سخت افزار

ESP32: ESP32-cam

قلب اصلی ESP32-CAM یک تراشه سیستم-روی-چیپ (SoC) از نوع ESP32-S است که توسط شرکت Ai-Thinker طراحی شده. به عنوان یکSoC ، تراشه ESP32-S در واقع یک کامپیوتر کامل در یک تراشه است که شامل میکروپروسسور، حافظهRAM ، فضای ذخیره‌سازی و دستگاه‌های جانبی می‌شود. قابلیت‌های این تراشه به خودی خود خیلی جذاب هستند، اما برد توسعه ESP32-CAM امکانات و ویژگی‌های بیشتری را هم به آن اضافه کرده.
بیایید با هم هر یک از این اجزا را به‌طور جداگانه بررسی کنیم

!

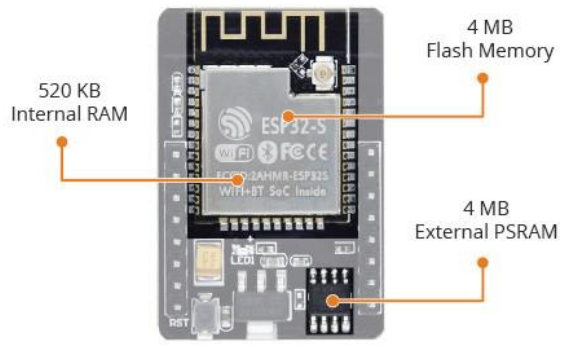
پردازنده ESP32-S

ماژول ESP32-CAM به پردازنده‌ی ESP32-S مجهز شده است، که یک مدار چاپی سطحی از شرکت Ai-Thinker است. این ماژول معادل ماژول ESP-WROOM-32 از شرکت Espressif بوده و از لحاظ شکل فیزیکی و مشخصات کلی، مشابه است.

پردازنده ESP32-S شامل یک میکروپردازنده Tensilica Xtensa® LX6 با دو هسته ۳۲ بیتی است که با سرعت شگفت‌انگیز ۲۴۰ مگاهرتز کار می‌کند! همین ویژگی‌ها ESP32-S را برای انجام وظایف سنگین مانند پردازش ویدئو، تشخیص چهره و حتی هوش مصنوعی مناسب ساخته است.

حافظه

حافظه نقش اساسی در انجام وظایف پیچیده دارد، و به همین دلیل، ESP32-S به ۵۲۰ کیلوبایت RAM داخلی مجهز است که بر روی همان تراشه‌ای قرار دارد که سایر اجزای آن تعبیه شده‌اند.

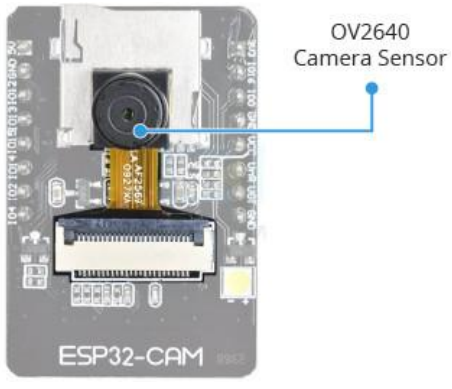


با این حال، این مقدار حافظه ممکن است برای وظایف پرمصرف از نظر RAM کافی نباشد. به همین دلیل، ESP32-CAM دارای ۴ مگابایت PSRAM خارجی (Pseudo-Static RAM) است که ظرفیت حافظه را افزایش می‌دهد. این مقدار حافظه برای پردازش‌های سنگین صوتی یا گرافیکی کاملاً کافی است.

اما تمام این ویژگی‌ها بدون فضای ذخیره‌سازی مناسب برای برنامه‌ها و داده‌ها بی‌فایده خواهد بود. در اینجا نیز تراشه ESP32-S می‌درخشد، زیرا ۴ مگابایت حافظه فلش داخلی را در خود جای داده است.

دوربین

سنسور دوربین OV2640 در ESP32-CAM، این برد را از سایر بردهای توسعه ESP32 متمایز می‌کند و آن را برای پروژه‌های ویدئویی مانند زنگ درب ویدئویی یا دوربین مراقبت کودک (Nanny Cam) ایده‌آل می‌سازد.



دوربین OV2640 دارای رزولوشن ۲ مگاپیکسل است که حداکثر وضوح تصویر آن به ۱۶۰۰×۱۲۰۰ پیکسل می‌رسد. این میزان وضوح برای بسیاری از کاربردهای نظارتی کاملاً کافی است.

علاوه بر این، ESP32-CAM با طیف گسترده‌ای از سنسورهای دوربین سازگار است، که انعطاف‌پذیری بیشتری در انتخاب سخت‌افزار به کاربر می‌دهد.

ذخیره‌سازی

اضافه شدن یک شیار کارت حافظه microSD به ESP32-CAM یک ویژگی جذاب و کاربردی محسوب می‌شود. این قابلیت امکان افزایش نامحدود فضای ذخیره‌سازی را فراهم می‌کند و این برد کوچک را به گزینه‌ای عالی برای ثبت داده‌ها (Data Logging) یا ذخیره تصاویر تبدیل می‌سازد.

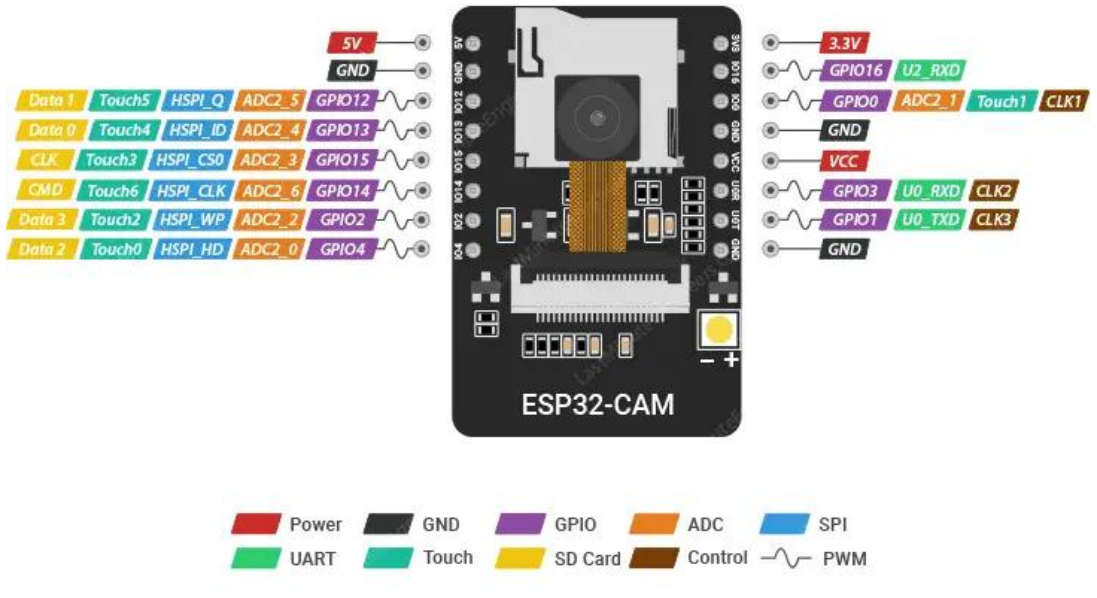
مصرف انرژی ESP32-CAM

میزان مصرف انرژی ESP32-CAM بسته به نوع استفاده متفاوت است.

در حالت عادی و زمانی که ویدئو استریم نمی‌شود، مصرف انرژی حدود ۸۰ میلی‌آمپر ساعت است. هنگام استریم ویدئو، این مقدار به حدود ۱۰۰ تا ۱۶۰ میلی‌آمپر ساعت می‌رسد. اگر فلش دوربین نیز روشن باشد، مصرف انرژی ممکن است به ۲۷۰ میلی‌آمپر ساعت افزایش یابد.

پین‌اوت ESP32-CAM

برد ESP32-CAM دارای ۱۶ پین است که برای سهولت، پین‌های با عملکرد مشابه در کنار یکدیگر قرار گرفته‌اند. توضیحات پین‌ها به شرح زیر است:



پین‌های تغذیه

- V5 و V3: ESP32-CAM3 می‌تواند از طریق پین‌های ۳.۳ ولت یا ۵ ولت تغذیه شود. با این حال، بسیاری از کاربران مشکلاتی را هنگام تغذیه با ۳.۳ ولت گزارش داده‌اند. بنابراین توصیه می‌شود این برد همیشه از طریق پین ۵ ولت تغذیه شود.

- VCC: معمولاً ولتاژ خروجی این پین ۳.۳ ولت است که از رگولاتور داخلی تأمین می‌شود، اما می‌توان با استفاده از لینک Zero-ohm نزدیک به پین VCC آن را به خروجی ۵ ولت تغییر داد.

- GND: این پین مخصوص اتصال به زمین است.

پین‌های GPIO

- GPIO: تراشه ESP32-S دارای ۳۲ پین GPIO است، اما بسیاری از آن‌ها برای استفاده داخلی دوربین و PSRAM رزرو شده‌اند. در نتیجه، ESP32-CAM تنها ۱۰ پین GPIO قابل استفاده دارد. این پین‌ها می‌توانند برای وظایفی مانند UART، SPI، ADC و Touch تنظیم شوند.

پین‌های UART

- تراشه ESP32-S دارای دو رابط UART به نام‌های UART0 و UART2 است. با این حال، تنها پین RX (GPIO 16) مربوط به UART2 در دسترس است. بنابراین، UART0 (GPIO 1 و GPIO 3) تنها رابط UART قابل استفاده در ESP32-CAM است.

- به دلیل نبود پورت USB در ESP32-CAM، از این پین‌ها برای فلش کردن برد و اتصال به دستگاه‌های UART مانند GPS، حسگر اثر انگشت و سنسورهای فاصله استفاده می‌شود.

پین‌های کارت حافظه MicroSD

- این پین‌ها برای اتصال به کارت MicroSD به کار می‌روند. در صورتی که از کارت حافظه استفاده نمی‌کنید، می‌توانید این پین‌ها را به عنوان ورودی و خروجی معمولی استفاده کنید.

پین‌های ADC

- تنها پین‌های ADC2 در ESP32-CAM در دسترس هستند. با این حال، این پین‌ها به دلیل استفاده داخلی توسط درایور Wi-Fi هنگام فعال بودن وای‌فای، قابل استفاده نیستند.

پین‌های لمسی (Touch)

- ESP32-CAM دارای ۷ پین لمسی خازنی است. این پین‌ها می‌توانند تغییرات خازن (مانند نزدیکی انگشت انسان) را تشخیص دهند.

پین‌های SPI

- ESP32-CAM دارای یک SPI (VSPI) است که در حالت‌های Slave و Master عمل می‌کند.

پین‌های PWM

- ESP32-CAM دارای ۱۰ کانال PWM است (شامل تمام پین‌های GPIO) که توسط یک کنترلر PWM مدیریت می‌شوند. خروجی PWM برای کنترل موتورهای دیجیتال و LEDها کاربرد دارد.

برنامه نویسی ESP32-CAM

برای اطلاعات بیشتر و بررسی پین‌هایی که استفاده از آن‌ها ایمن است یا باید با احتیاط استفاده شوند، می‌توانید به راهنمای کامل پین‌اوتESP32-CAM ما مراجعه کنید.

برنامه نویسی ESP32-CAM می‌تواند چالش برانگیز باشد، زیرا این برد فاقد پورت USB داخلی است. به همین دلیل، کاربران نیاز به سخت‌افزار اضافی برای آپلود برنامه‌ها از محیط Arduino IDE دارند. این فرایند پیچیده نیست، اما کمی زمان‌بر و ناخوشایند است.

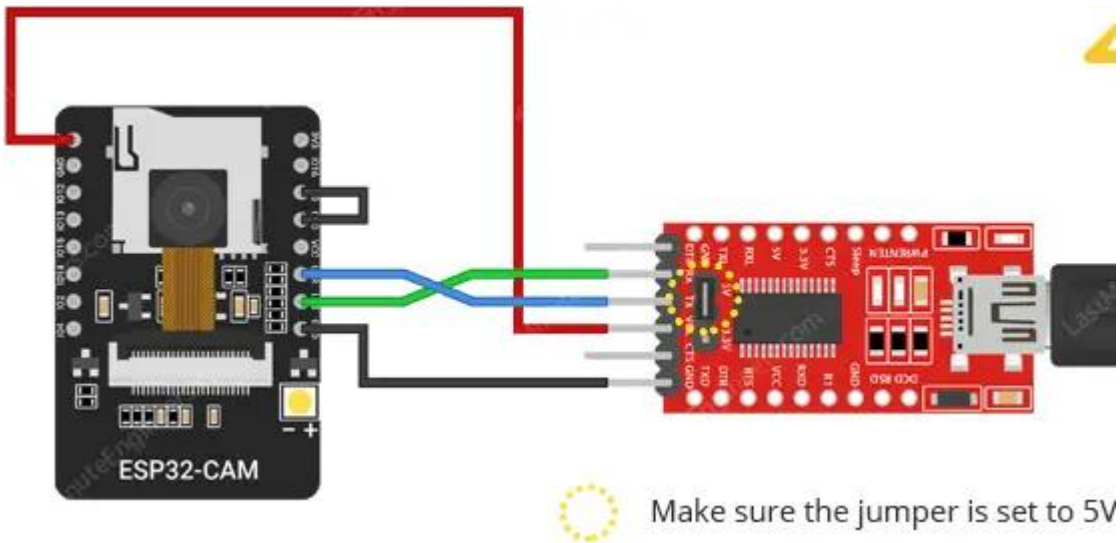
برای برنامه‌نویسی این دستگاه، به یکی از موارد زیر نیاز دارید:

-مبدل USB به سریال(FTDI Adapter)

-ماژول ESP32-CAM-MB Programmer Adapter

استفاده از مبدل FTDI (USB به TTL)

اگر تصمیم به استفاده از مبدل FTDI دارید، مراحل زیر را دنبال کنید:



تصال مبدل FTDI به ESP32-CAM

1. بسیاری از مبدل‌های FTDI دارای یک جامپر برای انتخاب ولتاژ خروجی 3.3 ولت یا 5 ولت هستند.

- از آنجا که ESP32-CAM با 5 ولت تغذیه می‌شود، مطمئن شوید که جامپر روی 5 ولت تنظیم شده باشد.

2. توجه داشته باشید که پین GPIO 0 باید به زمین (GND) متصل شود.

- این اتصال تنها در هنگام برنامه‌نویسی ESP32-CAM ضروری است. پس از اتمام برنامه‌ریزی، این اتصال باید قطع شود.

3. یادتان باشد! هر بار که قصد دارید یک اسکچ جدید آپلود کنید، باید این اتصال (GPIO 0 به GND) را دوباره برقرار کنید.

