدد "65535" در نگاه اول عجیب به نظر می رسد - چرا این و چرا به سادگی اعداد 0-100 نیست؟ پاسخ به این واقعیت مربوط می شود به ماژول پیکو که با سیستم اعداد باینری کار می کند، جایی که تنها مقادیر ممکن برای یک رقم، 0 یا 1 است. یک عدد باینری 16 بیتی ساخته می شود از 16 رقم و حداکثر مقدار ممکن 16 رقم یک است پس: 1111111111111111.

اگر آن را دوباره به اعداد، سیستم شمارش 0 تا 9 که انسان ها استفاده می کنند، تبدیل کنید 65535 دریافت می کنید.

با توجه به توضیحات بالا کد کامل این بخش بشکل زیر می باشد و آنرا در Thonny IDE بنویسید و اجرا نمائید.

```
#RPI PICO ProMake Carrier Board
#Control LED brightness with PWM


from machine import Pin, PWM
from utime import sleep

LED = PWM(Pin(25))
LED.freq(1000)

while True:

    for duty in range(65535):
        LED.duty_u16(duty)
        sleep(0.0003)

    for duty in range(65535, 0, -1):
        LED.duty_u16(duty)
        sleep(0.0003)
    sleep(05)
```



چالش و تمرین

- طوری برنامه را بازنویسی نمائید که زمان روشن شدن دو برابر زمان خاموش شدن باشد.
- چه تغییری در برنامه باید انجام شد که در زمان فشردن کلید LED به حالت چشمک زدن بشود و در در حالت غیر فشرده مجدد شدت نور مشابه برنامه ارائه شده عمل کند.
- برنامه ای بنویسید که وقتی کلید فشرده شد برنامه کنترل روشنایی اجرا شود و با فشار مجدد متوقف شود و به همین ترتیب مجدد با فشردن کلید برنامه اجرا شود.

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

معرفی

در بسیاری از دستگاهها و سیستم های کامپیوتری برای اعلان یک هشدار و یا خطر از آلارم صوتی استفاده می شود لذا در طراحی سخت افزار المانی برای اعلام و پخش صدا قرار می دهند. بازر یکی از انواع المانهای تولید صوت هست که به دلیل سادگی و قیمت ارزان در بیشتر موارد از آن استفاده می شود. حتی در مادر بورد های کامپیوتر بازر وجود دارد و هنگام بوت شدن سیستم صدای بوق چک آن شنیده می شود.

در این درس قصد داریم که نحوه کارکردن با بازر را فرا بگیریم. برای این کار از بورد کریر رزبری پیکو استفاده می کنیم که دارای یک بازر است که به پین GP11 طبق چاپ روی بورد متصل است. برای این کار نوت های موسیقیایی را تعریف می کنیم و سپس با ارسال به بازر، یک موسیقی کوتاه که می تواند نقش یک پیام اعلان باشد را پخش می کنیم.

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 [1] و بازرهای منفعل [2]. بازرهای فعال بازرهایی هستند که دارای نوسانگر [3] 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.

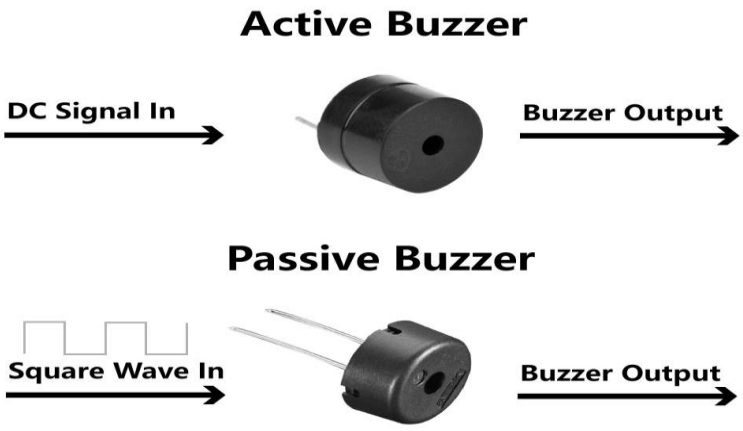
بازر استفاده شده در کیت ProMake از نوع Passive یا منفعل می‌باشد. سیگنال موردنیاز آن نیز مطابق توضیحات درس قبل با استفاده از PWM تولید می‌شود.

---

[1] Active Buzzer

[2] Passive Buzzer

[3] Oscillator

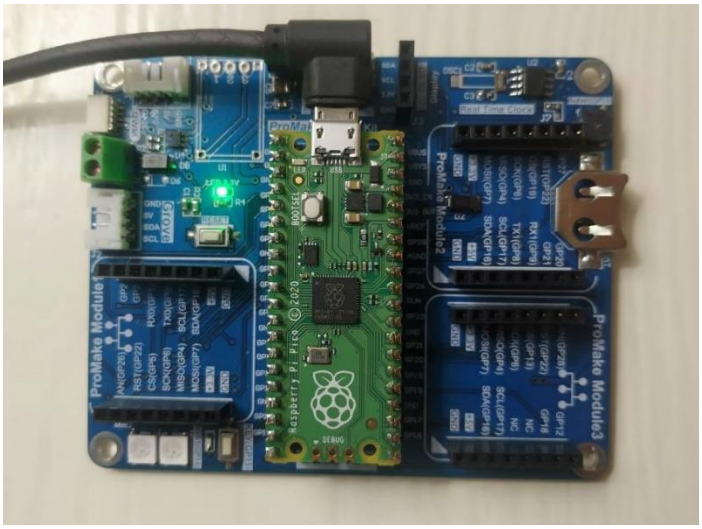

Active Buzzer
DC Signal In
Buzzer Output
Passive Buzzer
Square Wave In
Buzzer Output

اقلام مورد نیاز

- کیت رزبری پیکو ProMake(برای استفاده از بازر روی آن)
- ماژول رزبری پیکو + کابل USB میکرو

آماده‌سازی سخت افزار

ماژول رزبری پیکو 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 Thonny IDE کدی با ساختار زیر نوشته و اجرا می‌کنیم.

```
#RPI PICO ProMake Carrier Board
#Playing Music with a Buzzer on Raspberry Pi Pico with ProMake Carrier Board


from machine import Pin , PWM
from utime import sleep

buzzer = PWM(Pin(11))


tones = { "B0": 31,"C1": 33,"CS1": 35,"D1": 37,"DS1": 39,"E1": 41,"F1":
44,"FS1": 46,"G1": 49,"GS1": 52,"A1": 55,"AS1": 58,
          "B1": 62,"C2": 65,"CS2": 69,"D2": 73,"DS2": 78,"E2": 82,"F2":
87,"FS2": 93,"G2": 98,"GS2": 104,"A2": 110,"AS2": 117,
          "B2": 123,"C3": 131,"CS3": 139,"D3": 147,"DS3": 156,"E3": 165,"F3":
175,"FS3": 185,"G3": 196,"GS3": 208,"A3": 220,
          "AS3": 233,"B3": 247,"C4": 262,"CS4": 277,"D4": 294,"DS4": 311,"E4":
330,"F4": 349,"FS4": 370,"G4": 392,"GS4": 415,
```

```
        "A4": 440,"AS4": 466,"B4": 494,"C5": 523,"CS5": 554,"D5": 587,"DS5":
622,"E5": 659,"F5": 698,"FS5": 740,"G5": 784,
        "GS5": 831,"A5": 880,"AS5": 932,"B5": 988,"C6": 1047,"CS6":
1109,"D6": 1175,"DS6": 1245,"E6": 1319,"F6": 1397,
        "FS6": 1480,"G6": 1568,"GS6": 1661,"A6": 1760,"AS6": 1865,"B6":
1976,"C7": 2093,"CS7": 2217,"D7": 2349,"DS7": 2489,
        "E7": 2637,"F7": 2794,"FS7": 2960,"G7": 3136,"GS7": 3322,"A7":
3520,"AS7": 3729,"B7": 3951,"C8": 4186,"CS8": 4435,
        "D8": 4699,"DS8": 4978 }

song =
["E5","G5","A5","P","E5","G5","B5","A5","P","E5","G5","A5","P","G5","E5"]

def playtone(frequency):
    buzzer.duty_u16(1000)
    buzzer.freq(frequency)

def bequiet():
    buzzer.duty_u16(0)

def playsong(mysong):
    for i in range(len(mysong)):
        if (mysong[i] == "P"):
            bequiet()
        else:
            playtone(tones[mysong[i]])
        sleep(0.3)
    bequiet()
playsong(song)
```

این برنامه بازر را به تولید صدای بوق غیر ممتد وا می‌دارد که بخشی از یک موسیقی است. اگر نوتهای موسیقی را می شناسید می توانید آهنگ مورد علاقه خودتان را بسازید!

در بخش اول مطابق درسهای قبل کتابخانه ها و توابع مورد نیاز را وارد برنامه می کنیم. حال برای نواخت نوت و به صدا درآوردن بازر نیاز به اتصال یک کانال PWM به پایه ورودی بازر هست که طبق بلوک دیاگرام کریر بورد این پایه به GP11 متصل است. پس PWM را به این پین انتصاب می دهیم.

در بخش بعدی نوتهای موسیقیایی را تعریف می کنیم که درواقع فرکانسهای متفاوتی هست که بصورت ثابت عددی تعریف می شوند. در بخش بعدی با استفاده از نوتهایی که تعریف شده یک تکه آهنگ با توالی مشخص مانند سازنده آهنگ می سازیم! شما می توانید آهنگ خودتان را با استفاده از برخی ابزارهای آنلاین در اینترنت جستجو و ملودی مرتبط را بسازید.

از سه تابع برای نواخت ملودی استفاده شد است. در ابتدا تابع پخش را می نویسیم. در بخش song با تعریف حرف P سکوت ایجاد می کنیم. مشابه شرط if در صورت رسید به P تابع `bequiet()` را برای سکوت تعریف می شود.در بین نواخت هر نوت از 300 میلی ثانیه استفاده می کنیم.

```
def playsong(mysong):
    for i in range(len(mysong)):
        if (mysong[i] == "P"):
            bequiet()
        else:
            playtone(tones[mysong[i]])
        sleep(0.3)
    bequiet()
```

چالش و تمرین

- طوری برنامه را بازنویسی نمائید که ملودی تولدت مبارک پخش شود.
- چه تغییری در برنامه باید انجام شد که در زمان فشردن کلید ملودی پخش شود و در حالت غیر فشرده LED روی ماژول پیکو چشمک زن شود.
- برنامه ای بنویسید که همزمان با تولید ملودی LED روشن و سپس خاموش شود.
- آیا امکان افزایش سطح صدای ملودی در برنامه وجود دارد؟ سعی کنید برنامه ای بنویسید که متناسب با نوت ها سطح صدا افزایش و در انتهای ملودی صدا کاهش یابد.

درس پنجم: کنترل RGB LEDهای روی کیت

معرفی

یکی از امکانات روی کریر بورد وجود دو عدد LED از نوع RGB است. در واقع این قطعه دارای سه LED با رنگهای اصلی قرمز، آبی و سبز است و با ترکیب آنها می توان رنگهای مختلفی را ایجاد کرد. در خیلی از کاربردها LED تک رنگ برای نمایش و حالتهای مختلف دستگاه و اعلان وضعیت کافی نیست و جایی که استفاده از چند LED با رنگهای مختلف هم مشکل مکان وفضا داریم و اختصاص چند پین به LEDها امکان پذیر نیست.