ویژگی‌های مهم مبدل USB به سریال

- مبدل‌ها از CH340G USB-to-Serial Converter استفاده می‌کنند که انتقال داده‌ها بین کامپیوتر و ESP32-CAM را مدیریت می‌کند.
- ماژول‌ها معمولاً دارای دکمه‌های RESET و BOOT هستند.
- یک LED برای نمایش وضعیت تغذیه و یک رگولاتور ولتاژ برای تأمین انرژی کافی به ESP32-CAM تعبیه شده است.

با استفاده از این ابزارها، برنامه‌نویسی ESP32-CAM راحت‌تر خواهد بود.

Servo

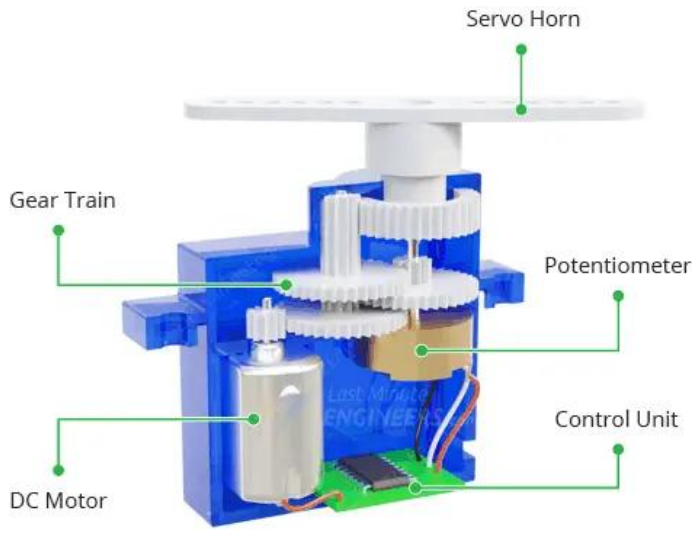
نحوه کارکرد سروو موتور و اتصال آن به آردوینو

در دنیای موتورها گزینه‌های متنوعی برای انتخاب وجود دارد، اما انتخاب موتور مناسب برای پروژه اهمیت زیادی دارد. اگر پروژه شما نیاز به کنترل موقعیت دقیق دارد، سروو موتور معمولاً بهترین گزینه است. کافی است به آن دستور دهید که در چه جهتی قرار گیرد و سروو این کار را برای شما انجام می‌دهد. به همین سادگی!

سروو موتورها در پروژه‌های مختلف رباتیک بسیار کاربردی هستند، مانند هدایت چرخ‌های جلو در مدل RC یا چرخاندن یک حسگر در یک وسیله نقلیه رباتیک.

سروو موتور چیست و چه چیزی آن را دقیق می‌کند؟

سروو موتورها این قابلیت را دارند که حرکت‌های فیزیکی را با دقت بسیار بالایی کنترل کنند، زیرا به طور کلی به یک موقعیت مشخص حرکت می‌کنند و چرخش مداوم ندارند. این موتورها به راحتی قابل اتصال و کنترل هستند زیرا درایور موتور به طور داخلی در آنها تعبیه شده است.



1. موتور DC کوچک: این موتور از طریق دنده‌ها (گیربکس) به شفت خروجی متصل است.

2. شفت خروجی: این شفت به یک بازوی سروو (Servo Horn) متصل است و از طریق یک پتانسیومتر (Potentiometer) موقعیت خروجی کنترل می‌شود.

3. پتانسیومتر (Pot): وظیفه ارائه بازخورد موقعیت به واحد کنترل را بر عهده دارد. این بازخورد به تقویت‌کننده خطا (Error Amplifier) در واحد کنترل ارسال می‌شود.

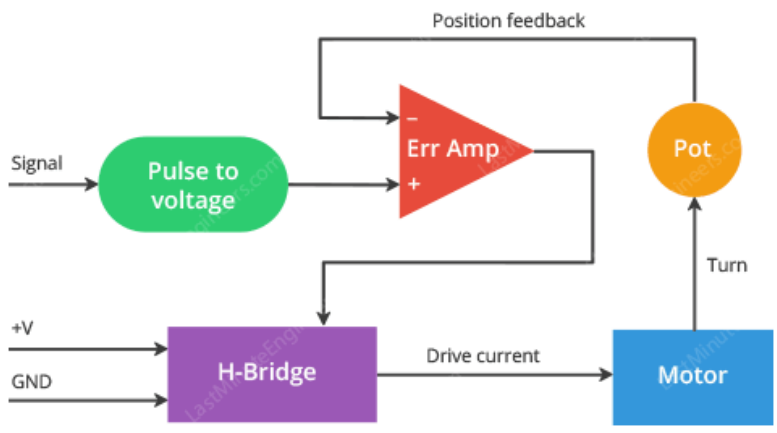
4. واحد کنترل: موقعیت فعلی موتور را با موقعیت هدف مقایسه کرده و در صورت وجود خطا، جریان موتور را تنظیم می‌کند تا موقعیت فعلی با موقعیت هدف تطابق پیدا کند.

نحوه عملکرد

این مکانیسم در مهندسی کنترل به عنوان سروومکانیزم شناخته می‌شود، یا به اختصار سروو نامیده می‌شود. سروو یک سیستم کنترل حلقه بسته است که از بازخورد منفی برای تنظیم سرعت و جهت موتور استفاده می‌کند تا نتیجه مطلوب حاصل شود.

کاربرد سروو موتور

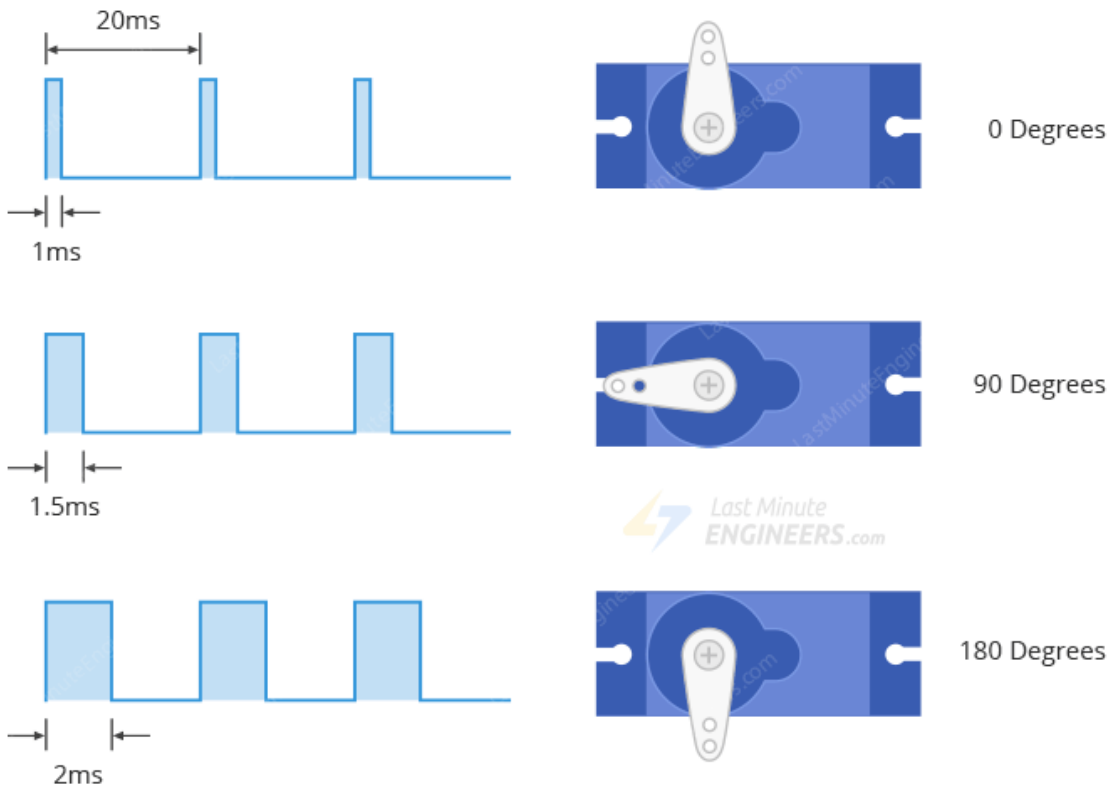
- هدایت چرخ‌ها در مدل‌های رادیو کنترل (RC).
- کنترل حسگرهای رباتیک با دقت بالا.
- استفاده در پروژه‌های بازوهای رباتیک و دستگاه‌هایی که به موقعیت‌دهی دقیق نیاز دارند.



سروو موتورها چگونه کار می‌کنند؟

کنترل سروو موتور از طریق ارسال سری پالس‌هایی به آن انجام می‌شود. یک سروو موتور معمولی انتظار دریافت یک پالس در هر ۲۰ میلی‌ثانیه را دارد (به عبارت دیگر، سیگنال باید فرکانس ۵۰ هرتز داشته باشد).

طول پالس موقعیت سروو موتور را تعیین می‌کند:



- پالس کوتاه (حدود ۱ میلی‌ثانیه یا کمتر): سروو را به زاویه ۰ درجه (یک موقعیت انتهایی) می‌چرخاند.

- پالس با طول ۱.۵ میلی‌ثانیه: سروو را به زاویه ۹۰ درجه (موقعیت میانی) می‌چرخاند.

- پالس با طول ۲ میلی‌ثانیه یا بیشتر: سروو را به زاویه ۱۸۰ درجه (موقعیت انتهایی دیگر) می‌چرخاند.

نحوه تعیین موقعیت سروو

پالس‌هایی با طول بین ۱ میلی‌ثانیه تا ۲ میلی‌ثانیه سروو را به موقعیتی متناسب با عرض پالس حرکت می‌دهند. به عبارت دیگر، عرض پالس دقیقاً مشخص می‌کند که سروو در چه زاویه‌ای قرار گیرد.

مثال: یک پالس با عرض ۱.۲ میلی‌ثانیه ممکن است سروو را به زاویه ۳۰ درجه و یک پالس با عرض ۱.۷ میلی‌ثانیه ممکن است آن را به زاویه ۱۳۵ درجه بچرخاند.

نکات مهم:

1. استاندارد ثابتی برای رابطه بین پالس‌ها و موقعیت وجود ندارد.

- ممکن است نیاز باشد که کد آردوینو خود را برای تنظیم محدوده حرکتی سروو تنظیم کنید.

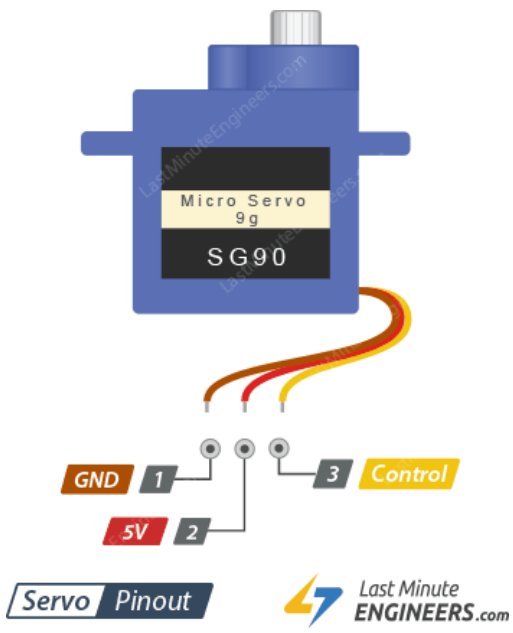
2. تفاوت بین برندها:

- طول پالس می‌تواند بین برندهای مختلف متغیر باشد. به عنوان مثال، برخی سروووها ممکن است برای زاویه ۱۸۰ درجه به پالس ۲.۵ میلی‌ثانیه و برای زاویه ۰ درجه به پالس ۰.۵ میلی‌ثانیه نیاز داشته باشند.

این سازوکار ساده اما مؤثر، دقت بالای سروو موتور را برای کاربردهای مختلف تضمین می‌کند.

پین‌های سروو موتور

سروو موتورها به طور معمول دارای سه اتصال هستند که به شرح زیر است:



1. GND: زمین مشترک برای موتور و منطق کنترلی.
2. V5: ولتاژ مثبت که سروو موتور را تغذیه می‌کند.
3. Control: ورودی برای سیستم کنترلی.

اتصال سروو موتور به ESP

در این آزمایش، از سروو میکرو SG90 استفاده خواهیم کرد. مشخصات این سروو به شرح زیر است:

- ولتاژ عملیاتی: ۴.۸ تا ۶ ولت DC (۵ ولت معمولی)
- محدوده چرخش: ۱۸۰ درجه (۹۰ درجه در هر جهت)
- جریان مصرفی:
  - ۱۰ میلی‌آمپر در حالت بیکار.
  - ۱۰۰ تا ۲۵۰ میلی‌آمپر هنگام حرکت.

> توجه: چون جریان مصرفی این سروو کمتر از ۲۵۰ میلی‌آمپر است، می‌توان آن را از طریق خروجی ۵ ولت آردوینو تغذیه کرد.
> اگر سرووی شما بیشتر از ۲۵۰ میلی‌آمپر مصرف می‌کند، از یک منبع تغذیه جداگانه استفاده کنید.

مراحل اتصال:

1. اتصال سیم قرمز:
   - به پین ۵ ولت ESP وصل کنید.
2. اتصال سیم مشکی/قهوه‌ای:
   - به پین GND ESP متصل کنید.
3. اتصال سیم نارنجی/زرد:
   - به پین PWM فعال (مثلاً پین شماره ۹) وصل کنید.