در چنین حالاتی RGB LED ها بسیار منطقی و راهگشا هستند. در این درس با قطعه WS2812 آشنا می شویم و آنرا با رنگ و روشنایی مختلف روشن می کنیم. قطعه RGB1 از طریق پین GP14 به رزبری پیکو متصل است و نیز RGB2 نیز به RGB1 متصل است.

پیش نیاز

معرفی WS2812B RGB LED

یکی از امکانات روی کریر بورد وجود دو عدد LED از نوع RGB است که دارای شماره فنی WS2812 است. این قطعه با سایز 5050 بصورت SMD در کنار اسلات یک قرار دارد. دو قطعه LED با نام RGB1,RGB2 بطور سری بهم متصل شده اند.

رنگ LED توسط تراشه داخلی با دریافت فرمان دیجیتال از میکروکنترلر کنترل می شود. تعداد زیادی از این ال ای دی ها را می توان پشت هم قرار داد و تنها با 1 پین آنها را به صورت دیجیتال کنترل کرد. حداکثر 1024 LED را می توان بصورت سری بهم متصل نمود و 30 فریم در ثانیه آنها را رفرش کرد. (30 رفرش در ثانیه)

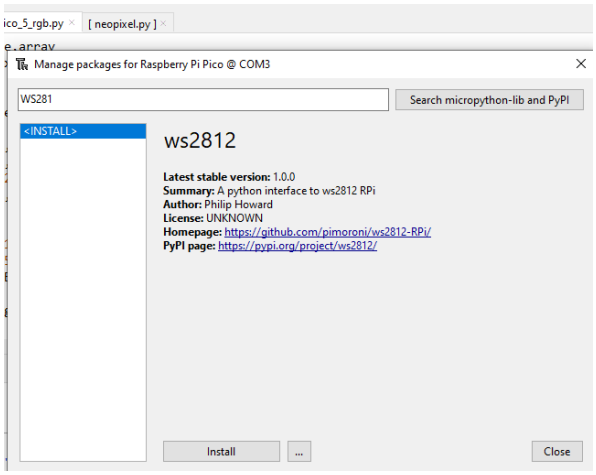
LED از نوع RGB است که هر رنگ را می توان با 8 بیت کنترل کرد. در نتیجه، هر LED می تواند 16777216 نوع رنگ را نشان دهد. (24 بیت)

نصب کتابخانه

کتابخانه‌های میکروپایتون 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. درواقع کتابخانه ها و درایورهای سخت افزاری براساس مشخصات سخت افزارها و اطلاعات فنی آنها و براساس پروتکلهای ارتباطی مختلف کنترل تا سطح ریجیسترها را برعهده می گیرند. حال برای کار با سخت

افزار در کد اصلی صرفا با فراخوانی کتابخانه ها بدون درگیر شدن با کدهای سطح پائین و با سهولت سخت افزارها را کنترل می کنیم.

برای کار با RGB LED های روی کریر بورد از کتابخانه WS2812 می‌توان استفاده نمود. برای نصب این کتابخانه مشهور می‌توان در بخش Tools –>Manage packages نام کتابخانه WS2812 را جستجو و نصب نمود.



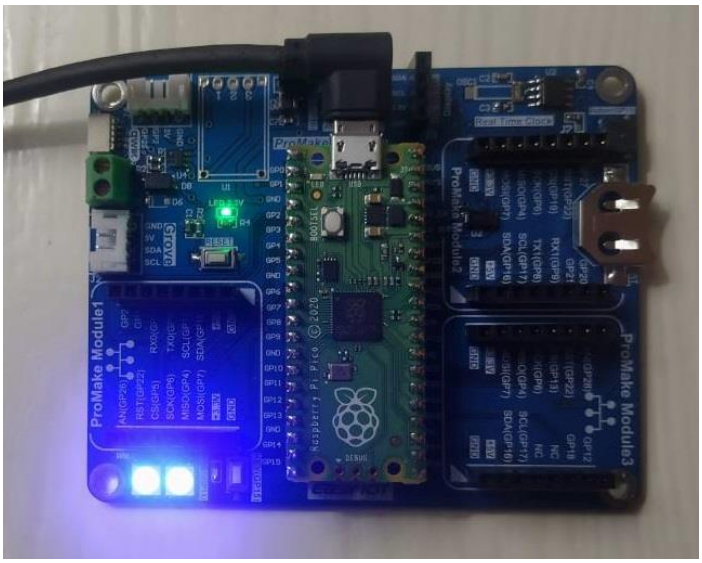
یک راه دیگر برای استفاده از کتابخانه RGB وارد کردن مستقیم آن در مسیر اجرای برنامه است. در این درس درایور را مستقیم وارد برنامه می کنیم. برنامه neopixel.py را از صفحه محصول کیت مقدماتی می توانید در کنار سایر برنامه های موجود در این درسنامه دانلود نمائید.

اقلام مورد نیاز

- کریر رزبری پیکو ProMake(برای استفاده از RGB LED های روی آن)
- ماژول رزبری پیکو + کابل میکرو USB

آماده‌سازی سخت افزار

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



کدنویسی و شرح کد

ابتدا از کتابخانه neopixel تابع Neopixel را به برنامه اضافه می کنیم. برای تعریف تعداد و شماره پین متصل به RGB از دستورات زیر استفاده می کنیم. در خط اول تعداد LED را مشخص می کنیم و در خط دوم شماره پین متصل به رزبری پیکو را مشخص می کنیم.

```
LED_NUM = 2
LED_PIN = 14
```

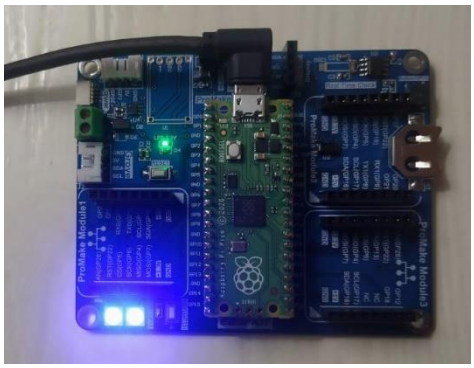
در ادامه تابع Neopixel را با تعداد LED و شماره پین و "GRB" مد تعریف رنگ را تنظیم می کنیم.

```
pixels = Neopixel(LED_NUM , LED_PIN , "GRB")
```

در ادامه برنامه رنگها تعریف می کنیم و با استفاده از حلقه While و دو حلقه For ابتدا هر دو led را با هم طبق تعریف لیست COLOR روشن می کنیم و سپس یک در میان با حلقه دوم For دو LED را متفاوت روشن می کنیم.

```
#RPI PICO ProMake Carrier Board
#Control WS2812B RGB LED

import time
from neopixel import Neopixel

LED_NUM = 2
LED_PIN = 14

pixels = Neopixel(LED_NUM , LED_PIN , "GRB")
```

```
BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)
WHITE = (255, 255, 255)

COLOR = [BLACK , RED , YELLOW , GREEN , CYAN , BLUE , PURPLE , WHITE]
pixels.brightness(100)


while True:

    for color in COLOR:

        pixels.set_pixel(0, color)
        pixels.set_pixel(1, color)
        pixels.show()
        time.sleep(0.2)

    for i in range(len(COLOR)-1):

        pixels.set_pixel(0, COLOR[i])
        pixels.set_pixel(1, COLOR[i+1])
        pixels.show()
        time.sleep(0.2)
```



چالش و تمرین:

- طوری برنامه را بازنویسی نمائید که میزان روشنایی در یک حلقه بطور پیوسته افزایش یابد.
- برنامه ای بنویسید که بصورت تصادفی رنگ LED ها را تغییر دهد.
- برنامه را طوری بازنویسی نمائید که رنگهای رنگین کمان را بترتیب نمایش یابد.
- با استفاده از کلید فشاری برنامه را طوری بازنویسی نمائید که در زمان فشردن کلید رنگ RGB بترتیب زرد و آبی روشن شود ملودی بازر پخش شود و با رها کردن کلید برنامه اصلی اجرا شود.

بخش چهارم: دریافت سیگنال از ورودی آنالوگ

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

معرفی

یکی از ماژولهای پرومیک موجود در این کیت ماژول سنسور گاز سری MQ هست که در مطلع درسنامه در مورد آن توضیح داده شد. اساسا نشت گازهای سمی، قابل اشتعال در محیط می تواند بسیار خطرناک است و تشخیص و واکنش بموقع می تواند جلوی آسیب و خطرات آتی را بگیرد.

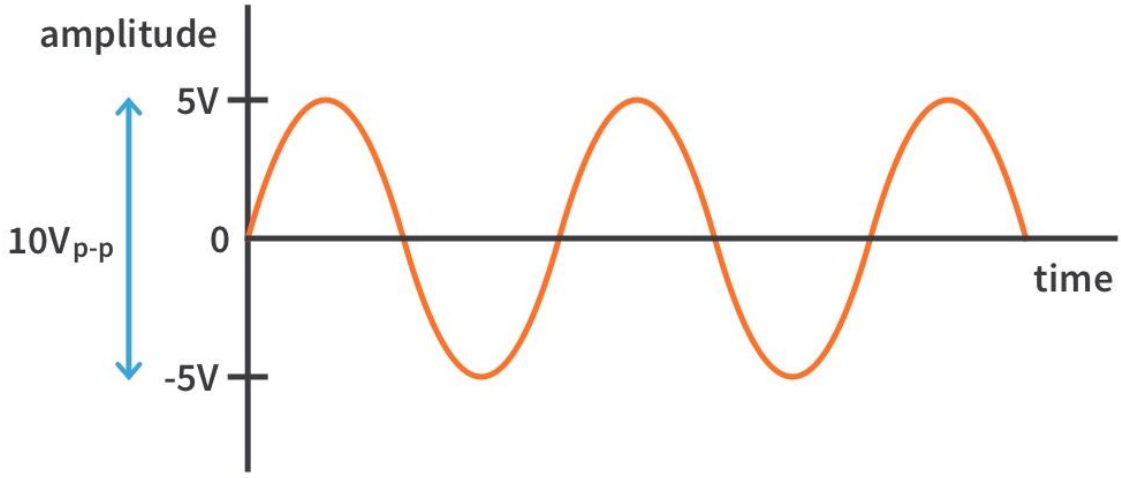
در این با استفاده از ماژول گاز MQ و پراب گاز MQ2 سعی می کنیم گازهای قابل اشتعال را کریر بورد رزبری پیکو تشخیص دهیم. خروجی سنسور سیگنال آنالوگ است. برای دریافت آن در مازول رزبری پیکو می بایست از یکی از پینهای ورودی آنالوگ استفاده کنیم.

در بخش ابتدایی توضیح داده شد که ماژول رزبری پیکو دارای 3 پین برای ورودی آنالوگ و ADC می باشد. این پینها بترتیب GP26,GP27 و GP28 هستند. در طراحی کریر بورد رزبری پیکو پرومیک برای هر سه اسلات 1و2و3 یک ورودی آنالوگ ADC در نظر گرفته شده است. اسلات یک به پین GP26، اسلات دو به GP27و اسلات سوم به GP28 متصل است. لذا با طراحی خوب کریر بورد براحتی و بدون محدودیت ماژول گاز MQ در هر 3 اسلات قابل نصب است. ما در اینجا از اسلات شماره یک استفاده می کنیم. در ادامه بحث قدری به معرفی سیگنال آنالوگ می پردازیم.