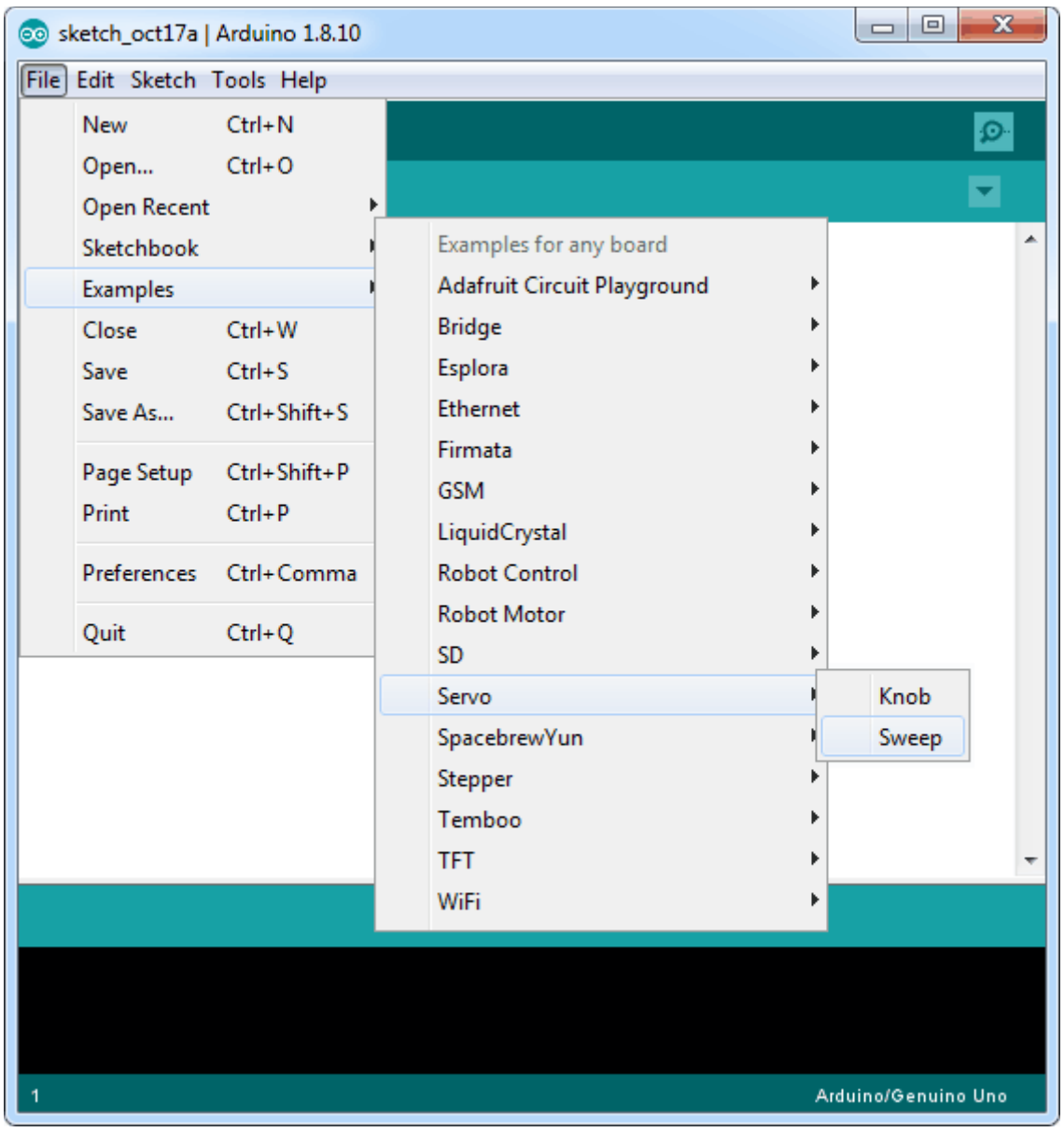
جدول اتصالات

سیم سروو موتور	پین ESP
قرمز (۵ ولت)	پین ۵ ولت ESP
مشکی/قهوه‌ای (GND)	پین GND ESP
نارنجی/زرد (Control)	پین PWM شماره ۹ ESP

پس از اتصال، می‌توانید کدهای مربوطه را برای کنترل سروو موتور در Arduino IDE آپلود کنید

مثال آردوینو ۱ – حرکت رفت و برگشتی (Sweep)

برای شروع، از یکی از مثال‌های پیش‌فرض Arduino IDE استفاده خواهیم کرد. از منوی Examples، بخش Servo را انتخاب کرده و اسکچ Sweep را بارگذاری کنید.



پس از آپلود کد روی برد، شفت سروو موتور شروع به حرکت رفت و برگشتی در بازه ۱۸۰ درجه خواهد کرد.

کد:

```
1. cpp
2. CopyEdit
3. `#include <Servo.h>  int servoPin = 9; Servo servo; int angle = 0;  //   موقعیت فعلی سروو به درجهvoid setup() {     servo.attach(servoPin); }
void loop() {      //      حرکت از ۰ تا ۱۸۰ درجهfor(angle = 0; angle < 180; angle++) {         servo.write(angle);         delay(15);     }
//      حرکت از ۱۸۰ به ۰ درجهfor(angle = 180; angle > 0; angle--) {         servo.write(angle);         delay(15);     } }`
4.
```

توضیح کد:

۱. استفاده از کتابخانه Servo

کنترل سروو موتور به صورت مستقیم پیچیده است، اما خوشبختانه Arduino IDE شامل یک کتابخانه به نام Servo است. این کتابخانه دستورات ساده‌ای را برای کنترل سروو ارائه می‌دهد. در ابتدای کد، این کتابخانه را وارد می‌کنیم:

```
1. cpp
2. CopyEdit
3. `#include <Servo.h>>`
4.
```

۲. تعیین پین کنترل سروو

پین آردوینو که سیم کنترل سروو موتور به آن متصل است، مشخص می‌شود:

```
1. cpp
2. CopyEdit
3. `int servoPin = 9;`
4.
```

۳. ایجاد شیء سروو

برای کنترل سروو، یک شیء از کلاس Servo ایجاد می‌کنیم:

```
1. cpp
2. CopyEdit
3. `Servo servo;`
4.
```

> نکته: می‌توان تا هشت سروو مختلف را به این شکل تعریف کرد:

```
1. cpp
2. CopyEdit
3. `Servo servo1; Servo servo2;`
4.
```

۴. تعریف متغیر زاویه

متغیر `angle` برای ذخیره مقدار زاویه فعلی سروو (بر حسب درجه) استفاده می‌شود:

```
1. cpp
2. CopyEdit
3. `int angle = 0;`
4.
```

۵. اتصال شیء سروو به پین کنترل

در تابع `setup`، از دستور `attach` استفاده می‌کنیم تا شیء سروو را به پین مشخص شده متصل کنیم:

```
1. cpp
2. CopyEdit
3. `servo.attach(servoPin);`
4.
```

۶. حلقه‌های حرکتی

در تابع `loop`، دو حلقه `for` قرار دارند:

- حلقه اول: سروو را از زاویه ۰ درجه به ۱۸۰ درجه حرکت می‌دهد.
- حلقه دوم: سروو را از ۱۸۰ درجه به ۰ درجه بازمی‌گرداند.

۷. به‌روزرسانی موقعیت سروو

دستور `servo.write(angle)` برای تنظیم موقعیت سروو به زاویه دلخواه استفاده می‌شود:

```
1. cpp
2. CopyEdit
3. `servo.write(angle);`
4.
```

نحوه عملکرد کد:

1. حرکت رفت: زاویه سروو از ۰ تا ۱۸۰ درجه افزایش می‌یابد.
2. حرکت برگشت: زاویه سروو از ۱۸۰ درجه به ۰ درجه کاهش می‌یابد.
3. هر حرکت با تاخیر ۱۵ میلی‌ثانیه اجرا می‌شود تا سروو فرصت کافی برای تنظیم موقعیت خود داشته باشد.

با استفاده از این کد، می‌توانید سروو موتور را با دقت کنترل کنید و حرکات رفت و برگشتی آن را مشاهده کنید.

عیب‌یابی

گاهی اوقات، سروو موتور ممکن است عملکرد نامنظمی داشته باشد، به خصوص اگر آن را مستقیماً از آردوینو تغذیه کنید. دلیل این مشکل این است که سروو در هنگام راه‌اندازی مقدار زیادی جریان مصرف می‌کند که ممکن است باعث ریست شدن برد آردوینو شود.

راه‌حل:

برای حل این مشکل، یک خازن الکترولیتی نسبتاً بزرگ با ظرفیت ۴۷۰ میکروفاراد تا ۱۰۰۰ میکروفاراد در ورودی تغذیه مدار قرار دهید. نکات زیر را رعایت کنید:

1. پایه بلندتر خازن را به پین ۵ ولت متصل کنید.
2. پایه کوتاه‌تر خازن (پایه منفی) را به پین GND متصل کنید.

نحوه عملکرد خازن:

- خازن بار الکتریکی را ذخیره می‌کند.
- زمانی که موتور شروع به کار می‌کند و جریان زیادی مصرف می‌کند، خازن انرژی ذخیره‌شده را آزاد می‌کند.
- این کار تضمین می‌کند که جریان به طور یکنواخت بین منبع تغذیه آردوینو و خازن توزیع شود و از مشکلاتی مانند ریست شدن آردوینو جلوگیری می‌کند.

نکته: اطمینان حاصل کنید که خازن به درستی قطبیت (+ و -) وصل شده باشد، زیرا اتصال نادرست ممکن است به خازن آسیب برساند یا باعث عملکرد نادرست شود.

Ultrasonic

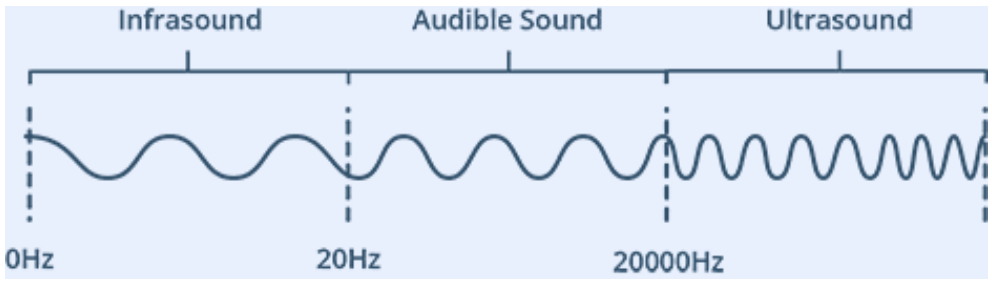
نحوه کار سنسور اولتراسونیک HC-SR04 و اتصال آن به آردوینو

پروژه آردوینو بعدی خود را با قدرت‌های شبیه به خفاش تجهیز کنید! سنسور فاصله‌سنج اولتراسونیک HC-SR04 می‌تواند فاصله اجسام را تا ۱۳ فوت (حدود ۴ متر) اندازه‌گیری کند. این قابلیت در هنگام جلوگیری از برخورد ربات به دیوار بسیار مفید است.

این سنسورها کم‌مصرف (مناسب برای دستگاه‌های باتری‌خور)، مقرون‌به‌صرفه، آسان برای اتصال و بسیار محبوب در بین علاقه‌مندان به الکترونیک هستند.

اولتراسونیک چیست؟

اولتراسوند یا امواج اولتراسونیک، نوعی موج صوتی با فرکانس بالا است که از محدوده شنوایی انسان فراتر می‌رود.



- انسان‌ها می‌توانند صداهایی را بشنوند که در بازه‌ی فرکانس ۲۰ هرتز (صداهای عمیق و بم) تا ۲۰,۰۰۰ هرتز (صدای سوت زیر) قرار دارند.
- امواج اولتراسونیک دارای فرکانسی بیش از ۲۰,۰۰۰ هرتز هستند و به همین دلیل برای گوش انسان غیرقابل شنیدن‌اند.

ویژگی‌های کلیدی HC-SR04

- حداکثر فاصله اندازه‌گیری: ۴ متر.

-حداقل فاصله اندازه‌گیری: ۲ سانتی‌متر.

- دقت اندازه‌گیری: ۳± میلی‌متر.

- مصرف انرژی: بسیار کم، مناسب برای پروژه‌های باتری‌خور.

- نحوه عملکرد: با ارسال و دریافت امواج اولتراسونیک.

نحوه عملکرد HC-SR04

1. ارسال امواج صوتی:
- سنسور HC-SR04 از طریق پین Trig یک پالس اولتراسونیک ارسال می‌کند.
2. برخورد با جسم و بازگشت:
- موج صوتی پس از برخورد با جسم به سمت سنسور بازمی‌گردد.
3. محاسبه فاصله:
- سنسور از زمان رفت و برگشت موج صوتی برای محاسبه فاصله استفاده می‌کند.
- فرمول فاصله: $Distance=Time×Speed\ of\ Sound2Distance = \frac{Time \times Speed \ of \ Sound}{2}Distance=2Time×Speed\ of\ Sound$ (سرعت صوت در هوا تقریباً ۳۴۳ متر بر ثانیه است).

مروری بر سخت‌افزار سنسور HC-SR04

سنسور فاصله‌سنج اولتراسونیک HC-SR04 شامل دو ترانسدیوسر اولتراسونیک است:

1. فرستنده (Transmitter):

سیگنال الکتریکی را به پالس‌های صوتی اولتراسونیک با فرکانس ۴۰ کیلوهرتز تبدیل می‌کند.

2. گیرنده (Receiver):

این بخش پالس‌های ارسال‌شده را دریافت می‌کند و یک پالس خروجی تولید می‌کند که عرض آن متناسب با فاصله جسم مقابل است.

ویژگی‌های عملکردی

- محدوده اندازه‌گیری: ۲ سانتی‌متر تا ۴۰۰ سانتی‌متر (حدود ۱۳ فوت).
- دقت: ۳ میلی‌متر.
- عملکرد غیرتماسی: مناسب برای تشخیص فاصله بدون نیاز به تماس فیزیکی.
- ولتاژ عملیاتی: ۵ ولت DC (قابل اتصال مستقیم به آردوینو یا میکروکنترلرهای منطقی ۵ ولت).

مشخصات فنی

مشخصه	مقدار
ولتاژ عملیاتی	DC 5V
جریان عملیاتی	۱۵ میلی‌آمپر
فرکانس عملیاتی	۴۰ کیلوهرتز
حداکثر فاصله	۴ متر
حداقل فاصله	۲ سانتی‌متر
دقت اندازه‌گیری	±۳ میلی‌متر
زاویه اندازه‌گیری	۱۵ درجه
سیگنال ورودی Trig	پالس TTL با عرض ۱۰ میکروثانیه
ابعاد	۴۵ × ۲۰ × ۱۵ میلی‌متر

نکات مهم

- اتصال ساده: این سنسور به دلیل ولتاژ عملیاتی ۵ ولت، مستقیماً به آردوینو یا سایر میکروکنترلرهای ۵ ولت متصل می‌شود.

پین‌های سنسور اولتراسونیک HC-SR04

در این بخش، به بررسی پین‌های این سنسور می‌پردازیم:

شرح پین‌ها

1. VCC

- وظیفه: تأمین برق سنسور.
- اتصال: به پین ۵ ولت آردوینو متصل می‌شود.

2. Trig (Trigger)

- وظیفه: ارسال پالس‌های صوتی اولتراسونیک.
- نحوه عملکرد: با تنظیم این پین به حالت HIGH به مدت ۱۰ میکروثانیه، سنسور یک پالس اولتراسونیک ارسال می‌کند.

3. Echo

- وظیفه: دریافت سیگنال بازتاب‌شده.
- نحوه عملکرد:
  - این پین زمانی که پالس اولتراسونیک ارسال می‌شود، به حالت HIGH می‌رود.
  - زمانی که سنسور بازتاب موج را دریافت کرد، این پین به حالت LOW بازمی‌گردد.
  - با اندازه‌گیری مدت زمان HIGH بودن پین Echo، می‌توان فاصله را محاسبه کرد.

4. GND

- وظیفه: اتصال به زمین.
- اتصال: به پین GND آردوینو متصل می‌شود.

نکات کلیدی

- VCC: باید به ولتاژ ۵ ولت متصل شود تا سنسور به درستی کار کند.
- Trig: برای فعال کردن پالس اولتراسونیک، این پین به مدت ۱۰ میکروثانیه به HIGH تنظیم می‌شود.
- Echo: زمان HIGH بودن این پین نشان‌دهنده زمان رفت و برگشت پالس است که برای محاسبه فاصله استفاده می‌شود.
- GND: برای تکمیل مدار، به زمین متصل می‌شود.

اتصال به آردوینو

- VCC → ۵ ولت آردوینو
- Trig → پین دیجیتال (به عنوان مثال، پین شماره ۷)
- Echo → پین دیجیتال (به عنوان مثال، پین شماره ۶)
- GND → GND آردوینو

نحوه عملکرد سنسور فاصله‌سنج اولتراسونیک HC-SR04

فرایند ارسال و دریافت سیگنال:

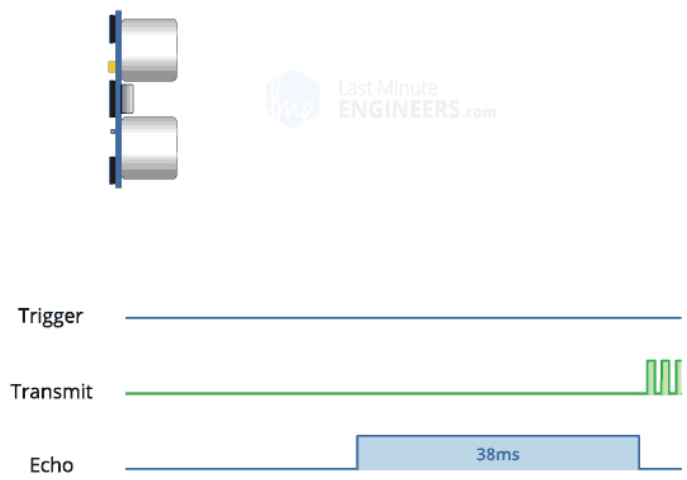
1. ارسال پالس اولتراسونیک:

وقتی پین Trigger به مدت ۱۰ میکروثانیه به حالت HIGH تنظیم می‌شود، سنسور یک مجموعه از ۸ پالس صوتی با فرکانس ۴۰ کیلوهرتز ارسال می‌کند.

- این الگوی ۸ پالس به‌گونه‌ای طراحی شده است که گیرنده بتواند این پالس‌ها را از نویزهای محیطی تمایز دهد.

2. شروع سیگنال Echo:

- پس از ارسال پالس‌ها، پین Echo به حالت HIGH می‌رود و نشان می‌دهد که سنسور در حال انتظار برای دریافت بازتاب است.

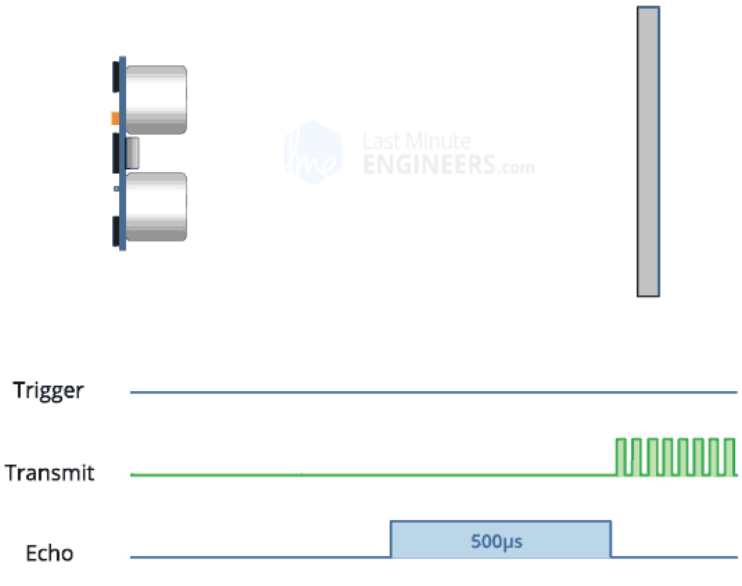


- اگر پالس‌ها بازتاب نشوند، سیگنال Echo پس از ۳۸ میلی‌ثانیه (۳۸ms) تایم‌اوت می‌شود و به حالت LOW بازمی‌گردد، که نشان‌دهنده نبود مانع در محدوده سنسور است.

- اگر پالس‌ها بازتاب شوند، سیگنال Echo بلافاصله پس از دریافت پالس به حالت LOW می‌رود.

4. عرض پالس Echo:

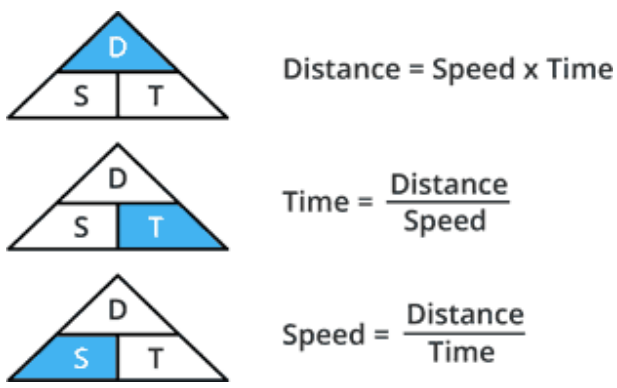
- عرض پالس Echo بسته به زمان رفت و برگشت سیگنال از ۱۵۰ میکروثانیه تا ۲۵ میلی‌ثانیه متغیر است.



محاسبه فاصله:

عرض پالس دریافتی برای محاسبه فاصله جسم استفاده می‌شود. این محاسبه با استفاده از فرمول ساده سرعت-مسافت-زمان که در دبیرستان آموخته‌ایم، انجام می‌شود:

$$Distance=Speed\times TimeDistance = Speed \times TimeDistance=Speed×Time$$



مثال:

فرض کنیم یک جسم در فاصله نامعلومی از سنسور قرار دارد و عرض پالس Echo دریافتی ۵۰۰ میکروثانیه است. فاصله جسم را به صورت زیر محاسبه می‌کنیم:

1. سرعت صوت در هوا:

- سرعت صوت $340 \, m/s340 \, m/s340m/s$ است.

- برای محاسبه راحت‌تر، آن را به $0.034 \, cm/\mu s0.034 \, cm/\mu s0.034cm/μs$ تبدیل می‌کنیم.

2. محاسبه فاصله اولیه:

$$Distance=0.034 \, cm/\mu s\times 500 \, \mu s=17 \, cmDistance = 0.034 \, cm/\mu s \times 500 \, \mu s = 17 \, cmDistance=0.034cm/μs×500μs=17cm$$

3. توجه به زمان رفت و برگشت:

چون سیگنال زمان رفت و برگشت را نشان می‌دهد، فاصله واقعی با تقسیم بر ۲ به دست می‌آید:

$$Distance=\frac{17 \, cm}{2} = 8.5 \, cmDistance=217cm=8.5cm$$

بنابراین، فاصله جسم از سنسور ۸.۵ سانتی‌متر است.

خلاصه فرایند

1. پین Trigger را برای ۱۰ میکروثانیه به HIGH تنظیم کنید.
2. سیگنال Echo را بررسی کنید و مدت زمان HIGH بودن آن را اندازه بگیرید.
3. با استفاده از فرمول: $Distance=\frac{Speed \times Time}{2}Distance=2Speed×Time$ فاصله را محاسبه کنید.

اتصال سنسور HC-SR04 به آردوینو

اتصال پین‌ها

اتصال سنسور اولتراسونیک HC-SR04 به آردوینو بسیار ساده است. ابتدا سنسور را روی یک بردبرد قرار دهید و به صورت زیر آن را به آردوینو متصل کنید:

|سنسور HC-SR04||آردوینو|

|---|---|---|

|VCC|→|5V|

|Trig|→|پین دیجیتال ۹|

|Echo|→|پین دیجیتال ۱۰|

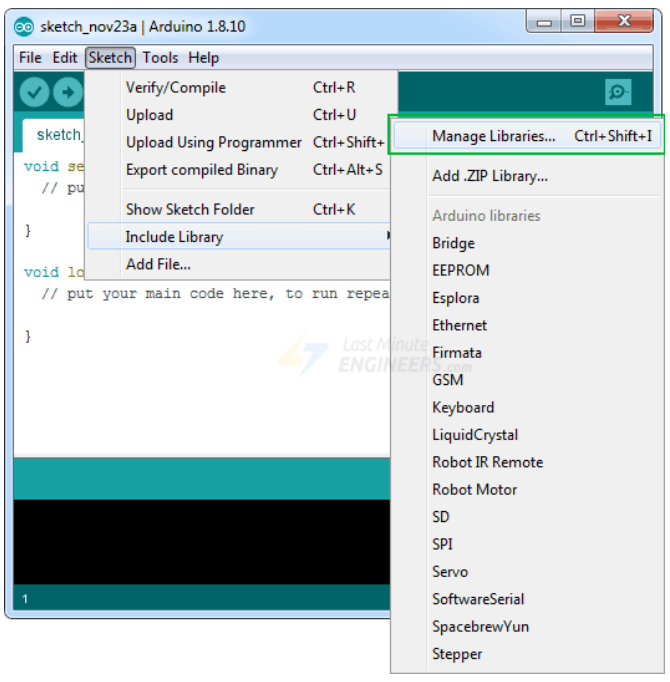
|GND|→|GND|

نصب کتابخانه

کنترل دستی پین‌ها برای ارسال و دریافت سیگنال‌های سنسور HC-SR04 کار زیادی می‌طلبد. خوشبختانه، کتابخانه‌هایی مانند NewPing این فرایند را ساده می‌کنند.

مراحل نصب NewPing Library

1. به Sketch > Include Libraries > Manage Libraries... بروید.
2. صبر کنید تا Library Manager لیست کتابخانه‌ها را به‌روزرسانی کند.
3. در قسمت جستجو، عبارت newping را وارد کنید.
4. اولین نتیجه را انتخاب کرده و روی Install کلیک کنید.



د نمونه آردوینو

کد زیر فاصله را محاسبه کرده و در مانیتور سریال نمایش می‌دهد:

cpp

CopyEdit

```
// حداکثر فاصله برای NewPing اضافه کردن کتابخانه #include "NewPing.h" // اتصال پین‌ها #define TRIGGER_PIN 9 #define ECHO_PIN 10 // ایجاد شیء NewPing NewPing sonar(TRIGGER_PIN, ECHO_PIN, اندازه‌گیری (بر حسب سانتی‌متر) #define MAX_DISTANCE 400 MAX_DISTANCE); void setup() { Serial.begin(9600); // شروع ارتباط سریال } void loop() { Serial.print("Distance = "); Serial.print(sonar.ping_cm()); // فاصله بر حسب سانتی‌متر Serial.println(" cm"); delay(500); // تاخیر برای به‌روزرسانی }
```

TFMini-S LiDAR

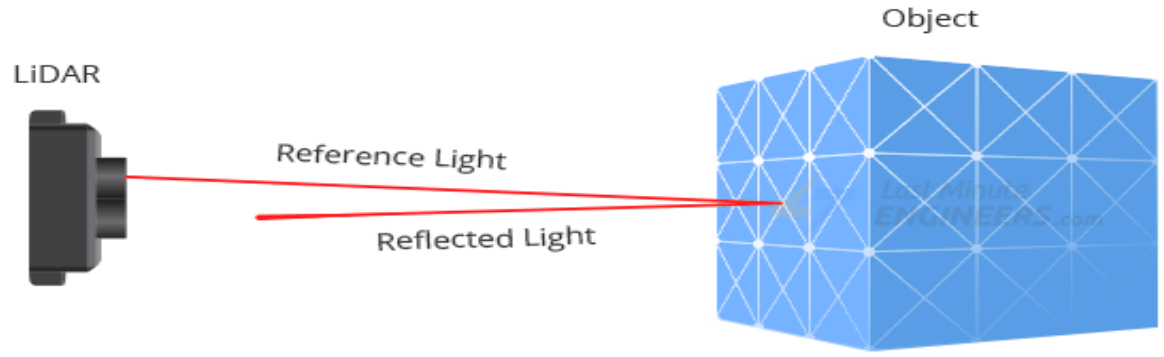
خودروهای خودران یکی از بزرگ‌ترین دستاوردهای فناوری از زمان پیدایش اینترنت هستند. این خودروها، برای پیمایش موفق در جاده، باید از موقعیت و وجود اشیاء فیزیکی در اطراف خود آگاه باشند. در اینجا سنسور LiDAR (نصب‌شده بر روی سقف خودرو) نقش مهمی ایفا می‌کند و یک نمای سه‌بعدی از جاده را در اختیار خودرو قرار می‌دهد.

اگرچه فناوری LiDAR جدید نیست و اولین اسناد مربوط به آن به سال ۱۹۶۳ بازمی‌گردد، هزینه بالای تجهیزات لیزر، این فناوری را محدود به سازمان‌های دولتی و نظامی کرده بود. اما کاهش اخیر هزینه‌ها، این فناوری را در دسترس علاقه‌مندان و سازندگان DIY قرار داده است. یکی از سنسورهای LiDAR ارزان و دقیق که به طور گسترده استفاده می‌شود، TFMini-S است.

این آموزش نحوه اتصال ماژول TFMini-S به آردوینو را برای اندازه‌گیری دقیق فاصله نشان می‌دهد. ابتدا مروری بر LiDAR خواهیم داشت.

LiDAR چیست و چگونه کار می‌کند؟

LiDAR مخفف عبارت Light Detection And Ranging است و ترکیبی از کلمات Light (نور) و RADAR (رادار) است. این فناوری مانند رادار عمل می‌کند، اما به جای امواج رادیویی، از نور استفاده می‌کند.



نحوه عملکرد:

1. تابش لیزر:

سنسور LiDAR یک پرتو لیزر به سمت جسم شلیک می‌کند.

2. بازگشت لیزر:

پرتو لیزر پس از برخورد به جسم بازمی‌گردد و توسط سنسور دریافت می‌شود.

3. محاسبه زمان رفت و برگشت:

با اندازه‌گیری زمان رفت و برگشت پرتو، فاصله جسم از سنسور محاسبه می‌شود.

$$Distance = \frac{Speed \ of \ Light \times Time}{2}$$

عوامل مؤثر بر اندازه‌گیری:

- محیط: شرایط محیطی (مانند غبار یا مه) می‌تواند دقت اندازه‌گیری را تحت تأثیر قرار دهد.
- بازتاب‌پذیری جسم: اجسامی با سطح بازتابی بیشتر (مانند فلزات براق) بهتر شناسایی می‌شوند.