پیش نیاز:

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 هر کدام 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 سنسور ها، 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.

میکروکنترلز رزبری پیکو مانند هر دستگاه دیجیتال دیگری شامل هزاران ترانزیستور و سوئیچ های کوچک داخلی هست که صرفا دارای سطح روشن و خاموش هستند و لذا ماژول پیکو برای درک سیگنال پیوسته آنالوگ راهکاری مستقیم ندارد و باید ابتدا سیگنال آنالوگ با مداری به دیجیتال تبدیل شود. این تبدیل را مدار ADC انجام می دهد که مخفف Analog to Digital Convertor است و دارای دقت و نکات فنی زیادی است. همانطور که از نامش پیداست این مبدل سیگنال آنالوگ را به سیگنال دیجیتال تبدیل می کند. یکی از ویژگیهای مهم تبدیل ADC دقت بیتهای دیجیتال تبدیل است.

رزبری پیکو با چیپ داخلی RP2024 دارای دقت 12 بیتی برای هر 4 کانال ADC خود است. این بدین معنی است که سیگنال آنالوگ ورودی به اعداد دیجیتال رنج 0 تا 4095 تبدیل می شود. زبان میکروپایتون در ساختار داخلی خود آنرا به دو بایت و 16 بیت تبدیل می کند (دقت نمائید که دقت و کیفیت سیگنال با تبدیل 12 به 16 بیت تغییر نمی کند) و در روال برنامه نویسی از `read_u16()` استفاده می کنیم.

دقت نمائید که کانال چهارم آنالوگ اشاره در رزبری پیکو به سنسور آنالوگ دمای داخلی متصل است که در درسهای آتی به آن اشاره خواهیم کرد.

آشنایی با ساختار 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.

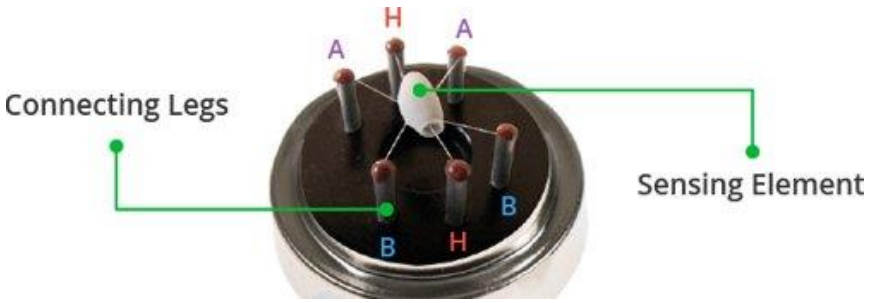
قابلیت‌های سنسورهای گاز MQ

- رنج تشخیص: 10تا 1000 ppm
- ولتاژکاری: 5 ولت
- خروجی آنالوگ: 0 تا 5 ولت
- حساسیت بالا برای تشخیص گاز
- پروب‌های قابل استفاده برروی ماژول ProMake
  - MQ-2: گازهای قابل اشتعال 300 تا 10000 ppm
  - MQ-3: گاز الکل 25 تا 500 ppm
  - MQ-4: گاز، متان 300 تا 10000 ppm
  - MQ-5: گاز، متان، گاز زغال سنگ 300 تا 5000 ppm
  - MQ-6: گاز LPG، ایزوبوتان، پروپان، 100 تا 10000 ppm
  - MQ-7: مونوکسید کربن CO 10 تا 1000 ppm

ساختار سنسورهای MQ

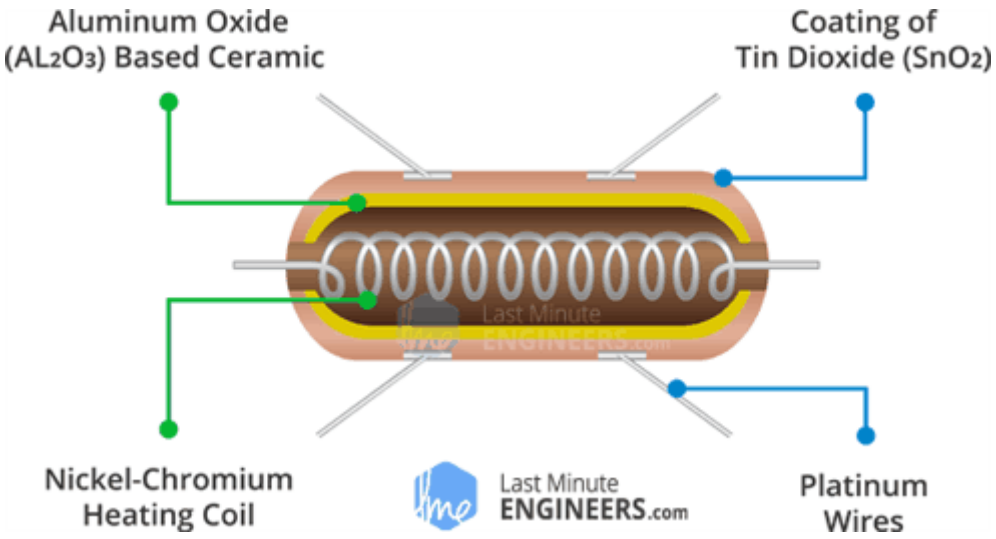
سنسورهای گاز سری MQ از هیتر داخلی کوچک استفاده می‌کنند لذا با دولایه توری از فولاد ضد زنگ که "شبکه ضد انفجار" نامید می شود، پوشانده شده است. بدین ترتیب می‌توان اطمینان حاصل نمود که حرارت داخلی پروب منجر به بروز انفجار گازهای قابل اشتعالی که قصد شناسایی آنها را داریم نمی شود.

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.



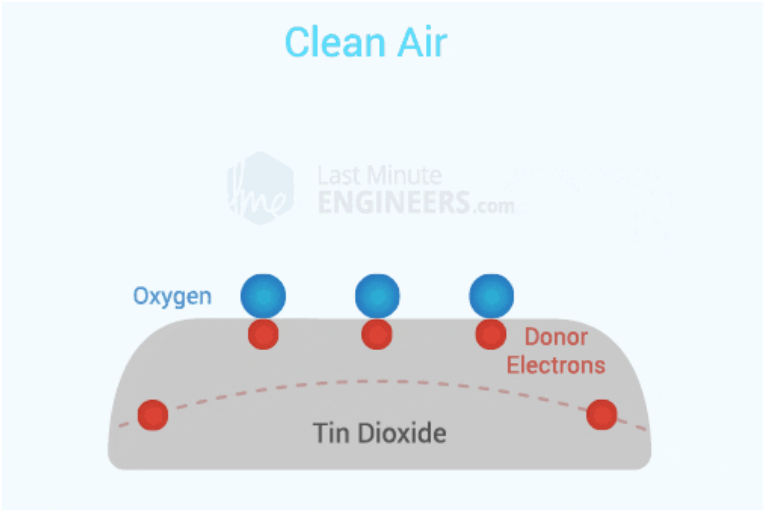
اگر این لایه توری را برداریم حسگر داخلی را مشابه تصویر فوق خواهیم یافت. حساسه این حسگر داری 6 پایه می باشد. دو عدد از این پایه ها که با حرف H نشان داده شده اند به سیم پیچی از جنس آلیاژ نیکل-کروم متصل هستند که وظیفه تولید حرارت را بر عهده دارند.

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.

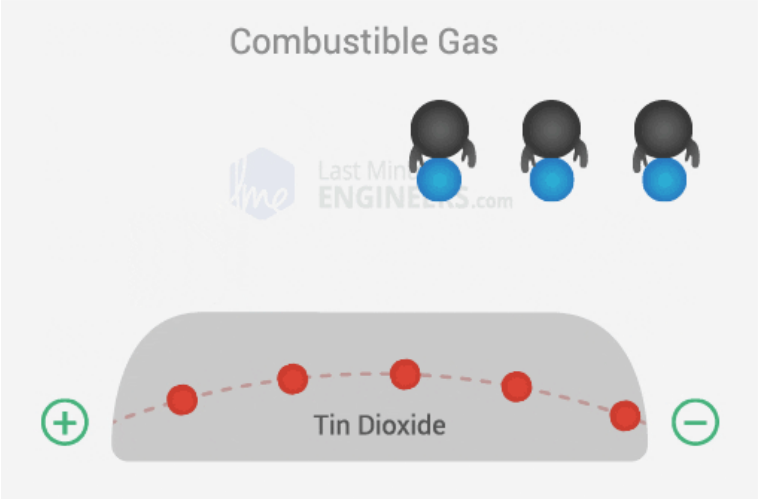


نحوه عملکرد سنسور MQ-2

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.



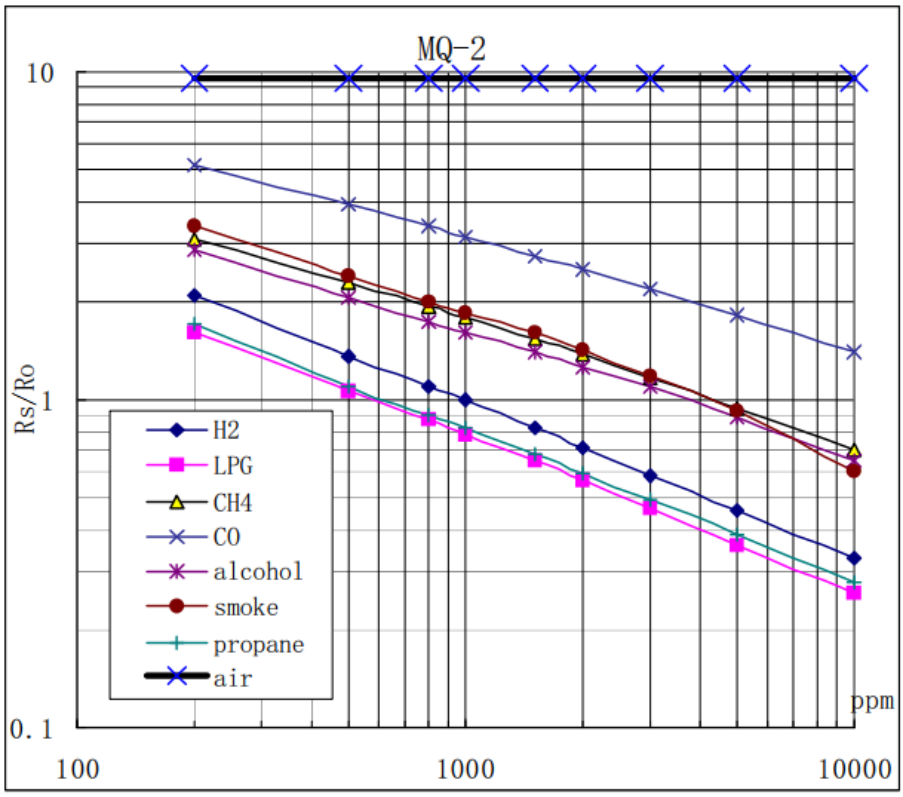
لذا با افزایش غلظت گازهای قابل احتراق ولتاژخروجی این حسگر بالا رفته و با کاهش غلظت این گازها این ولتاژپایین می‌آید.

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 24 تا 48 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 5 تا 10 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.

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.