نقشه‌برداری سه‌بعدی:

با چرخش یا جابجایی سنسور LiDAR، یک نقشه سه‌بعدی از محیط به سرعت ساخته می‌شود. این نقشه معمولاً به صورت نقطه‌ای (Point Cloud) نمایش داده می‌شود و دید واضحی از محیط اطراف ارائه می‌دهد.

بررسی سخت‌افزار TFMini-S LiDAR

TFMini-S یک سنسور LiDAR دقیق و تک‌نقطه‌ای از شرکت Benewake است که از فناوری ToF (Time of Flight) برای اندازه‌گیری فاصله استفاده می‌کند. این سنسور برای پروژه‌های رباتیک و تعاملی که نیاز به اندازه‌گیری دقیق فاصله دارند، ایده‌آل است.

این سنسور که به اندازه یک فلش USB است، امکان استفاده از LiDAR را در پروژه‌هایی فراهم می‌کند که پیش از این به سنسورهای کوچکی مانند فاصله‌سنج‌های مادون قرمز سری SHARP GP محدود بودند.

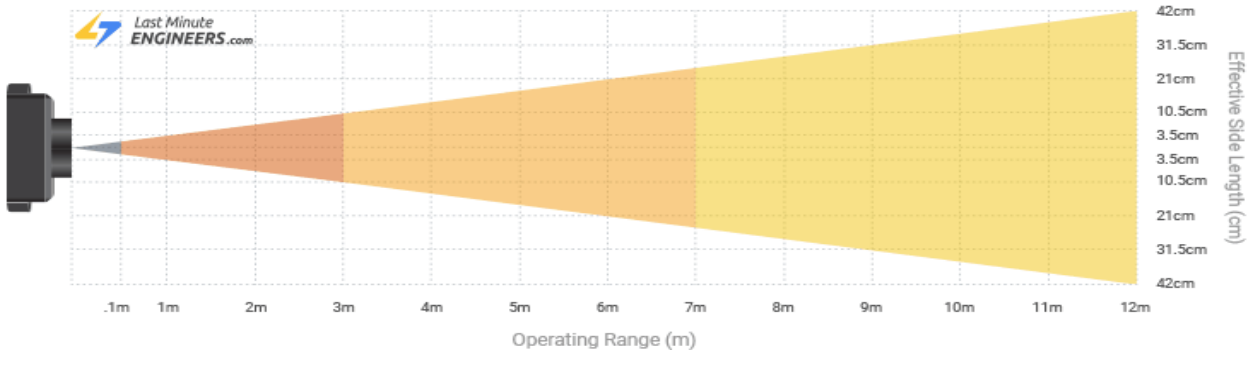
ویژگی‌های اصلی TFMini-S

1. دامنه اندازه‌گیری:
   - حداقل فاصله: ۱۰ سانتی‌متر.
   - حداکثر فاصله: ۱۲ متر.
2. دقت اندازه‌گیری:
   - ۶± سانتی‌متر برای فاصله‌های تا ۶ متر.
   - ۱٪± برای فاصله‌های بیشتر از ۶ متر.
3. **منبع نور:**

برخلاف سنسورهای LiDAR گران‌قیمت که از نور لیزر استفاده می‌کنند، TFMini-S از یک LED مادون قرمز با طول موج ۸۵۰ نانومتر و اپتیک متمرکز استفاده می‌کند. این امر باعث کاهش هزینه این دستگاه شده است.

دامنه تشخیص مؤثر

عملکرد سنسور TFMini-S به شرایط محیطی، نور، آب‌وهوا و بازتاب‌پذیری جسم هدف بستگی دارد:



فاصله ۰ تا ۱۰ سانتی‌متر به عنوان منطقه کور (Blind Zone) محسوب می‌شود و داده‌ها در این محدوده قابل‌اعتماد نیستند.

- شرایط شدید:

- در نور شدید روز (حدود k۱۰۰ لوکس) و در حال تشخیص اجسام سیاه با بازتاب ۱۰٪، دامنه اندازه‌گیری به ۰.۱ تا ۳ متر محدود می‌شود.

- شرایط عادی نور خورشید:

- در شدت نور خورشید متوسط (حدود k۷۰ لوکس) دامنه عملکرد بین ۰.۱ تا ۷ متر است.

- محیط داخلی یا نور کم:

- در محیط‌های داخلی یا نور کم، دامنه اندازه‌گیری به حداکثر ۰.۱ تا ۱۲ متر می‌رسد.

رابط‌های ارتباطی

- به طور پیش‌فرض، TFMini-S از طریق رابط UART با پین‌های RX و TX و نرخ ارتباط bps115200 ارتباط برقرار می‌کند.
- این سنسور همچنین می‌تواند با ارسال دستورات مناسب به رابط I2C تغییر حالت داده شود.

فرکانس اندازه‌گیری

- سنسور TFMini-S می‌تواند تا ۱۰۰۰ اندازه‌گیری در ثانیه انجام دهد (پیش‌فرض: ۱۰۰ اندازه‌گیری در ثانیه).
- افزایش فرکانس خروجی دقت را کاهش می‌دهد. بنابراین، با توجه به نیاز پروژه خود، فرکانس خروجی را تنظیم کنید.

توان ورودی

- ولتاژ عملیاتی: ۵ ولت DC.
- جریان مصرفی در هنگام اندازه‌گیری: حدود ۱۴۰ میلی‌آمپر (حداکثر جریان: ۲۰۰ میلی‌آمپر).
- در تست‌ها، سنسور به تنهایی حدود ۷۰ میلی‌آمپر جریان مصرف می‌کرد. بنابراین، می‌توان آن را با استفاده از پورت USB (۵ ولت/۵۰۰ میلی‌آمپر) تغذیه کرد.

نکته مهم: سنسور TFMini-S فاقد محافظ ولتاژ است؛ بنابراین، نوسانات ولتاژ ورودی را در محدوده ۰.۱ ولت نگه دارید.

سطوح منطقی

- سنسور با ولتاژ ۵ ولت تغذیه می‌شود، اما پین‌های I/O آن تنها از منطق ۳.۳ ولت پشتیبانی می‌کنند.
- توصیه: هنگام استفاده از سنسور با میکروکنترلرهای ۵ ولت، از مبدل سطح منطقی (Logic Level Converter) استفاده کنید.
- اگر فقط قصد خواندن داده‌ها در حالت UART را دارید، نیازی به مبدل سطح منطقی نیست؛ زیرا خروجی دستگاه‌های ۳.۳ ولت با میکروکنترلرهای ۵ ولت سازگار است.

خلاصه ویژگی‌ها

- ابعاد کوچک: مناسب برای پروژه‌های محدود به فضا.
- دقت بالا: مناسب برای کاربردهای رباتیک و تعاملی.
- قابلیت تنظیم رابط ارتباطی و فرکانس: انعطاف‌پذیری بالا.

مشخصات فنی TFMini-S

در اینجا مشخصات سنسور TFMini-S آورده شده است:

مشخصه	مقدار
دامنه تشخیص	۱۰ سانتی‌متر – ۱۲ متر
دقت اندازه‌گیری	۱ سانتی‌متر
دقت فاصله	±۶ سانتی‌متر تا ۶ متر و ±۱٪ بعد از آن
ولتاژ ورودی	۵ ولت DC
ولتاژ	TTL UART۳.۳ ولت
جریان مصرفی	۱۴۰ میلی‌آمپر (معمولی)، ۸۰۰ میلی‌آمپر (پیک)
فرکانس اندازه‌گیری	۱ تا ۱۰۰۰ اسکن در ثانیه (قابل تنظیم)
طول موج نور	۸۵۰ نانومتر
زاویه دید	°۲.۳
رابطهای ارتباطی	UART و I2C
نرخ	Baud۱۱۵۲۰۰

پین‌های TFMini-S

ESF

 # معرفی ESP-IDF (چارچوب توسعه اینترنت اشیا اسپرسف)

ESP-IDF (Espressif IoT Development Framework) چارچوب رسمی توسعه ارائه شده توسط شرکت Espressif برای توسعه برنامه‌های اینترنت اشیا (IoT) با استفاده از میکروکنترلرهای سری ESP32 است. این چارچوب امکانات پیشرفته‌ای را برای دسترسی به سخت‌افزار و کنترل نرم‌افزار ارائه می‌دهد و برای ساخت سیستم‌های مقیاس‌پذیر و بهینه اینترنت اشیا ایده‌آل است.

ویژگی‌های کلیدی ESP-IDF

1. کنترل سطح پایین:

دسترسی مستقیم به امکانات سخت‌افزاری ESP32 از جمله:

- GPIO
- I2C
- SPI
- PWM
- Wi-Fi
- Bluetooth

2. پشتیبانی از سیستم‌عامل زمان واقعی (RTOS):

این چارچوب بر پایه FreeRTOS ساخته شده است که امکان توسعه برنامه‌های چندوظیفه‌ای و زمان واقعی را فراهم می‌کند.

3. پشتیبانی گسترده از کتابخانه‌ها:

شامل کتابخانه‌هایی برای:

- Wi-Fi
- Bluetooth
- HTTPS
- MQTT
- به‌روزرسانی‌های OTA

4. ماژولار و مقیاس‌پذیر:

مناسب برای پروژه‌های ساده تا سیستم‌های پیچیده IoT.

5. متن‌باز:

کاملاً متن‌باز است و توسط Espressif به طور فعال پشتیبانی می‌شود.

چرا از ESP-IDF برای ESP32 استفاده کنیم؟

در حالی که Arduino IDE برای مبتدیان مناسب و کاربرپسند است، ESP-IDF برای توسعه‌دهندگان پیشرفته که به موارد زیر نیاز دارند، طراحی شده است:

- کنترل بیشتر بر روی سخت‌افزار و نرم‌افزار.
- دسترسی به APIهای سطح پایین.
- بهینه‌سازی عملکرد برای وظایف سنگین.
- راه‌حل‌های مقیاس‌پذیر برای سیستم‌های پیچیده اینترنت اشیا.

نصب ESP-IDF برای توسعه ESP32

۱. نصب ESP-IDF

برای نصب ESP-IDF، مراحل زیر را دنبال کنید:

- مخزن رسمی را کلون کنید:

```
1.      bash
2.      CopyEdit
3.      `git clone --recursive https://github.com/espressif/esp-idf.git cd esp-idf ./install.sh`
4.
5.
```

۲. تنظیم متغیرهای محیطی

برای تنظیم محیط، دستور زیر را اجرا کنید:

```
1. bash
2. CopyEdit
3. `. ./export.sh`
4.
```

ساختار یک پروژه ESP-IDF

یک پروژه ESP-IDF ساختاری ماژولار دارد و شامل موارد زیر است:

1. دایرکتوری main:

کد اصلی برنامه شما (مثل main.c یا main.cpp) در اینجا قرار می‌گیرد.

2. دایرکتوری components:

یک دایرکتوری اختیاری برای ماژول‌ها و کتابخانه‌های قابل استفاده مجدد.

3. فایل CMakeLists.txt:

فایل‌های پیکربندی برای ساخت پروژه با استفاده از CMake.

4. فایل sdkconfig:

فایل پیکربندی برای سفارشی‌سازی ویژگی‌های ESP-IDF.

مثال پروژه ESP-IDF: چشمک زدن LED

۱. تنظیم پروژه

یک پروژه جدید ایجاد کنید:

```
1. bash
2. CopyEdit
3. `idf.py create-project blink cd blink`
4.
```

۲. ویرایش main.c

کد زیر را وارد کنید:

```
1. CopyEdit
2. `#include <stdio.h> #include "freertos/FreeRTOS.h" #include "freertos/task.h" #include "driver/gpio.h"
#define BLINK_GPIO GPIO_NUM_2  void app_main() {     gpio_pad_select_gpio(BLINK_GPIO);
gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);      while (1) {         gpio_set_level(BLINK_GPIO, 1);  //
 روشن کردنLED         vTaskDelay(1000 / portTICK_PERIOD_MS);  // تاخیر ۱ ثانیه         ای
gpio_set_level(BLINK_GPIO, 0);  //  خاموش کردنLED         vTaskDelay(1000 / portTICK_PERIOD_MS);  //  تاخیر
ای ۱ ثانیه} }`
3.
```

ساخت و فلش کردن پروژه

اتصال ESP32 خود را برقرار کنید و دستورات زیر را اجرا کنید:

```
1. bash
2. CopyEdit
3. `idf.py set-target esp32 idf.py build idf.py flash idf.py monitor
4.
```

`

برای خروج از مانیتور، از ترکیب کلیدهای Ctrl+] استفاده کنید.

مزایای ESP-IDF برای پروژه‌های پیشرفته ESP32

1. دسترسی سطح پایین:
   امکان کنترل مستقیم امکانات ESP32 برای عملکرد دقیق و بهینه.

2. پشتیبانی از RTOS:
   ساخت برنامه‌هایی با محدودیت‌های زمان واقعی و چندوظیفه‌ای.

3. قابلیت سفارشی‌سازی:
   تنظیم ویژگی‌های سخت‌افزاری و نرم‌افزاری (مثل تخصیص حافظه و پروتکل‌های ارتباطی).

4. ابزارهای دیباگینگ کامل:
   شامل دیباگ با JTAG، GDB و سیستم‌های لاگ.

5. مناسب برای تولید صنعتی:
   کنترل دقیق ویژگی‌ها باعث ارائه راه‌حل‌های بهینه و مقیاس‌پذیر برای سیستم‌های واقعی IoT می‌شود.

کاربردهای ESP-IDF

- سیستم‌های اتوماسیون خانگی.
- دستگاه‌های مجهز به AI و ML (مانند استفاده از TensorFlow Lite).
- پروژه‌های ارتباط بی‌سیم (مانند BLE، Wi-Fi و سیستم‌های مبتنی بر MQTT).
- سیستم‌های نظارت و کنترل زمان واقعی.

Arduino

هنگام مقایسه با سایر پلتفرم‌ها، Arduino IDE برای مبتدیان کاربرپسندترین گزینه است. اگرچه ممکن است برای کار با ESP32 ایده‌آل نباشد، اما بسیاری از افراد با این برنامه آشنا هستند که شروع کار را بسیار آسان‌تر می‌کند.

قبل از اینکه بتوانید از Arduino IDE برای برنامه‌نویسی ESP32 استفاده کنید، ابتدا باید برد ESP32 (همچنین به عنوان ESP32 Arduino Core شناخته می‌شود) را از طریق Arduino Board Manager نصب کنید. این راهنما شما را در مراحل دانلود، نصب و تست ESP32 Arduino Core راهنمایی می‌کند.

Core چیست؟

Coreها برای سازگار کردن میکروکنترلرهای جدید با Arduino IDE و کتابخانه‌ها و اسکچ‌های موجود ضروری هستند.