اقلام مورد نیاز

- کریر بورد رزبری پیکو ProMake
- ماژول رزبری پیکو + کابل میکرو USB
- ماژول گاز MQ-2 ProMake

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake گاز MQ-2 در محل ProMake Module 1 و ماژول رزبری پیکو را در محل مربوطه بر روی کریر بورد قرار دهید و رزبری پیکو را توسط کابل USB به کامپیوتر متصل کنید.

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

کدنویسی و شرح کد

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

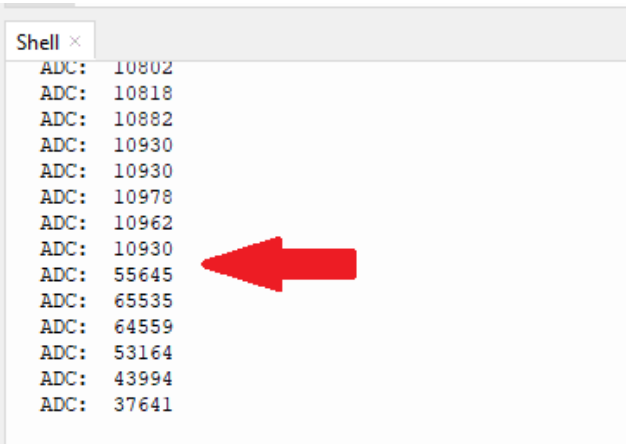
```
#RPI PICO ProMake Carrier Board
#Control WS2812B RGB LED

import machine
import utime

analog_value = machine.ADC(26)

while True:
    reading = analog_value.read_u16()
    print("ADC: ",reading)
    utime.sleep(0.8)
```

تابع خواندن مقادیر آنالوگ پایه‌ها

تابع `read_u16()`در میکرو پایتون مقدار آنالوگ را به یک عدد 2 بایتی تبدیل می کند. همانطور که قبلا اشاره شد، مبدل آنالوگ به دیجیتال رزبری پیکو دارای دقت 12 بیتی (یعنی مقادیر عدد صحیح بین $[0, 2^{12} - 1]$) است. به این معنی که ولتاژهای ورودی بین 0 تا 5 ولت را به مقادیر عدد صحیح بین 0 تا 4095 نگاشت می‌کند. بنابراین هر واحد $\frac{5}{4095} = 1.2\text{mV}$ است.

برای نمایش مقادیر خوانده ADC از سنسور گاز در محیط Shell نرم افزار Thonny از تابع print استفاده می کنیم و این مقادیر را با دوره زمانی 500 میلی ثنیه یکبار خوانده و چاپ می کنیم. نتیجه خروجی مشابه زیر خواهد بود. دقت نمائید که با توجه به ساختار سنسور MQ لازم است حداقل 10 دقیقه سنسور روشن باشد تا مقدار خروجی پایدار شود. برای تست می توانید یک فندک را سری سنسور نزدیک نمائید و بدون روشن کردن با فشردن دکمه آن قدری گاز متان را به پراب سنسور برسد در این حالت شما تغییر مقدار خروجی را در Shell خواهید دید.

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.



چالش و تمرین:

- طوری برنامه را بازنویسی نمائید که زمانیکه میزان خروجی سنسور از یک حد مشخصی(مانند 40000) بیشتر شد رنگ RGB1 قرمز چشمک زن و در بازه 40000 > x > 20000 به رنگ زرد چشمک زن و در بازه کمتر از 20000 رنگ سبز روشن شود.
- برنامه ای بنویسید که مشابه حالت با با رنج های عنوان شده علاوه بر روشن کردن RGB در رنگهای مختلف، بازر نیز با سطح صدای متفاوت و متناظر با رنج گاز اعلان هشدار نماید.

بخش چهارم: دریافت سیگنال از ورودی دیود(آشکارساز) IR

درس هفتم: ارتباط با دیود IR و دریافت فرمان از طریق اپلیکیشن موبایل

معرفی:

یکی از امکانات روی کریر بورد رزبری پیکو دیود آشکارساز IRهست که در ادامه در مورد نکات فنی آن توضیحات ارائه خواهد شد. دیود IR به پایه GP10 متصل است و با توجه به ماهیت دیجیتال خروجی آن براحتی می توان داده های دریافتی را خوانش کرد. از جهت تاریخی استفاده از IR انقلاب بزرگی در کنترل ریموت تجهیزات خصوصا تجهیزات خانگی رخ داد و سهولت زیادی در تجربه کاربری بوجود آمد. در این درس می خواهیم با استفاده از قطعه IR و نرم افزار موبایلی فرمان روشن کردن RGB و بازر را به کریر بورد ارسال کنیم.

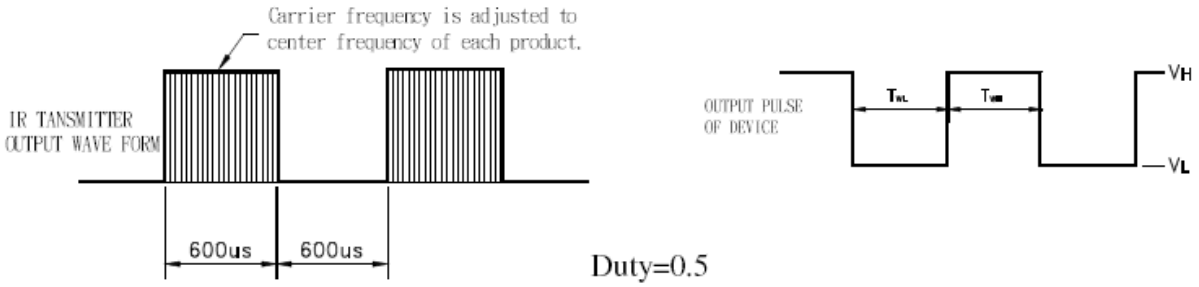
پیش نیاز 1: آشنایی با دیود IR با شماره فنی IRM-56384

آشکارسازهای IR ریزتراشه‌هایی کوچکی با یک فتوسل هستند که برای دریافت نور مادون قرمز تنظیم شده‌اند. اکثراً برای تشخیص کنترل از راه دور استفاده می شوند – هر تلویزیون و پخش کننده دی وی دی یکی از آنها را در پنل جلویش برای دریافت سیگنال IR دارد. در داخل کنترل از راه دور یک LED مادون قرمز وجود دارد که با انتشار پالس های IR به تلویزیون می گوید که کانال را روشن، خاموش یا تغییر دهد. نور مادون قرمز برای چشم انسان قابل مشاهده نیست.

فتوسل همانند آشکارسازهای IR به نور حساس هستند اما در عمل و کاربرد تفاوتهایی دارند که در زیر اشاره شده است:

- آشکارسازهای IR اختصاصا برای نور مادون قرمز فیلتر و تنظیم شده اند، آنها در تشخیص نور مرئی خوب نیستند. از سوی دیگر، فتوسل ها در تشخیص نور مرئی زرد/سبز خوب هستند، نه در نور مادون قرمز
- آشکارسازهای IR یک دمدولاتور داخلی دارند که به دنبال سیگنال IR مدوله شده در فرکانس 38 کیلوهرتز است. فقط روشن شدن یک LED IR تشخیص داده نمی شود، باید سیگنال IR با فرکانس 38 کیلوهرتز PWM چشمک بزنند. فتوسل ها هیچ نوع دمدولاتور داخلی ندارند و می توانند هر فرکانس (از جمله DC) را در بازه سرعت پاسخ فتوسل (که حدود 1 کیلوهرتز است) تشخیص دهند.

- آشکارسازهای IR دارای خروجی دیجیتالی هستند بدین شکل که یا سیگنال 38 کیلوهرتز IR را تشخیص می دهند و خروجی آنها پائین (0 ولت) است یا هیچ کدام را تشخیص نمی دهند و خروجی آن بالا (5 ولت) است. فتوسل ها مانند مقاومت ها عمل می کنند، بسته به میزان نوری که در معرض آنها قرار می گیرند، مقاومت تغییر می کند.



اقلام مورد نیاز

- کریر رزبری پیکو ProMake
- ماژول رزبری پیکو + کابل میکرو USB

آماده‌سازی سخت افزار

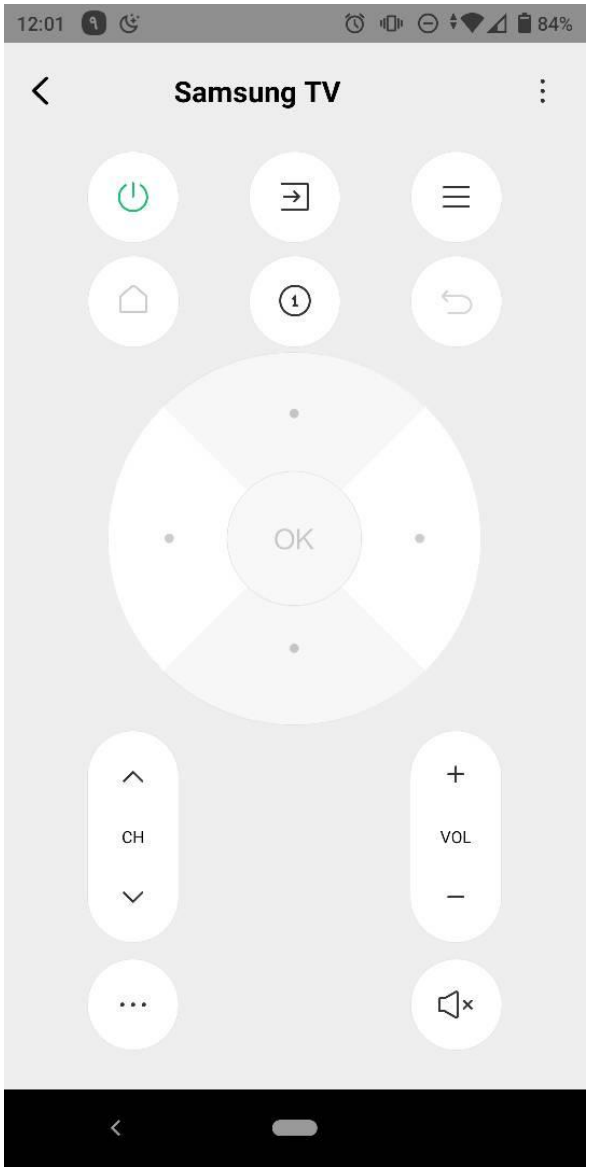
با توجه به چاپ روی برد ماژول رزبری پیکو را در محل مربوطه بر روی کریر بورد قرار دهید و آنرا توسط کابل USB به کامپیوتر متصل کنید.

کدنویسی و شرح کد

در نمونه کد زیر با دریافت کد IR از پین GP10 ، نمایشگر LED روی ماژول PICO روشن و خاموش می شود.

```
#RPI PICO ProMake Carrier Board
#Receiving IR Code and blink onboard LED and Buzzer

from machine import Pin, PWM
from utime import sleep_ms

ir=Pin(10,Pin.IN)
led=Pin(25,Pin.OUT)
buzzer = PWM(Pin(11))

while True:
    try:
        if ir.value()==0:
            led.value(1)
            buzzer.duty_u16(10000)
            buzzer.freq(1000)

        elif ir.value()==1:
            led.value(0)
            buzzer.duty_u16(0)

        sleep_ms(10)
    except KeyboardInterrupt:
        break
```

برای تست نرم افزار می توانید از انواع ریموت کنترل سخت افزاری و یا از نرم افزار های اندرویدی استفاده نمائید. در اینجا ما از برنامه Mi Remote استفاده کردیم. در نظر داشته باشید که هر برنامه اندرویدی و یا ریموت تلویزیون و یا دستگاه دیگری که در فرکانس 38kHz کار می کند، قابل استفاده است.



چالش و تمرین:

- طوری برنامه را بازنویسی نمائید که با دریافت صفر در ورودی RGB ها سبز و با دریافت یک به رنگ آبی روشن شوند.
- با مطالعه کدینگ NEC_8 برنامه را طوری تغییر دهید که بتوان مقادیر خام دریافتی را نمایش و طبق جدول دیکد، عدد یا دستور متناظر را نمایش دهد.

بخش پنجم: ارتباط از طریق I2C

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

معرفی:

یکی از پروتکلهای ارتباطی پرکاربرد در ماژولها و خصوصا IC ها واسط I2C هست که در ادامه به توضیح مشخصات و ویژگیهای آن می پردازیم. در این درس با استفاده از ماژول سنسور تگ پرومیک که دارای سنسور دما و رطوبت و نور برپایه I2C است و کریر بورد رزبری پیکو، اطلاعات سنسور خوانده می شود. سنسور دما/رطوبت قطعه SHT20 است که با پروتکل سریال I2C با پیکو ارتباط می گیرد. در ادامه اینترفیس I2C را معرفی می کنیم.

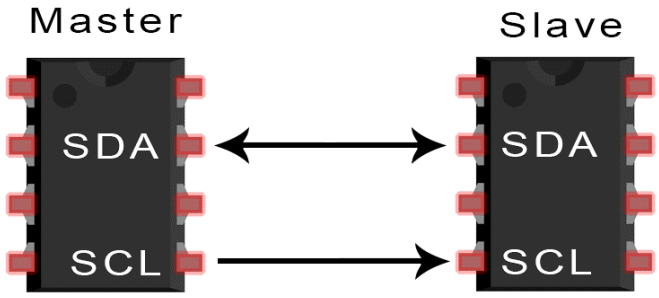
پیش نیاز : ارتباط I2C

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:

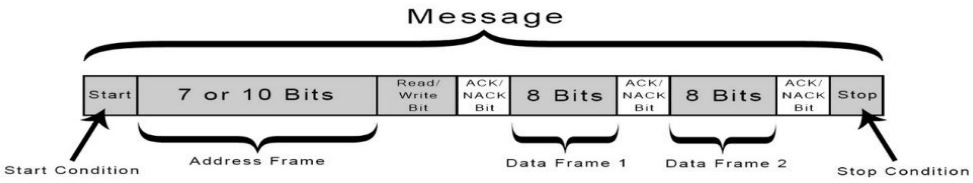
SDA (Serial Data) = خطی که برای ارسال و دریافت داده بین master و slave مورد استفاده قرار می‌گیرد.

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 بهslave 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 7 یا 10 بیتی منحصر به فرد برای هرSlave 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 ولتاژپایین) یا می‌خواهد از آن داده دریافت کند (سطح ولتاژبالا).

بیت ACK/NACK: به دنبال ارسال هر بخش از پیام یک بیت تصدیق کننده [4]/ عدم تصدیق کننده[5] 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.

[4] Ack

[5] Nack

مراحل انتقال داده از طریق پروتکل I2C

1- 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.

2- سپس آدرس 7 یا 10 بیتی و بیت read/write توسط Master برروی Bus فرستاده می‌شود و Master منتظر دریافت تصدیق می‌ماند.

3- 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 SDAدر حالت High ثابت باقی خواهد ماند.

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

5- 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.

6- 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 میکروپایتون از کتابخانه Machine کلاس تابع I2C را به برنامه وارد نمائید.

اقلام مورد نیاز

- کریر بورد رزبری پیکو ProMake
- ماژول ProMake Sensor TAG
- ماژول رزبری پیکو + کابل میکرو USB

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake Sensor TAG در یکی از ProMake Module قرار دهید. توجه داشته باشید که باس I2C در هر سه اسلات 1،2 و 3 وجود دارد و هیچ تفاوتی از جهت برنامه ندارد. ماژول رزبری پیکو را بر روی کریر پورد قرار دهید و پیکو را توسط کابل میکرو USB به کامپیوتر متصل کنید.

کدنویسی و شرح کد

ماژول رزبری پیکو دارای 2 باس سریال I2C مجزا است. برای ارتباط با باس I2C در کریر بورد رزبری پیکو از باس 0 با پینهای SCL:17 و SDA:16 استفاده می شود. مطابق برگه اطلاعات فنی قطعه SHT20 دارای آدرس 40 هگز است که می بایست طبق اطلاعات دیتا شیت برای خوانش دما و رطوبت بترتیب ابتدا دو مقدار متفاوت در آدرس قطعه بنویسیم و سپس مقدار خوانده شده متناظر با مقدار خوانده شده است.

تابع دما:

ابتدا مقدار 11110011 معادل F3 هگز را در آدرس 40 هگز می نویسیم که سپس بعد از یک وقفه 70 میلی ثانیه 2 بایت داده از قطعه خوانده می شود که مربوط به اطلاعات دما می باشد.

```
i2c.writeto(0x40,b'\xf3')
sleep(.070)
t=i2c.readfrom(0x40, 2)
```

مقدار خوانده شده طبق فرمول زیر به سانتیگراد تبدیل می شود.

```
-46.86+175.72*(t[0]*256+t[1])/65535
```

این مکانیزیم طبق دیتا شیت می باشد و لازم است که مراحل بترتیب اجرا شوند.

تابع رطوبت:

ابتدا مقدار 11110101 معادل F5 هگز را در آدرس 40 هگز می نویسیم که سپس بعد از یک وقفه 250 میلی ثانیه 2 بایت داده از قطعه خوانده می شود که مربوط به اطلاعات دما می باشد.

```
i2c.writeto(0x40,b'\xf5')
sleep(0.250)
t=i2c.readfrom(0x40, 2)
```

رطوبت خوانده شده طبق فرمول زیر به درصد تبدیل می شود.

```
-6+125*(t[0]*256+t[1])/65535
```

این مکانیزیم طبق دیتا شیت می باشد و لازم است که مراحل بترتیب اجرا شوند.

نمایش مقادیر در Shell:

برای نمایش مقادیر از دستور print استفاده می کنیم. با توجه به اینکه مقدار خام محاسبه شده فرمول دما و رطوبت با دقت 6 رقم اعشار می باشد . برای نمایش از دستور زیر و با متغیر float و دقت یک رقم اعشار استفاده می کنیم.

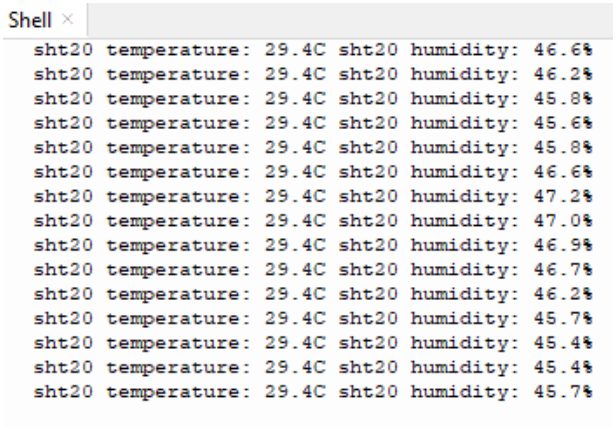
```
print("sht20 temperature: %0.1fC sht20 humidity: %0.1f%% " %(temper,humid)
```

کد کامل برنامه بشرح ذیل می باشد:

```
#RPI PICO ProMake Carrier Board
#Receiving temperature and Humidity by Sensor TAG Module

from machine import Pin, I2C, ADC
```

```
from utime import sleep


i2c = I2C(0, scl=Pin(17), sda=Pin(16))

################################################################################
def sht20_temperature():
    """Obtain the temperature value of SHT20 module
    Return:Temperature
    """
    i2c.writeto(0x40,b'\xf3')                          # Write byte "0xf3" to
address 0x40, SHT20
    sleep(.070)                                       # SHT20 measurement takes
time, must wait
    t=i2c.readfrom(0x40, 2)                            # Read 2 bytes of data
from the x40 address, SHT20
    return -46.86+175.72*(t[0]*256+t[1])/65535         # Perform temperature
conversion processing on the read data T=-46.86+175.72*St/2^16
################################################################################

def sht20_humidity():
    """Obtain the humidity value of SHT20 module
    Return:Humidity
    """
    i2c.writeto(0x40,b'\xf5')                          # Write byte "0xf5" to
address 0x40, SHT20
    sleep(0.025)                                       # SHT20 measurement takes
time, must wait
    t=i2c.readfrom(0x40, 2)                            # Read 2 bytes of data
from the x40 address, SHT20
    return -6+125*(t[0]*256+t[1])/65535                # Perform humidity
conversion processing on the read data RH=-6+125*Srh/2^16
################################################################################

while True:

    temper=sht20_temperature()
    humid=sht20_humidity()
    print("sht20 temperature: %0.1fC sht20 humidity: %0.1f%% "
%(temper,humid)
    sleep(1)
```

نتایج در پنجره Shell به شکل زیر است.



چالش و تمرین:

- با استفاده از LED های RGB برنامه ای بنویسید که در دمای درجه سانتیگراد 25 و کمتر از آن، RGB1,2 به رنگ سبز روشن شود و در دمای بالای 30 درجه و بیشتر رنگ RGB ها زرد شود.
- برنامه ای بنویسید که با فشردن کلید مقدار دما و رطوبت نمایش داده شود.
- طوری برنامه را بازنویسی نمائید که مقدار دما با دو رقم اعشار و مقدار رطوبت بدون اعشار نمایش داده شود.
- برنامه را طوری تغییر دهید که با فشردن کلید هر بار واحد دما از سانتیگراد به فارنهایت و بلعکس تبدیل شود.

درس نهم: نمایش اطلاعات بر روی نمایشگر OLED

معرفی

در درس قبل با رابط سریال I2C آشنا شدید و داده های دما و رطوبت خوانده شده از باس را در Shell نمایش داده شد. در ادامه نمایشگر OLED با رابط I2C را به کریر اضافه می کنیم و مقادیر سنسورها را در آن نمایش می دهیم.