- Arduino خودش Coreهایی را برای میکروکنترلرهای Atmel AVR که در بردهای خود استفاده می‌کند، توسعه داده است.

- با این حال، هر کسی می‌تواند برای بردهای خود Core بسازد، به شرطی که قوانین و الزامات تعیین شده توسط Arduino را رعایت کند.

Boards Manager

برخی از بردهای توسعه نیاز به نصب یک Core اضافی دارند. به همین دلیل، Arduino ابزاری به نام Boards Manager توسعه داده است که امکان افزودن Coreها به Arduino IDE را فراهم می‌کند.

مرحله 1: نصب یا به‌روزرسانی Arduino IDE

اولین گام برای نصب ESP32 Arduino Core این است که مطمئن شوید آخرین نسخه Arduino IDE روی کامپیوتر شما نصب شده است.

لینک دانلود Arduino IDE

مرحله 2: نصب درایور USB-to-Serial Bridge

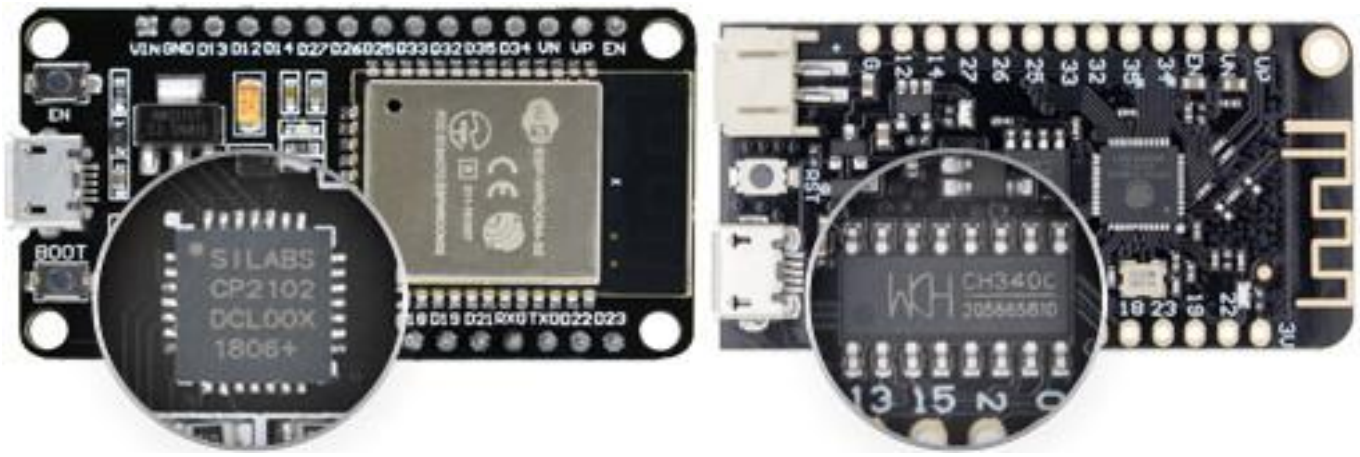
بسیاری از بردهای توسعه مبتنی بر ESP32 موجود هستند و بسته به طراحی برد، ممکن است نیاز به نصب درایورهای اضافی برای مبدل USB به سریال داشته باشید تا بتوانید کدها را روی ESP32 آپلود کنید.

مثال‌ها:

- ESP32 DevKit V1: از CP2102 برای تبدیل سیگنال‌های USB به UART استفاده می‌کند.

- WeMos ESP32 Lite: از CH340G بهره می‌برد.

- ESP32-CAM: فاقد مبدل USB به سریال داخلی است و نیاز به یک ماژول جداگانه دارد.



نکته مهم:

برد خود را با دقت بررسی کنید تا نوع مبدل USB-to-Serial را شناسایی کنید. معمولاً CP2102 یا CH340 در برد استفاده شده است.

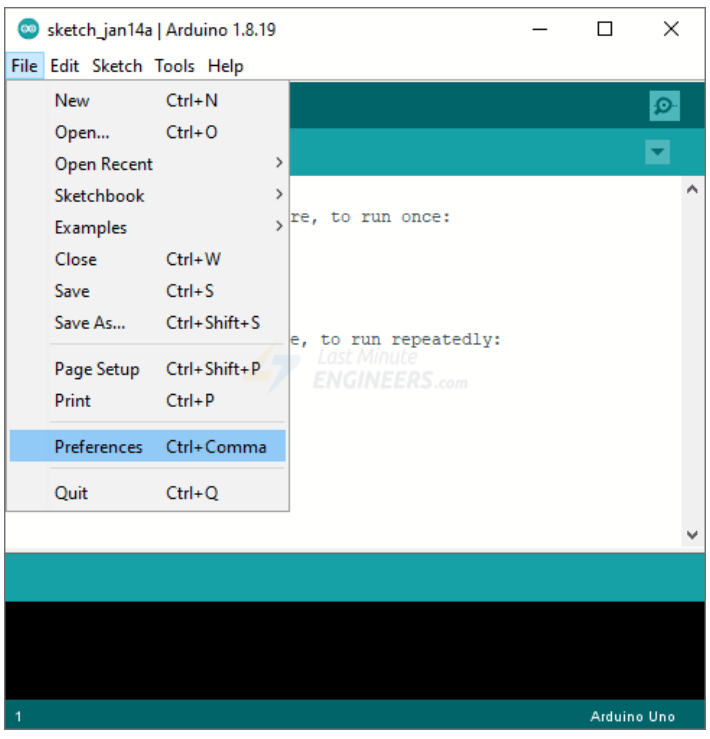
مرحله 3: نصب ESP32 Arduino Core

1. باز کردن Arduino IDE:

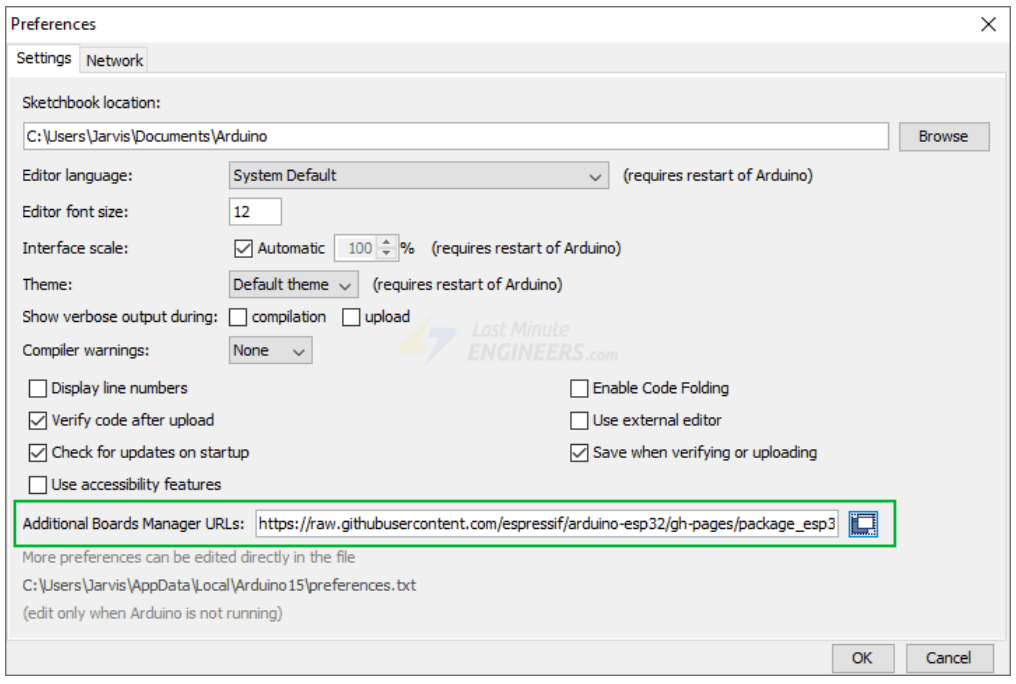
- به File > Preferences بروید.

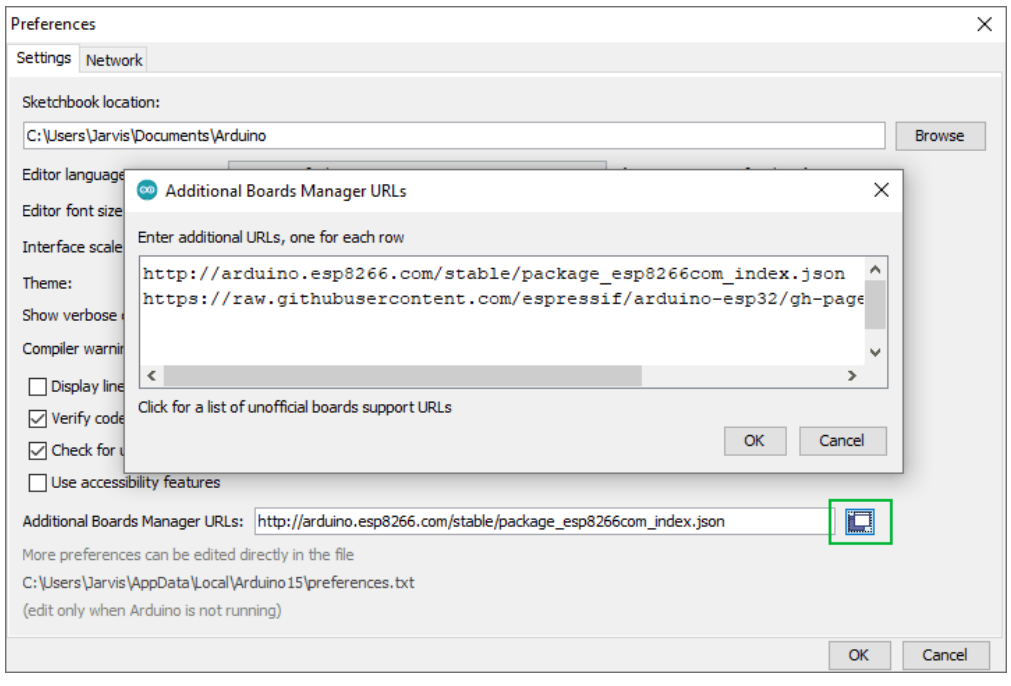
- در بخش Additional Board Manager URLs، آدرس زیر را وارد کنید:

`https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json`



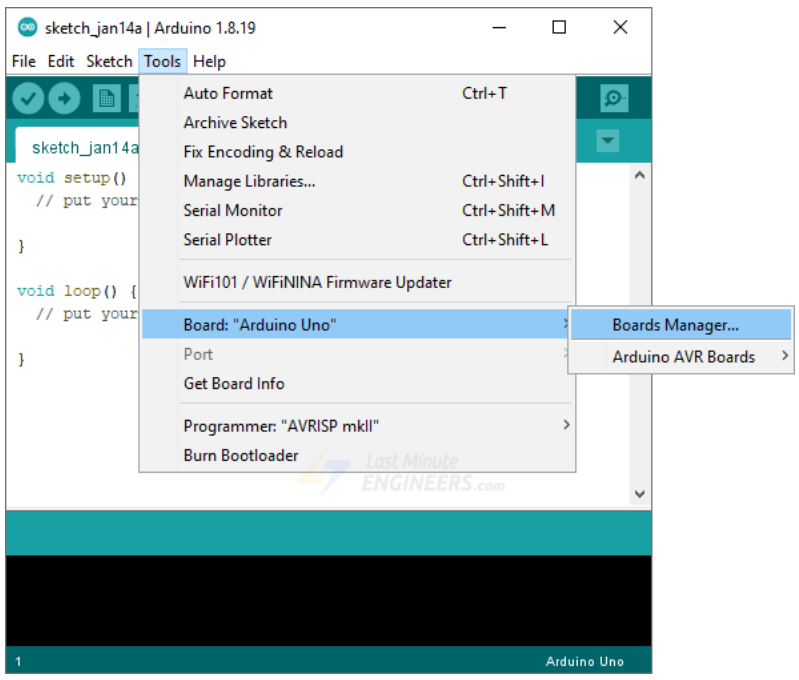
اگر قبلاً لینک‌های دیگری (مثل ESP8266) وارد کرده‌اید، روی آیکون کنار فیلد کلیک کنید و لینک‌های جدید را در سطرهای جداگانه اضافه کنید.

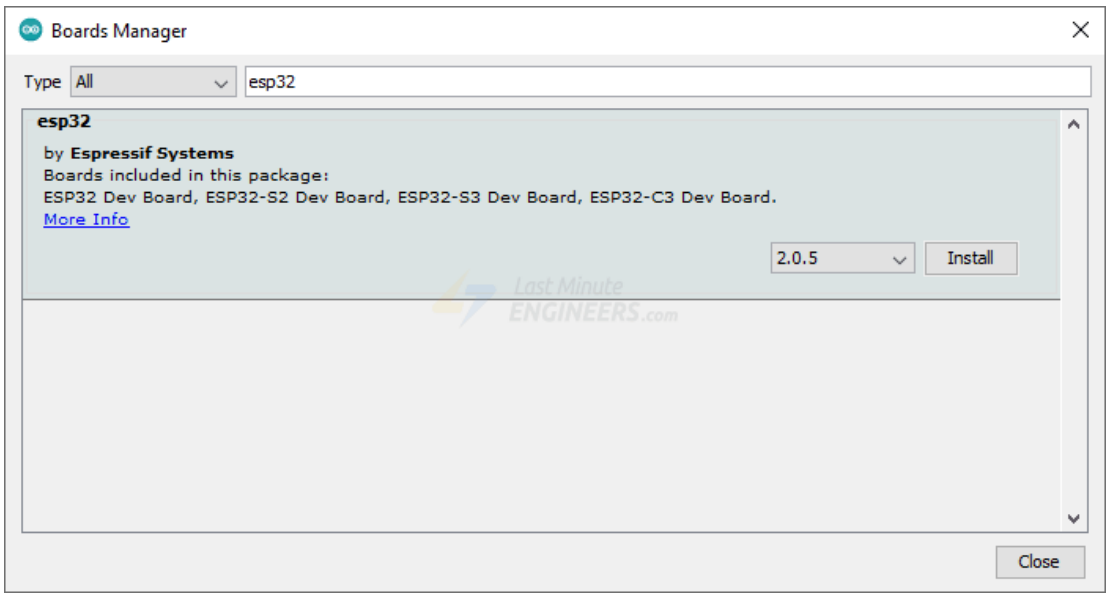




افزودن Core ESP32:

- به Tools > Board > Boards Manager بروید.
- در قسمت جستجو، عبارت esp32 را تایپ کنید.
- گزینه ESP32 by Espressif Systems را پیدا کرده و روی Install کلیک کنید.