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. نمایشگر 0.91 اینچی OLED دارای 4 پایه به شرح زیر است:

- VCC: تغذیه نمایشگر – 5 ولت
- GND: زمین
- SCL: کلاک پروتکل I2C
- SDA: خط داده پروتکل I2C

برای استفاده از آن دو کتابخانه استفاده می‌شود. که از اینجا[6] قابل دانلود است. و یا می‌توان از داخل Thonny نصب و از آن ها استفاده نمود.

```
import ssd1306
```

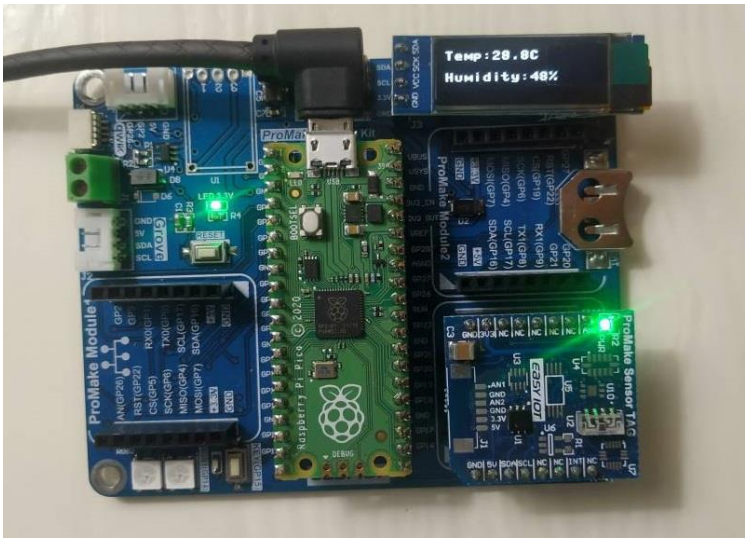
[6] https://github.com/stlehmann/micropython-ssd1306/blob/master/ssd1306.py

اقلام مورد نیاز

- کریر رزبری پیکو ProMake
- ماژول رزبری پیکو + کابل میکرو USB
- ماژول ProMake Sensor TAG
- نمایشگر OLED

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول ProMake Sensor TAG در یکی از ProMake Module قرار دهید. توجه داشته باشید که باس I2C در هر سه اسلات 1،2 و 3 وجود دارد و هیچ تفاوتی از جهت برنامه ندارد. ماژول رزبری پیکو را بر روی کریر پورد قرار دهید و پیکو را توسط کابل میکرو USB به کامپیوتر متصل کنید. نمایشگر OLED را نیز با توجه به مارکاژ پایه‌ها در جای تعبیه شده ، پین هدر مادگی 4 تایی J3 روی کریر 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.

```
#RPI PICO ProMake Carrier Board
#Receiving temperature and Humidity by Sensor TAG Module
#Display Sensor Data on OLED

from machine import Pin, I2C,ADC
from utime import sleep
import ssd1306


i2c = I2C(0, scl=Pin(17), sda=Pin(16))
oled = ssd1306.SSD1306_I2C(128, 32, i2c)

################################################################################

def sht20_temperature():
    """Obtain the temperature value of SHT20 module
    Return:Temperature
    """
    i2c.writeto(0x40,b'\xf3')                          # Write byte "0xf3" to
address 0x40, SHT20
    sleep(.070)                                       # SHT20 measurement takes
time, must wait
    t=i2c.readfrom(0x40, 2)                            # Read 2 bytes of data
from the x40 address, SHT20
    return -46.86+175.72*(t[0]*256+t[1])/65535         # Perform temperature
conversion processing on the read data T=-46.86+175.72*St/2^16
################################################################################

def sht20_humidity():
    """Obtain the humidity value of SHT20 module
    Return:Humidity
    """
    i2c.writeto(0x40,b'\xf5')                          # Write byte "0xf5" to
address 0x40, SHT20
    sleep(0.025)                                      # SHT20 measurement takes
time, must wait
    t=i2c.readfrom(0x40, 2)                            # Read 2 bytes of data
from the x40 address, SHT20
    return -6+125*(t[0]*256+t[1])/65535                # Perform humidity
conversion processing on the read data RH=-6+125*Srh/2^16
################################################################################

while True:

    temper=sht20_temperature()
    humid=sht20_humidity()


    oled.fill(0)
    oled.text("Temp:%0.1fC" %(temper),0,2,1)
    oled.text("Humidity:%d%%" %(humid),0,20,1)
    oled.show()

    print("sht20 temperature: %0.1fC sht20 humidity: %0.1f%%  "
%(temper,humid))
```

```
    sleep(1)
```

توابع کار با نمایشگر

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

```
import ssd1306
oled = ssd1306.SSD1306_I2C(128, 32, i2c)
```

پاک کردن نمایشگر و به عبارتی خاموش کردن تمامی پیکسل‌ها

```
oled.fill(0)
```

تنظیم موقعیت نشانگر و شروع نمایش

```
oled.text("Temp:%0.1fC" %(temper),0,2,1)
oled.text("Humidity:%d%%" %(humid),0,20,1)
```

نمایش متن

```
oled.show()
```

چالش و تمرین:

- با استفاده از توابع SSD1306 ، سایز فونت نوشته ها را تغییر دهید.
- برنامه ای بنویسید که با فشردن کلید مقدار دما و رطوبت در OLED نمایش داده شود.
- طوری برنامه را بازنویسی نمائید که مقدار دما با دو رقم اعشار و مقدار رطوبت بدون اعشار در OLED نمایش داده شود.

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

معرفی:

در درسهای هشتم و نهم توسط رابط سریال I2C با سنسور دما و رطوبت و نمایشگر OLED ارتباط گرفتیم و مقادیر مرتبط خوانده شد. در اینجا می خواهیم اطلاعات میزان نور محیط را با استفاده از سنسور نور روی ماژول سنسور تگ بخوانیم و نسبت به میزان نور محیط به ماژول رله فرمان می دهیم. در اینجا می توانید با رعایت تمام ملاحظات و احتیاط های لازم، خروجی رله را به یک لامپ (با ملاحظه جریان مصرفی) متصل نمائید و بدین شکل روشنایی اتاق را هوشمند نمائید.

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 مبدلA/D با دقت 16 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 120K 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 زیر استفاده می‌شود که از اینجا[7] قابل دانلود است.

```
import veml7700
```

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

```
veml = veml7700.VEML7700(address=0x10, i2c=i2c, it=100, gain=1/8)
```

در واقع برای استفاده از سنسور نور از یک دستور بسایر کاربردی try استفاده شده است که در صورتیکه سنسور در کریر نصب نباشد پیغام خطا در Shell نمایش می دهد.

[7] https://github.com/palouf34/veml7700

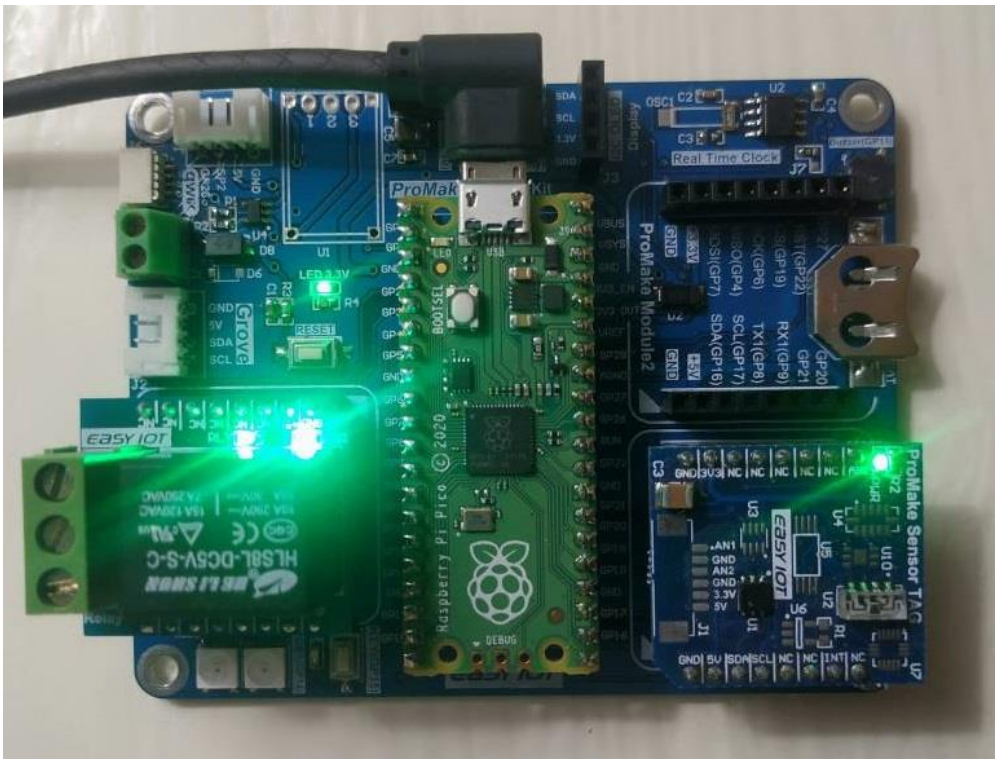
دو پارامتر بهره و زمان استفاده از این سنسور با استفاده از دو دستور زیر قابل تنظیم است. این پارامترها براساس تنظیمات نور محیط و تجربه کاری و اطلاعات فنی دیتا شیت تنظیم می شوند. در این برنامه از دو مقدار 100ms برای زمان و بهره 1/8 استفاده می شود. در صورت تنظیم نکردن مقادیر پیش فرض سنسور در نظر گرفته می شود.

اقلام مورد نیاز

- کریر رزبری پیکو ProMake
- ماژول رزبری پیکو + کابل میکرو USB
- ماژول ProMake Sensor TAG
- ماژول ProMake رله تک کانال (یا دو کانال)

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 3 و ProMake Module 1 و ماژول رزبری پیکو بر روی کریر بورد قرار دهید و پیکو 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.

```
#This code run on ProMake PI PICO Kit HW REV 1.2
#For getting data from SHT20 and VEML7700(Light Sensor)
# PM-RLY :Slot1

from machine import Pin, I2C,ADC
from time import sleep
import ssd1306
import veml7700


i2c = I2C(0, scl=Pin(17), sda=Pin(16), freq=10000)
O1 = machine.Pin(5 , machine.Pin.OUT , machine.Pin.PULL_UP) # Relay Output

try:
    veml = veml7700.VEML7700(address=0x10, i2c=i2c, it=100, gain=1/8)
    print("Ligt Sensor VEML7700 is founded.")

except:
    print("Ligt Sensor VEML7700 not found.")


while True:

    lux_val = veml.read_lux()
   # print("Lux_Value: %1d" %(lux_val))

    sleep(0.5)

    if (lux_val< 50):

        print("Ambient light is not enough.")
        print("Lux_Value: %1d" %(lux_val))
        O1.value(1)

    elif (lux_val > 500):

        print("Ambient light is sufficient")
        print("Lux_Value: %1d" %(lux_val))
        O1.value(0)
```

از یک حلقه بینهایت while استفاده می کنیم و با خواندن هر 500 میلی ثانیه یکبار سنسور نور، وضعیت فرمان رله را مشخص می کنیم. برای فرمان فعال شدن رله آستانه 50 لوکس استفاده شده و جهت غیر فعال کردن رله از آستانه 500 لوکس استفاده کردیم.

نکته اول اینکه میزان آستانه نور به شرایط محیطی و میزان نور لازم و مطلوب مرتبط است و بسادگی می توانید با توجه به شرایط واقعی این مقادیر را متناسب با پروژه تنظیم نمائید. برای دوری از نوسان حول نقطه آستانه از یک هسیترزیس استفاده کردیم.

چالش و تمرین:

- به برنامه نمایشگر oled را بیافزائید و نتایج قرائت سنسور نور و وضعیت رله را نمایش دهید.
- طوری برنامه را بازنویسی نمائید که در مقدار نور با لوکس زیر 50 رنگ RGB ها قرمز و بین 50 تا 500 زرد و بالای 500 به رنگ سبز روشن شوند.

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

معرفی:

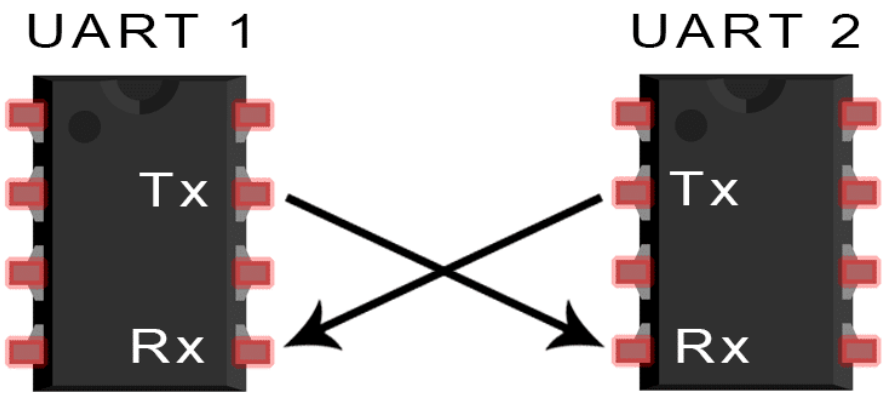
یکی از واسطهای ارتباطی مهم در موضوع اینترنت اشیاء ارتباط WiFi می باشد. این واسط ارتباطی دارای سهولت بالا و پهنای باند مناسب برای اکثر استفاده های عمومی و اختصاصی است. لذا برای ورود به دنیای متصل لازم است که با WiFi آشنا شویم. میکروکنترلرهایی مثل آردوینو نانو و رزبری پیکو در حالت عمومی دارای امکان ارتباط WiFi نیستند و لازم است که از یک ماژول برای این نیاز استفاده کرد.

البته اکنون سریهایی مثل PICO W دارای ارتباط WiFi و بلوتوث هستند و نیاز به ماژول بیرونی ندارد. اما در اکثر موارد می توان براحتی با یک ماژول خارجی امکان ارتباطی را به انواع میکروکنترلرهای مختلف افزود. یکی از ماژولهای محبوب در بخش ارتباط WiFi مربوط به شرکت Espressif می باشد که دارای نسلهای مختلفی از حیث نسل WiFi و توان پردازشگر می باشد.

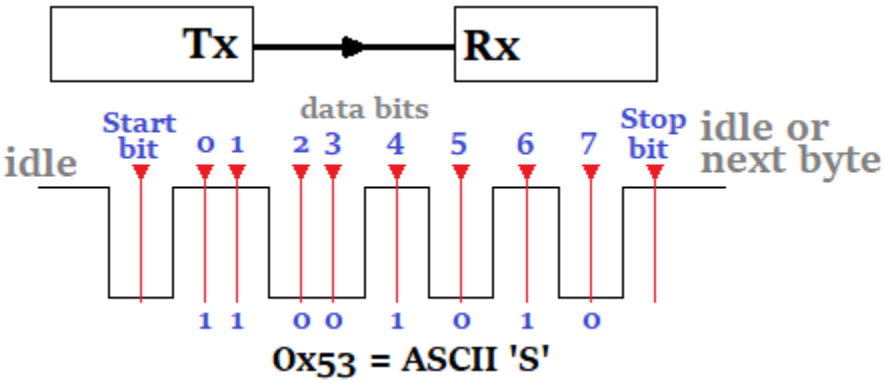
در این درس با ماژول ESP8266 از طریق واسط UART ارتباط می گیریم و اکسس پوینتهای موجود در پوشش ماژول را جستجو می کنیم و با یکی از آنها ارتباط می گیریم .

پیش نیاز 1: ارتباط UART

UARTیا همان (Universal Asynchronous Receiver Transmitter) یک ارتباط سریال آسنکرون با توانایی انتقال full duplex است که می‌تواند داده‌ها را به صورت سریال بین دو سخت افزار انتقال دهد. یک ارتباط سریال ارتباطی است که داده‌ها را به صورت بیت به بیت و یکی پس از دیگری منتقل می‌کند. در مقابل ارتباط سریال، ارتباط موازی قرار دارد که داده‌ها را به صورت موازی (چندین بیت با هم و در چند مسیر) انتقال می‌دهد.



در ارتباط آسنکرون، فرستنده و گیرنده باید بر اساس کلاک داخلی خودشان عمل کنند و هردو روی یک نرخ ارسال و دریافت یکسان تنظیم شده باشند. ارتباط صحیح در این نوع، مستلزم دارا بودن دقت کافی در کلاک داخلی دو سخت افزار است. به عبارت دیگر، دقت کلاک‌های داخلی دو سخت‌افزار فرستنده و گیرنده تعیین کننده حداکثر سرعت قابل دستیابی در ارتباط آسنکرون است.



یکی از پارامترهای مهم در ارسال سریال پارامتر Baud Rate است. در ارتباط UART آسنکرون فرستنده و گیرنده باید روی یک نرخ ارسال و دریافت یکسان تنظیم شوند. این نرخ ارسال و دریافت Baud Rate نامیده می‌شود. به عنوان مثال نرخ ارسال 9600 به این معنی است که داده‌ها با سرعت 9600 بیت در ثانیه ارسال و دریافت شوند. برای Baud Rate اعداد استانداردی مانند 2400، 4800، 9600، 19200، 38400، 115200 وجود دارد که 9600 و 115200 متداول‌ترین آن‌ها است. در ادامه تصویری از Baud Rate های معمول آورده شده است.

Transmission speed				Real transmission speed	
Bauds	Bit/s	Bit duration	Speed	Speed	Byte duration
50 Bd	50 bits/s	20.000 ms	6.25 bytes/s	5 bytes/s	200.000 ms
75 Bd	75 bits/s	13.333 ms	9.375 bytes/s	7.5 bytes/s	133.333 ms
110 Bd	110 bits/s	9.091 ms	13.75 bytes/s	11 bytes/s	90.909 ms
134 Bd	134 bits/s	7.463 ms	16.75 bytes/s	13.4 bytes/s	74.627 ms
150 Bd	150 bits/s	6.667 ms	18.75 bytes/s	15 bytes/s	66.667 ms
200 Bd	200 bits/s	5.000 ms	25 bytes/s	20 bytes/s	50.000 ms
300 Bd	300 bits/s	3.333 ms	37.5 bytes/s	30 bytes/s	33.333 ms
600 Bd	600 bits/s	1.667 ms	75 bytes/s	60 bytes/s	16.667 ms
1200 Bd	1200 bits/s	833.333 µs	150 bytes/s	120 bytes/s	8.333 ms
1800 Bd	1800 bits/s	555.556 µs	225 bytes/s	180 bytes/s	5.556 ms
2400 Bd	2400 bits/s	416.667 µs	300 bytes/s	240 bytes/s	4.167 ms
4800 Bd	4800 bits/s	208.333 µs	600 bytes/s	480 bytes/s	2.083 ms
9600 Bd	**9600 bits/s**	**104.167 µs**	**1200 bytes/s**	**960 bytes/s**	**1.042 ms**
19200 Bd	19200 bits/s	52.083 µs	2400 bytes/s	1920 bytes/s	520.833 µs
28800 Bd	28800 bits/s	34.722 µs	3600 bytes/s	2880 bytes/s	347.222 µs
38400 Bd	38400 bits/s	26.042 µs	4800 bytes/s	3840 bytes/s	260.417 µs
57600 Bd	57600 bits/s	17.361 µs	7200 bytes/s	5760 bytes/s	173.611 µs
76800 Bd	76800 bits/s	13.021 µs	9600 bytes/s	7680 bytes/s	130.208 µs
115200 Bd	**115200 bits/s**	**8.681 µs**	**14400 bytes/s**	**11520 bytes/s**	**86.806 µs**
230400 Bd	230400 bits/s	4.340 µs	28800 bytes/s	23040 bytes/s	43.403 µs
460800 Bd	460800 bits/s	2.170 µs	57600 bytes/s	46080 bytes/s	21.701 µs
576000 Bd	576000 bits/s	1.736 µs	72000 bytes/s	57600 bytes/s	17.361 µs
921600 Bd	921600 bits/s	1.085 µs	115200 bytes/s	92160 bytes/s	10.851 µs

در ماژول ProMake WiFi با قرار دادن در اسلات یک به UART0 ماژول پیکو متصل می شود و در ضمن ارتباط کراس هم برقرار می شود.

پیش نیاز 2 : AT command

برای کنترل چیپ ESP8266 بر روی ماژول ProMake Wi-Fi از سری دستوراتی به نام AT command استفاده می‌شود، 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 وای فای را تست می‌کند و در صورت موفقیت پاسخ OK دریافت می‌شود.

دستور AT+GMR

با دستور AT+GMR ورژن نرم افزاری ماژول وای فای خوانده می شود. گاها نیاز است که این ورژن را دانست و در صورت نیاز بروزرسانی نمود.

دستور AT+CWMODE=mode

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

پارامترهای Mode:

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

2 = با درج عدد 2 به جای کلمه Mode در دستور بالا، ماژول در حالت Access point قرار می‌گیرد. (حالت پیش فرض ماژول)

3 =با درج عدد 3 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 موجود 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 بر اساس نام و رمز مربوطه برقرار می‌کنند.

پارامترهای دریافتی:

- 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 یک یا چند اتصال همزمان وجود داشته باشد.

پارامتر ورودی mode:

- 0= فعال کردن تنها یک اتصال
- 1= 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 0 الی 4 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 2048 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 5 را به جای این پارامتر وارد کنید، چراکه طول رشته Hello پنج است.

پس از ارسال دستور فوق ماژول 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

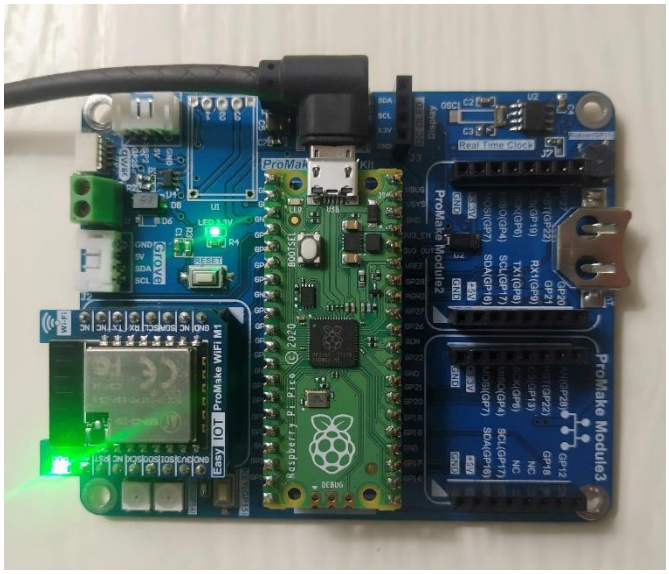
- **id=** این پارامتر تعیین کننده شناسه اتصالی است که قصد قطع ارتباط آنرا داریم و می‌بایست عددی بین 0 الی 4 به جای آن وارد کنیم. همچنین در صورتیکه تنها یک کانکشن دارید می‌توانید از وارد کردن این پارامتر صرف نظر کنید.