مرحله 4: انتخاب برد و پورت

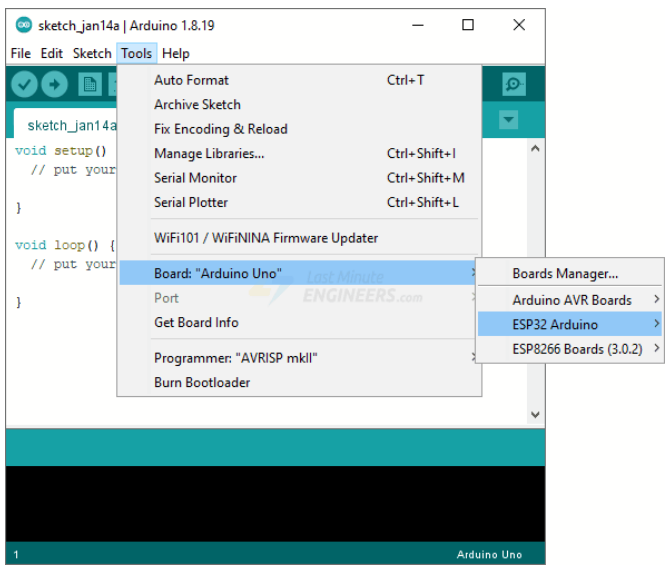
1. انتخاب برد:

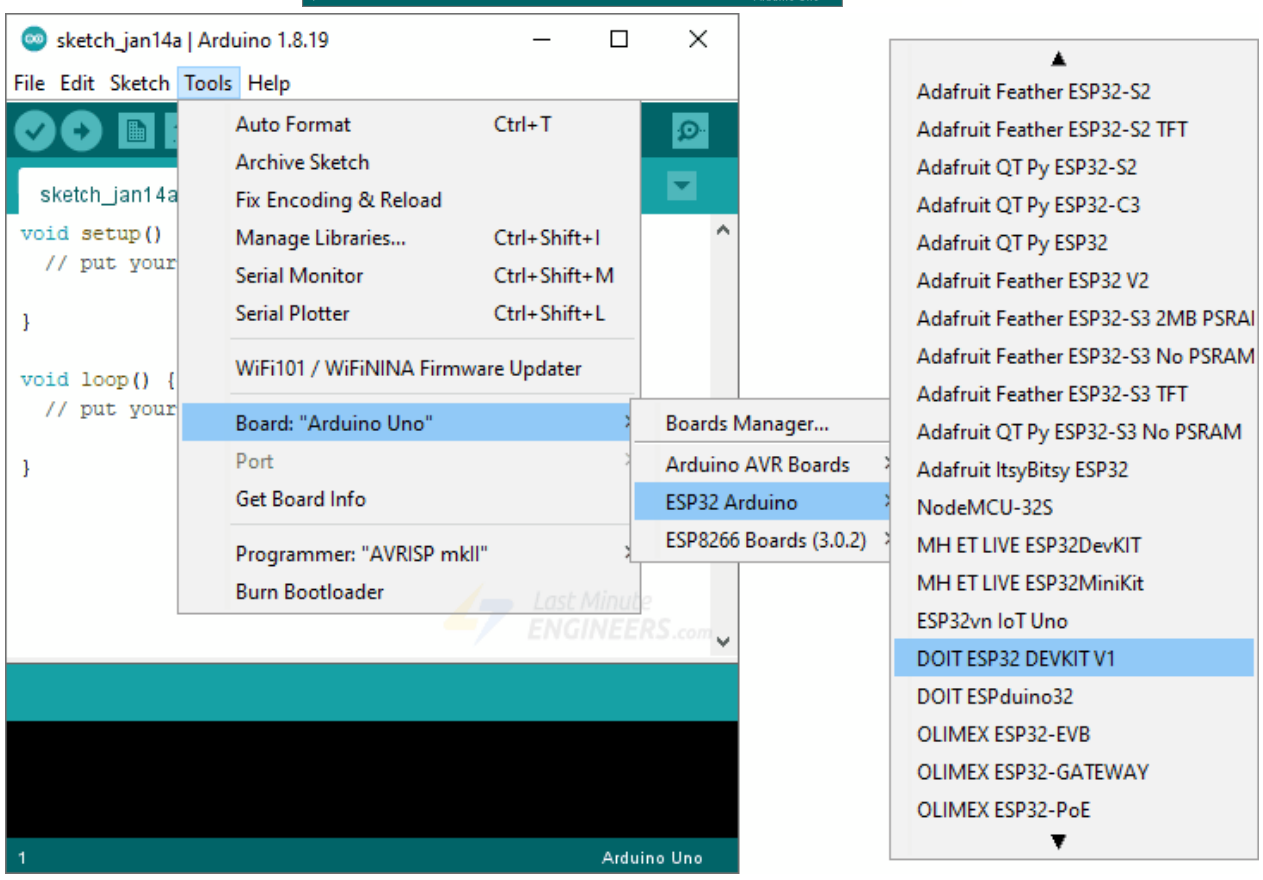
- پس از نصب Core، Arduino IDE را مجدداً راه‌اندازی کنید.
- به Tools > Board بروید و برد ESP32 خود را انتخاب کنید.

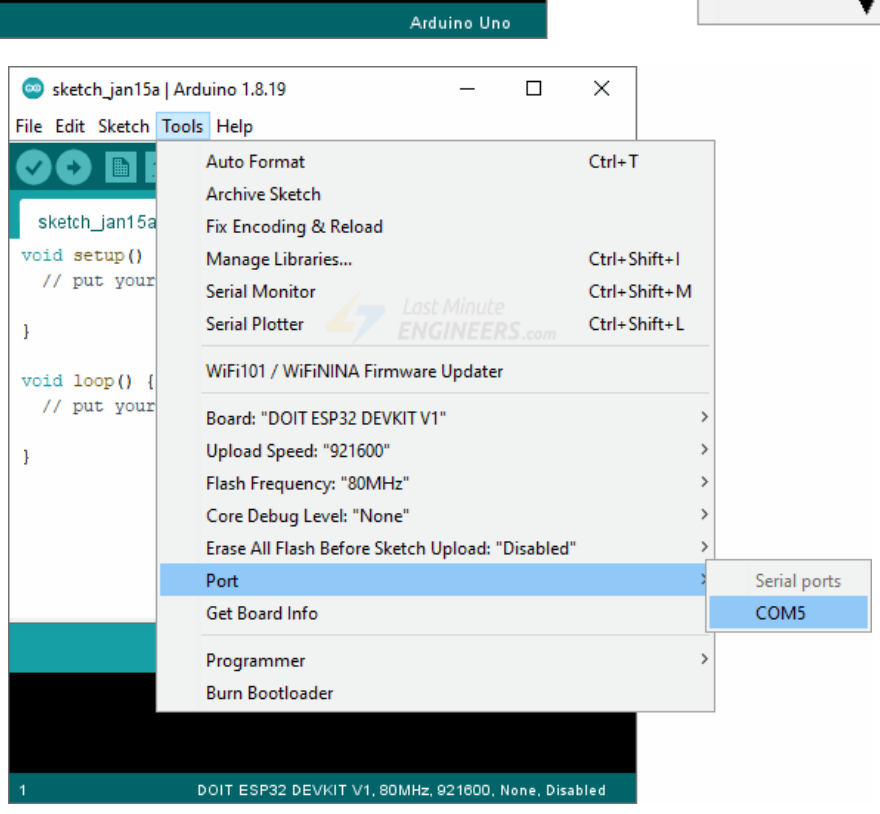
اگر برد خود را نمی‌شناسید، می‌توانید ESP32 Dev Module را انتخاب کنید.

2. انتخاب پورت:

- برد ESP32 خود را به کامپیوتر متصل کنید و از Tools > Port پورت صحیح را انتخاب کنید.







به‌روزرسانی ESP32 Arduino Core

همواره اطمینان حاصل کنید که آخرین نسخه ESP32 Arduino Core را نصب کرده‌اید:

-به Tools > Board > Boards Manager بروید.

-عبارت ESP32 را جستجو کنید.

-نسخه نصب‌شده را بررسی کنید. اگر نسخه جدیدتری موجود است، آن را نصب کنید.

اکنون می‌توانید شروع به نوشتن کد برای برد ESP32 خود در Arduino IDE کنید.

##مرحله ۵: آزمایش نصب

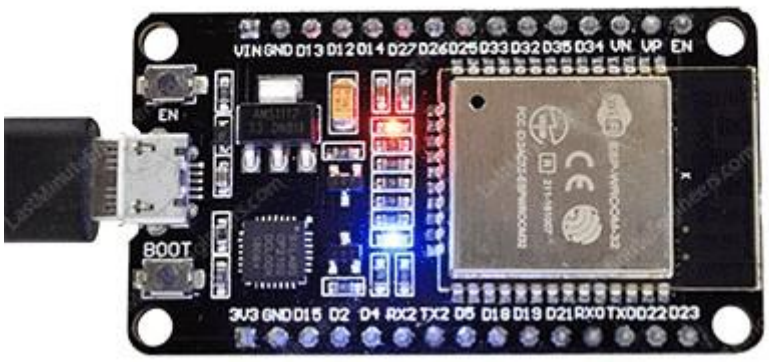
پس از اتمام مراحل قبلی، آماده‌اید تا اولین برنامه خود را با ESP32 آزمایش کنید Arduino IDE !را باز کنید. اگر برد خود را جدا کرده‌اید، دوباره آن را وصل کنید.

بیایید ساده‌ترین برنامه ممکن را آپلود کنیم – برنامه چشمک‌زن!(Blink)

این برنامه از LED داخلی که روی بیشتر بردهای توسعه ESP32 وجود دارد استفاده می‌کند. این LED به پین دیجیتال D2 متصل است، اما شماره آن ممکن است در بردهای مختلف متفاوت باشد.

```
1. cpp
2. CopyEdit
3. `int ledPin = 2;  void setup() {    pinMode(ledPin, OUTPUT); }  void loop() {       digitalWrite(ledPin, HIGH);   delay(500);
        digitalWrite(ledPin, LOW);    delay(500); }`
4.
```

اگر همه چیز به درستی انجام شده باشد، LEDداخلی برد ESP32 شما اکنون باید چشمک بزند! برای اجرای برنامه، ممکن است لازم باشد دکمه EN روی ESP32 خود را فشار دهید.



Tenserflow

مقدمه‌ای کوتاه درباره TensorFlow

TensorFlow یک فریمورک متن‌باز برای یادگیری ماشین است که توسط شرکت گوگل توسعه یافته است. این فریمورک به طور گسترده برای ایجاد، آموزش، و پیاده‌سازی مدل‌های یادگیری ماشین (ML) و یادگیری عمیق (DL) مورد استفاده قرار می‌گیرد. TensorFlow بسیار انعطاف‌پذیر بوده و از وظایف مختلفی مانند شناسایی تصویر، پردازش زبان طبیعی، و تحلیل سری‌های زمانی پشتیبانی می‌کند.

ویژگی‌های کلیدی عبارت‌اند از:

1. مقیاس‌پذیری: سازگار با سخت‌افزارهای مختلف، از جمله CPU، GPU، و TPU.

2. سهولت در پیاده‌سازی: امکان استقرار مدل‌ها روی سرورهای ابری، وب، دستگاه‌های موبایل یا حتی میکروکنترلرها.

3. TensorFlow Lite (TFLite): نسخه‌ای سبک و بهینه‌سازی‌شده برای دستگاه‌های لبه (Edge Devices)، از جمله میکروکنترلرها.

Edge Impulse: A Platform for Machine Learning on Edge Devices

Edge Impulse یک پلتفرم طراحی‌شده برای ساده‌سازی فرآیند ایجاد، آموزش و استقرار مدل‌های یادگیری ماشین (ML) به طور خاص برای دستگاه‌های لبه (Edge Devices) است. این پلتفرم برای میکروکنترلرها، دستگاه‌های اینترنت اشیا (IoT) و دیگر سخت‌افزارهای با منابع محدود مانند ESP32 بسیار مناسب است. Edge Impulse برای مبتدیان بسیار کاربرپسند است و یک رابط کاربری مبتنی بر وب برای جمع‌آوری داده‌ها، آموزش مدل‌ها و استقرار آن‌ها فراهم می‌کند.

ویژگی‌های کلیدی:

1. جمع‌آوری و برچسب‌گذاری داده‌ها: داده‌ها را به صورت مستقیم از دستگاه خود جمع‌آوری کرده یا از طریق پلتفرم آپلود کنید.

2. آموزش مدل خودکار: مدل‌های یادگیری ماشین را با رابط کاربری ساده آموزش داده و به لوله‌های پیش‌پیکربندی‌شده برای داده‌های تصویری، صوتی و حسگرها دسترسی پیدا کنید.

3. استقرار مدل: مدل‌هایی که به‌طور ویژه برای دستگاه‌های لبه بهینه شده‌اند را صادر کنید، از جمله TensorFlow Lite برای میکروکنترلرها و کد C++.

4. سازگاری با سخت‌افزار: از ESP32، Arduino، Raspberry Pi، STM32 و دیگر سخت‌افزارها پشتیبانی می‌کند.

استفاده از Edge Impulse با میکروکنترلرها (مثلاً ESP32)

Edge Impulse فرآیند کامل یادگیری ماشین را ساده می‌کند، به‌طوری‌که حتی برای مبتدیان با تجربه کدنویسی حداقل هم قابل استفاده است. در اینجا نحوه استفاده از آن با ESP32 آورده شده است:

جریان کار با Edge Impulse:

1. جمع‌آوری داده‌ها:

- جمع‌آوری داده‌های دنیای واقعی از حسگرها (دوربین، میکروفن، شتاب‌سنج و غیره) مستقیماً با استفاده از ابزارهای Edge Impulse.
- استفاده از Edge Impulse Data Forwarder برای پخش داده‌ها از ESP32 به پلتفرم برای جمع‌آوری داده‌ها.

مثال (با استفاده از CLI):

```
   1. bash
2.     Copy
3.     `edge-impulse-data-forwarder
4.
```

`

- منابع ورودی مانند مسافت فراصوت، تصاویر یا سیگنال‌های صوتی را مشخص کنید.

2. پیش‌پردازش داده‌ها و برچسب‌گذاری:

- پیش‌پردازش داده‌ها مستقیماً روی پلتفرم (مانند نرمال‌سازی تصاویر، تقسیم سیگنال‌های صوتی).
- برچسب‌گذاری داده‌ها برای آموزش (مثلاً "open_hand" برای شناسایی حرکت دست یا "clap" برای شناسایی صدا).

3. آموزش مدل:

- انتخاب از بلوک‌های یادگیری پیش‌ساخته (مانند طبقه‌بندی تصویر، شناسایی صوت یا رگرسیون).
- سفارشی‌سازی پارامترهای شبکه عصبی یا استفاده از تنظیمات پیش‌فرض برای سادگی.

پلتفرم یک رابط بصری برای:

- نظارت بر پیشرفت آموزش.
- بازخورد لحظه‌ای از عملکرد مدل (مانند دقت و خطا).