اقلام مورد نیاز

- کریر بورد رزبری پیکو ProMake
- ماژول رزبری پیکو + کابل میکرو USB
- ماژول ProMake WiFi M1

آماده‌سازی سخت افزار

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



کدنویسی و شرح کد

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

```
#RPI PICO ProMake Carrier Board
#Interfacing rpi PICO with ESP8266

from machine import Pin, UART
import utime


AP = SSID
PASS = PASSWORD


uart0 = UART(0, baudrate=9600, tx=machine.Pin(0), rx=machine.Pin(1), bits=8,
parity=None, stop=1)
esp_reset = Pin(22, machine.Pin.OUT)

#*****************************************************************************
**************************************************************
def Rx_ESP_Data():
    recv=bytes()
    while uart0.any()>0:
        recv+=uart0.read(1)
    res=recv.decode('utf-8')
    return res
#*****************************************************************************
**************************************************************
def Connect_WiFi(cmd, uart=uart0, timeout=3000):
    uart.write(cmd)
    utime.sleep(7.0)
    Wait_ESP_Rsp(uart, timeout)
#*****************************************************************************
**************************************************************
def Send_AT_Cmd(cmd, uart=uart0, timeout=3000):
    uart.write(cmd)
    Wait_ESP_Rsp(uart, timeout)
#*****************************************************************************
**************************************************************
def Wait_ESP_Rsp(uart=uart0, timeout=3000):
    prvMills = utime.ticks_ms()
    resp = b""
    while (utime.ticks_ms()-prvMills)<timeout:
        if uart.any():
            resp = b"".join([resp, uart.read(1)])

    try:
        print(resp.decode())
    except UnicodeError:
        print(resp)
#*****************************************************************************
**************************************************************

print("=== Start ===")
```

```
#hardware reset ESP
print("Hardware reset")
print()
esp_reset.value(1)
utime.sleep(0.5)
esp_reset.value(0)
utime.sleep(0.5)
esp_reset.value(1)

Send_AT_Cmd('AT\r\n')           #Test AT startup
Send_AT_Cmd('AT+GMR\r\n')       #Check version information
#Send_AT_Cmd('AT+RESTORE\r\n')  #Restore Factory Default Settings
Send_AT_Cmd('AT+CWMODE=1\r\n') #Set the Wi-Fi mode = Station mode
Send_AT_Cmd('AT+CWMODE?\r\n')  #Query the Wi-Fi mode again

preScanTime = utime.ticks_ms()
Send_AT_Cmd('AT+CWLAP\r\n', timeout=10000)  #List Available APs
print("Time used to Scan AP: ",
      utime.ticks_diff(utime.ticks_ms(), preScanTime),
      "(ms)")

Connect_WiFi('AT+CWJAP="'+ AP +'","' + PASS + '"\r\n', timeout=5000) #Connect
to AP
Send_AT_Cmd('AT+CIFSR\r\n',timeout=5000)    #Obtain the Local IP Address

print ('ESP8266 Configuration is done.')
```

در ابتدا برای کار کردن با بخش سریال از کتابخانه ماشین تابع UART را فرآخوانی می کنیم و سپس پارامترهای آنرا به شکل زیر تنظیم می کنیم.

```
from machine import UART
uart0 = UART(0, baudrate=9600, tx=machine.Pin(0), rx=machine.Pin(1), bits=8,
parity=None, stop=1)
```

با ارسال دستور AT وضعیت صحت ماژول را بررسی می کنیم. در صورت درستی OK را در پاسخ می گیرم.

```
Send_AT_Cmd('AT\r\n')
```

با ارسال دستور AT+GMR ورژن فریم ور ماژول ESP8266 نمایش داده می شود.

```
Send_AT_Cmd('AT+GMR\r\n')
```

با ارسال دستور AT+CWMODE وضعیت نوع ارتباط را مشخص می کند که در بخش پیش نیاز توضیح داده شد.

```
Send_AT_Cmd('AT+CWMODE=1\r\n')
```

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

```
Send_AT_Cmd('AT+CWLAP\r\n', timeout=10000)
```

و با ارسال دستور زیر که حاوی نام و رمز اکسس پوینتی است که قصد اتصال به آن را داریم، ماژول Wi-Fi به آن متصل می‌شود.

```
Connect_WiFi('AT+CWJAP="'+ AP +'","' + PASS + '"\r\n', timeout=5000)
```

با دستور زیر آدرس IP و MAC ماژول ESP8266 را در Shell د ریافت می کنیم.

```
Send_AT_Cmd('AT+CIFSR\r\n',timeout=5000)
```

نتیجه در Shell به صورت زیر خواهد بود.

```
=== Start ===
Hardware reset

AT

OK

AT+GMR
AT version:2.2.0.0(s-90458f0 - ESP32C3 - Jun 18 2021 10:24:22)
SDK version:v4.3-beta3-195-g6be10fab0
compile time(5fb0957):Jul  5 2021 13:46:35
Bin version:2.2.0(MINI-1)

OK

AT+CWMODE=1

OK

AT+CWMODE?
+CWMODE:1

OK

AT+CWLAP
+CWLAP:(4,"Rauof",-44,"ac:07:5f:b7:40:9b",9,-1,-1,5,3,7,0)
+CWLAP:(4,"MobinNet0981",-45,"ac:07:5f:b7:40:9c",9,-1,-1,5,3,7,0)
+CWLAP:(2,"fara",-70,"78:54:2e:d6:54:25",1,-1,-1,5,3,7,1)
+CWLAP:(4,"Sadra.",-73,"bc:0f:9a:4f:4a:ab",6,-1,-1,5,3,7,0)
+CWLAP:(3,"melody",-75,"10:fe:ed:2a:a4:95",11,-1,-1,5,3,7,0)
+CWLAP:(3,"Sarina",-85,"34:0a:33:a8:6b:38",6,-1,-1,4,4,7,1)
+CWLAP:(4,"Rastegaran-Main-Lan",-87,"3c:15:fb:cf:0e:30",3,-1,-1,5,3,7,0)
+CWLAP:(3,"Rastegaran-Ad-Hoc-2.4G",-91,"54:b8:0a:fe:75:09",1,-1,-1,4,4,7,1)
```

چالش و تمرین:

- به برنامه نمایشگر OLED را بیافزائید و AT Command و جواب ماژول را در نمایشگر نمایش دهید.
- طوری برنامه را بازنویسی نمائید که زمانیکه ارتباط WiFi برقرار شد رنگ RGB سبز شود.

پروژه اول: تولید موسیقی و رقص نور

معرفی:

با توجه به آموخته های قبلی در اولین پروژه رقص نور RGB و صدای نوت بازر را ترکیب می کنیم و با استفاده از امکان جالب دو هسته ای رزبری پیکو با دو حلقه ب نهایت همزمان دو عمل را بدون وقفه انجام می دهیم. در ادامه نحوه استفاده از هسته دوم CPU ماژول رزبری پیکو را بیان می کنیم.

اقلام مورد نیاز

- کریر رزبری پیکو ProMake(جهت استفاده از RGB LED ها و بازر روی آن)
- ماژول رزبری پیکو + کابل میکرو USB

آماده‌سازی سخت افزار

با توجه به نشانگرهای جهت ماژول رزبری پیکو 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 Thonny IDE کد زیر را نوشته و اجرا نمایید.

```
#RPI PICO ProMake Carrier Board
#Project one buzzer and RGB LED

from machine import Pin , PWM
from neopixel import Neopixel
from utime import sleep
import _thread

LED_NUM = 2
LED_PIN = 14

pixels = Neopixel(LED_NUM , LED_PIN , "GRB")
buzzer = PWM(Pin(11))
pixels.brightness(100)

tones = {"B0": 31,"C1": 33,"CS1": 35,"D1": 37,"DS1": 39,"E1": 41,"F1":
44,"FS1": 46,"G1": 49,"GS1": 52,"A1": 55,"AS1": 58,"B1": 62,"C2": 65,
```

```
"CS2": 69,"D2": 73,"DS2": 78,"E2": 82,"F2": 87,"FS2": 93,"G2": 98,"GS2":
104,"A2": 110,"AS2": 117,"B2": 123,"C3": 131,"CS3": 139,"D3": 147,
"DS3": 156,"E3": 165,"F3": 175,"FS3": 185,"G3": 196,"GS3": 208,"A3":
220,"AS3": 233,"B3": 247,"C4": 262,"CS4": 277,"D4": 294,"DS4": 311,
"E4": 330,"F4": 349,"FS4": 370,"G4": 392,"GS4": 415,"A4": 440,"AS4":
466,"B4": 494,"C5": 523,"CS5": 554,"D5": 587,"DS5": 622,"E5": 659,
"F5": 698,"FS5": 740,"G5": 784,"GS5": 831,"A5": 880,"AS5": 932,"B5":
988,"C6": 1047,"CS6": 1109,"D6": 1175,"DS6": 1245,"E6": 1319,"F6": 1397,
"FS6": 1480,"G6": 1568,"GS6": 1661,"A6": 1760,"AS6": 1865,"B6": 1976,"C7":
2093,"CS7": 2217,"D7": 2349,"DS7": 2489,"E7": 2637,"F7": 2794,
"FS7": 2960,"G7": 3136,"GS7": 3322,"A7": 3520,"AS7": 3729,"B7": 3951,"C8":
4186,"CS8": 4435,"D8": 4699,"DS8": 4978}

song =
["E5","G5","A5","P","E5","G5","B5","A5","P","E5","G5","A5","P","G5","E5"]

BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)
BLUE = (0, 0, 255)
PURPLE = (180, 0, 255)
WHITE = (255, 255, 255)

COLOR = [BLACK , RED , YELLOW , GREEN , CYAN , BLUE , PURPLE , WHITE]


def pixels_fill(color):
    for i in range(len(ar)):
        pixels.set_pixel(i, color)

def playtone(frequency):
    buzzer.duty_u16(400)
    buzzer.freq(frequency)

def bequiet():
    buzzer.duty_u16(0)

def playsong(mysong):
    for i in range(len(mysong)):

        if (mysong[i] == "P"):
            bequiet()
        else:
            playtone(tones[mysong[i]])
                #for i in range(len(ar)):
        sleep(0.3)
    bequiet()

def second_thread_rgb():
    while True:
        for color in COLOR:
```

```
            pixels.set_pixel(0, color)
            pixels.set_pixel(1, color)
            pixels.show()
            sleep(0.2)

        for i in range(len(COLOR)-1):

            pixels.set_pixel(0, COLOR[i])
            pixels.set_pixel(1, COLOR[i+1])
            pixels.show()
            sleep(0.2)


_thread.start_new_thread(second_thread_rgb, ())

while True:

        playsong(song)
        sleep(2)
```

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.

همانطور که قبلا اشاره کردیم برای انجام دو عمل همزمان تغییر رنگ RGB و پخش نوت لازم است از هر دوهسته پردازشگر ماژول رزبری پیکو استفاده کنیم لذا طبق برنامه بالا رقص نور RGB را با هسته دوم انجام می دهیم.

در حالت طبیعی برنامه نویسی از یک هسته استفاده می شود و کل برنامه به ترتیب توسط هسته 0 اجرا می شوند. اما جاهایی لازم است که یک کار بدون وقفه انجام شود و لذا هسته دوم در این کاربردها بسیار باارزش خواهد بود.

برای استفاده از هسته دوم کتابخانه thread_ را در ابتدای برنامه فراخوانی می کنیم.

```
import _thread
```

سپس در ادامه برنامه تابع رقص نور را که مستقل از بازر می باشد و بدون وقف باید عمل کند را با دستور زیر در یک حلقه بینهایت در هسته دوم اجرا می کنیم.

```
_thread.start_new_thread(second_thread_rgb, ())
```

چالش و تمرین:

- به برنامه کلید را بیافزائید و شروع اجرای برنامه در هر دو هسته را منوط به فشار کلید نمائید بطوریکه بار اول رقص نور و بازر با هم شروع و با فشار مجدد کلید هر دو متوقف شوند و بدین ترتیب ادامه یابد.
- برنامه ای بنویسید که در بخش رقص نور بصورت رنگین کمان RGB ها روشن شوند و نوت را به یکی از سمفونیهای معروف تغییر دهید