4. آزمایش مدل:

- آزمایش مدل آموزش‌داده‌شده با داده‌های اعتبارسنجی.
- انجام آزمایش‌های زنده با دستگاه متصل برای مشاهده عملکرد در دنیای واقعی.

5. استقرار مدل:

- صادرات مدل در فرمت‌های مختلف:

- TensorFlow Lite برای میکروکنترلرها (TFLM).
- کتابخانه Arduino.
- کد C++ برای ادغام مستقیم.

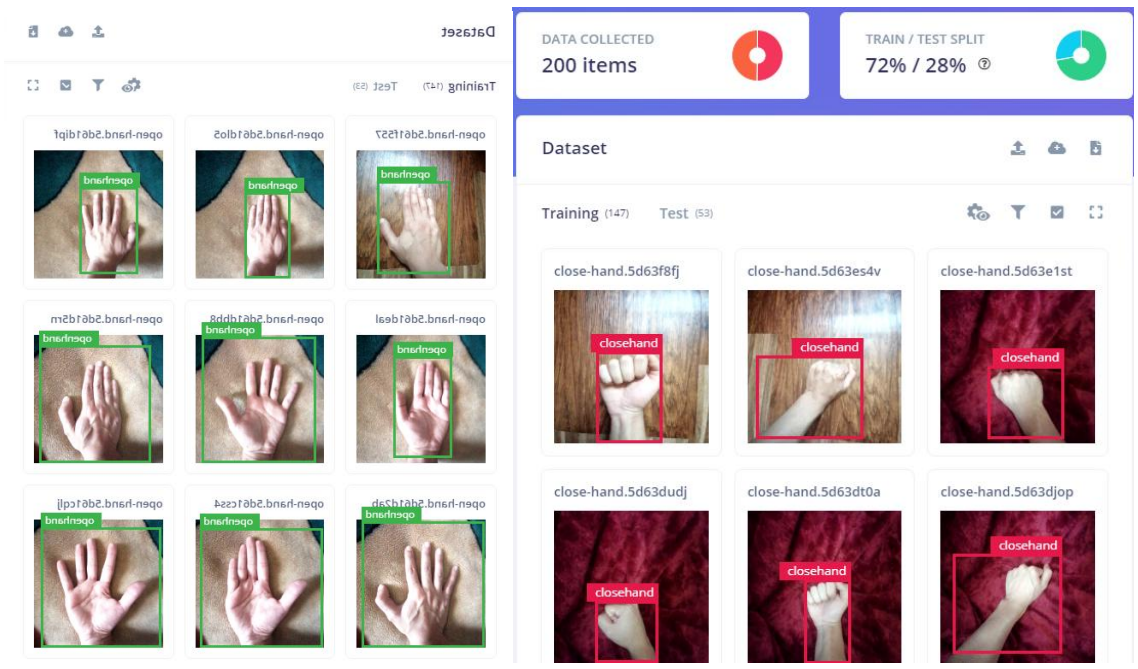
فرمت‌های استقرار نمونه:

- برای ESP32، معمولاً از مدل TensorFlow Lite یا کتابخانه Arduino استفاده می‌شود.'

شناسایی حرکت دست (Gesture Recognition) بر روی ESP32

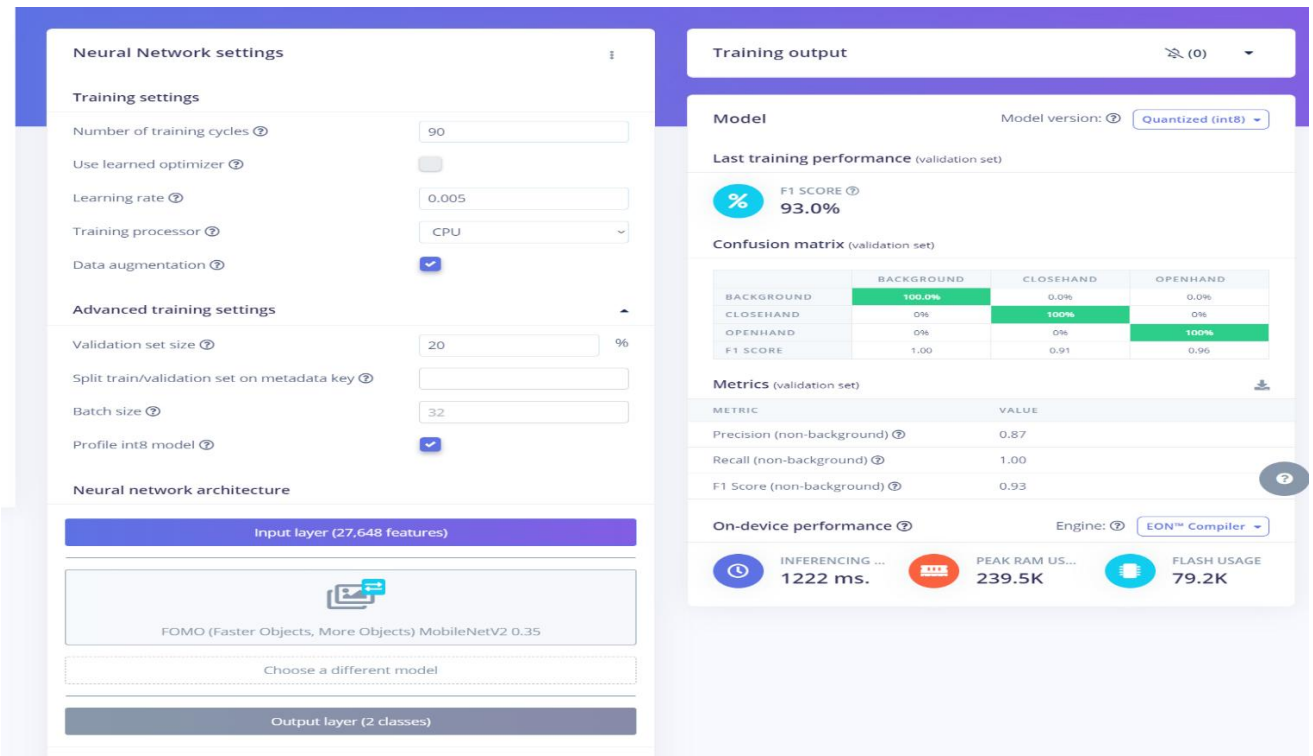
1. جمع‌آوری داده‌ها:

- تصاویر از حرکات دست (مثلاً "open_hand" و "closed_fist") را با استفاده از ESP32-CAM جمع‌آوری کنید.
- از Edge Impulse Studio برای آپلود یا ضبط این تصاویر استفاده کنید.



2. آموزش مدل:

- بلوک طبقه‌بندی تصویر را در Edge Impulse انتخاب کنید.
- مدل را برای شناسایی دو حرکت دست آموزش دهید.



3. صادرات مدل:

- مدل آموزش‌داده‌شده را به صورت کتابخانه Arduino صادر کنید:

- به استقرار در Edge Impulse Studio بروید.

- "کتابخانه Arduino" را انتخاب کرده و فایل `.zip` را دانلود کنید.

4. ادغام با ESP32:

- کتابخانه را در Arduino IDE وارد کنید.

- کد بنویسید تا از مدل برای استنتاج استفاده کرده و اقداماتی مانند حرکت دادن موتور سروو را انجام دهید.

```
#include <BEESHAH-project-1_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"
#include <ESP32Servo.h>
#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here:
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/CameraWebServer/camera_pins.h

// #define CAMERA_MODEL_ESP_EYE // Uncomment for ESP-EYE model
#define CAMERA_MODEL_AI_THINKER // Uncomment for AI Thinker model

#if defined(CAMERA_MODEL_ESP_EYE)
    #define PWDN_GPIO_NUM   -1
    #define RESET_GPIO_NUM  -1
    #define XCLK_GPIO_NUM    4
    #define SIOD_GPIO_NUM   18
    #define SIOC_GPIO_NUM   23
    #define Y9_GPIO_NUM     36
    #define Y8_GPIO_NUM     37
    #define Y7_GPIO_NUM     38
    #define Y6_GPIO_NUM     39
    #define Y5_GPIO_NUM     35
    #define Y4_GPIO_NUM     14
    #define Y3_GPIO_NUM     13
    #define Y2_GPIO_NUM     34
    #define VSYNC_GPIO_NUM   5
    #define HREF_GPIO_NUM   27
    #define PCLK_GPIO_NUM   25
#elif defined(CAMERA_MODEL_AI_THINKER)
    #define PWDN_GPIO_NUM   32
    #define RESET_GPIO_NUM  -1
    #define XCLK_GPIO_NUM    0
    #define SIOD_GPIO_NUM   26
    #define SIOC_GPIO_NUM   27
    #define Y9_GPIO_NUM     35
    #define Y8_GPIO_NUM     34
    #define Y7_GPIO_NUM     39
    #define Y6_GPIO_NUM     36
    #define Y5_GPIO_NUM     21
    #define Y4_GPIO_NUM     19
    #define Y3_GPIO_NUM     18
    #define Y2_GPIO_NUM      5
    #define VSYNC_GPIO_NUM  25
    #define HREF_GPIO_NUM   23
    #define PCLK_GPIO_NUM   22
#else
    #error "Camera model not selected"
#endif

// Constants
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS   320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS   240
#define EI_CAMERA_FRAME_BYTE_SIZE         3

// Variables
static bool debug_nn = false; // Enable debug info
static bool is_initialised = false;
uint8_t snapshot_buf = nullptr; // Points to the output of the capture

static camera_config_t camera_config = {
    .pin_pwdn = PWDN_GPIO_NUM,
    .pin_reset = RESET_GPIO_NUM,
    .pin_xclk = XCLK_GPIO_NUM,
    .pin_sscb_sda = SIOD_GPIO_NUM,
    .pin_sscb_scl = SIOC_GPIO_NUM,
    .pin_d7 = Y9_GPIO_NUM,
    .pin_d6 = Y8_GPIO_NUM,
    .pin_d5 = Y7_GPIO_NUM,
    .pin_d4 = Y6_GPIO_NUM,
    .pin_d3 = Y5_GPIO_NUM,
    .pin_d2 = Y4_GPIO_NUM,
    .pin_d1 = Y3_GPIO_NUM,
    .pin_d0 = Y2_GPIO_NUM,
    .pin_vsync = VSYNC_GPIO_NUM,
    .pin_href = HREF_GPIO_NUM,
    .pin_pclk = PCLK_GPIO_NUM,
    .xclk_freq_hz = 20000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,
    .pixel_format = PIXFORMAT_JPEG,
    .frame_size = FRAMESIZE_QVGA,
    .jpeg_quality = 12,
    .fb_count = 1,
```

```
    .fb_location = CAMERA_FB_IN_PSRAM,
    .grab_mode = CAMERA_GRAB_WHEN_EMPTY
};

// Servo setup
Servo myservo;
int servoPin = 15;
int currentServoPos = 0;

// Function to move the servo
void moveServoToPosition(int targetPos) {
    myservo.write(targetPos);
    delay(500);
    currentServoPos = targetPos;
}

// Function declarations
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t out_buf);

/
  Arduino setup function
 /
void setup() {
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);
    myservo.attach(servoPin, 1000, 2000);
    moveServoToPosition(currentServoPos);

    Serial.begin(115200);
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    if (!ei_camera_init()) {
        ei_printf("Failed to initialize Camera!\r\n");
    } else {
        ei_printf("Camera initialized\r\n");
    }

    ei_printf("\nStarting continuous inference in 2 seconds...\n");
    ei_sleep(2000);
}

/
  Main loop for inferencing and servo control
 /
void loop() {
    if (ei_sleep(5) != EI_IMPULSE_OK) return;

    snapshot_buf = (uint8_t)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS  EI_CAMERA_RAW_FRAME_BUFFER_ROWS  EI_CAMERA_FRAME_BYTE_SIZE);
    if (snapshot_buf == nullptr) {
        ei_printf("ERR: Failed to allocate snapshot buffer!\n");
        return;
    }

    ei::signal_t signal;
    signal.total_length = EI_CLASSIFIER_INPUT_WIDTH  EI_CLASSIFIER_INPUT_HEIGHT;
    signal.get_data = &ei_camera_get_data;

    if (!ei_camera_capture(EI_CLASSIFIER_INPUT_WIDTH, EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf)) {
        ei_printf("Failed to capture image\r\n");
        free(snapshot_buf);
        return;
    }

    ei_impulse_result_t result = { 0 };
    EI_IMPULSE_ERROR err = run_classifier(&signal, &result, debug_nn);

    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        free(snapshot_buf);
        return;
    }

    bool close_hand_detected = false;
    bool open_hand_detected = false;

    for (size_t i = 0; i < result.bounding_boxes_count; i++) {
        auto bbox = result.bounding_boxes[i];
        ei_printf("Detected gestures: %s (%.2f) [x: %d, y: %d, width: %d, height: %d]\n",
                  bbox.label, bbox.value, bbox.x, bbox.y, bbox.width, bbox.height);

        if (strcmp(bbox.label, "closehand") == 0 && bbox.value > 0.8) {
            close_hand_detected = true;
        } else if (strcmp(bbox.label, "openhand") == 0 && bbox.value > 0.6) {
            open_hand_detected = true;
        }
    }

    if (open_hand_detected && currentServoPos != 180) {
        ei_printf("Detected: Open Hand\n");
        moveServoToPosition(180);
        ei_printf("Servo moved to open position (180 degrees).\n");
    } else if (close_hand_detected && currentServoPos != 0) {
        ei_printf("Detected: Close Fist\n");
        moveServoToPosition(0);
        ei_printf("Servo moved to closed position (0 degrees).\n");
    } else {
        ei_printf("No clear gesture detected.\n");
    }

```

```
189.     free(snapshot_buf);
190. }
191.
```

استقرار و آزمایش:

- ESP32 را با فریمور تولیدشده فلش کنید.

- با انجام حرکات دست آزمایش کنید و پاسخ موتور سروو را مشاهده کنید.

EON™ Compiler

Same accuracy, 17% less RAM, 29% less ROM.

3)

	IMAGE	G DETECTION	TOTAL
LATENCY	15 ms.	1,222 ms.	**1,237 ms.**
RAM	4.0K	239.5K	**239.5K**
FLASH	-	79.2K	-
ACCURACY			-

مزایای Edge Impulse برای ESP32:

- بدون نیاز به کدنویسی برای آموزش: رابط بصری نیاز به اسکریپت‌نویسی پیچیده را از بین می‌برد.

- مدل‌های بهینه‌سازی‌شده: مدل‌هایی بسیار کارآمد و مناسب برای دستگاه‌های با منابع محدود تولید می‌کند.

- سازگاری سخت‌افزاری: با طیف گسترده‌ای از میکروکنترلرها سازگار است.