

الحمد لله رب العالمين



دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش کارآموزی

گرایش سیستم های دیجیتال

عنوان:

گزارش کارآموزی

نگارش:

رضا رمضان پور

استاد کارآموزی:

دکتر نرجس الهدی محمد زاده

۱۱ شهریور ۱۳۹۸

## چکیده:

استارتاپ «وینگ» زیرمجموعه ای از شرکت رایمون است و هدف آن تولید دستگاه های حمل و نقل عمومی و برقی جهت کاهش آلودگی شهر است. این مجموعه تلاش دارد با کنترل کردن دستگاه های آما ای که شرکت های خوب دنیا برای استفاده عمومی ساختند، این بستر را در ایران نیز فراهم آورد تا مردم بتوانند برای جابجایی از آن ها استفاده کنند.

اینکار یک زیر مجموعه از اینترنت اشیاست، به این دلیل که دستگاه در لحظه پیام هایی را برای سرور تحت کنترل ارسال می کند و نیز سرور هم با آن تعامل دارد. اینترنت اشیا، به طور کلی اشاره دارد به بسیاری از چیزها شامل اشیا و وسایل محیط پیرامون که به شبکه اینترنت متصل شده و توسط اپلیکیشن های موجود در تلفن های هوشمند و تبلت قابل کنترل و مدیریت هستند.

در فصل اول به اختصار به توضیح در مورد محیط شرکت خواهم پرداخت و استارتاپ هایی که در آن حضور و فعالیت دارند.

در فصل دوم به توضیح فعالیت خودم در استارتاپ «وینگ» خواهم پرداخت و به بررسی اهمیت کار خود در آن استارتاپ می پردازم و نهایتاً در فصل آخر نتیجه گیری خود را از این کارآموزی بیان خواهم کرد.

## فهرست

۱ فصل اول: آشنایی با شرکت.....	۱
۱-۱ آشنایی با شرکت .....	۱
۲-۱ ساختار شرکت .....	۲
۳-۱ استارتاپ وینگ .....	۳
۲ فصل دوم: فعالیت های انجام شده در کارآموزی .....	۴
۱-۲ مقدمه.....	۴
۲-۲ معرفی میکروکنترلر ESP32 .....	۵
۲-۳ توضیح در مورد Platform Io IDE .....	۸
۲-۴ ESP IOT Development Framework .....	۱۰
۲-۵ Over-the-air Programming .....	۱۱
۲-۶ Message Queuing Telemetry Transport .....	۱۷
۲-۷ پخش صدا .....	۲۵
۲-۸ MPU9250 .....	۲۸
۲-۹ کارهای جانبی .....	۳۲
۲-۹-۱ راه اندازی GPS Neo 6m .....	۳۲

۲-۹-۲ تحقیق در مورد نحوه پیاده سازی webgl بر روی esp8266 ..... ۳۳

۲-۹-۳ Tensorflow Lite ..... ۳۴

۲-۱۰ جمع بندی ..... ۳۵

۳ فصل سوم: جمع بندی، نتیجه گیری و پیشنهادها ..... ۳۶

۳-۱ جمع بندی ..... ۳۶

۳-۲ نتیجه گیری ..... ۳۶

۳-۳ پیشنهادها ..... ۳۷

چند عکس از توضیحات پروژه ..... ۳۸

مراجع ..... ۴۰

## فهرست شکل ها

- شکل ۱-۱: تصویری از شرکت [1]..... ۱
- شکل ۲-۱: یخچال باینو [2]..... ۲
- شکل ۳-۱: اسکوتر برقی وینگ [3]..... ۳
- شکل ۱-۲: گذرگاه های ارتباطی [5]..... ۶
- شکل ۲-۲: میکروکنترلر ESP32..... ۶
- شکل ۳-۲: برد آموزشی..... ۷
- شکل ۴-۲: نماد Platform Io..... ۸
- شکل ۵-۲: سرعت build شدن کد روی IDE های مختلف [6]..... ۹
- شکل ۶-۲: نماد Espressif..... ۱۰
- شکل ۷-۲: single mode partition [8]..... ۱۲
- شکل ۸-۲: ota mode partition [8]..... ۱۳
- شکل ۹-۲: نحوه انجام ota [9]..... ۱۴
- شکل ۱۰-۲: اتصال به ESP32..... ۱۶
- شکل ۱۱-۲: مثالی از ارسال پیام [11]..... ۱۸
- شکل ۱۲-۲: QoS0..... ۱۹
- شکل ۱۳-۲: QoS1..... ۲۰

- شکل ۲-۱۴: QoS2 ..... ۲۱
- شکل ۲-۱۵: Hive Broker ..... ۲۲
- شکل ۲-۱۶: Eclipse Broker ..... ۲۲
- شکل ۲-۱۷: Mosquitto Broker ..... ۲۲
- شکل ۲-۱۸: محیط برنامه MQTTBOX ..... ۲۳
- شکل ۲-۱۹: یافتن certificate ..... ۲۴
- شکل ۲-۲۰: پارتیشن جدید حافظه فلش ..... ۲۵
- شکل ۲-۲۱: مراحل ویرایش فایل صوتی [12] ..... ۲۶
- شکل ۲-۲۲: تصویری از فایل صدای ذخیره شده ..... ۲۶
- شکل ۲-۲۳: عکس ماژول امپلی فایر [13] ..... ۲۷
- شکل ۲-۲۴: ویژگی های MPU9250 ..... ۲۸
- شکل ۲-۲۵: نشان دادن محور چرخش ها [15] ..... ۲۹
- شکل ۲-۲۶: MPU9250 ..... ۳۱
- شکل ۲-۲۷: ماژول GPS NEO 6m ..... ۳۲
- شکل ۲-۲۸: یک مثال از خروجی ماژول GPS [16] ..... ۳۳
- شکل ۲-۲۹: نماد Tensorflow Lite ..... ۳۴
- شکل ۲-۳۰: استفاده از Tensorflow در گوشی ..... ۳۴

## ۱ فصل اول: آشنایی با شرکت رایمون (هاردتک)

### ۱-۱ معرفی شرکت

این شرکت واقع در ارتش غرب است و دسترسی به آن برای من که در خوابگاه طرشت ۳ سکونت داشتم یک مقدار سخت بود.

البته این شرکت محلی برای اقامت در شب دارد و همچنین تمام نیازها برای زندگی در آنجا را رفع می‌کند به همین دلیل بنده نیز چند شب در آنجا سکونت داشتم.

این شرکت در باغ بزرگی وابسته به بنیاد برکت واقع است.

در زیر تصویری از این شرکت را مشاهده می‌کنید:



۱-۱ تصویری از شرکت [1]



## ۲-۱ ساختار شرکت

در این شرکت استارتاپ ها به دو بخش استارتاپ های نرم افزاری و سخت افزاری تقسیم می شوند.

ساختار این استارتاپ ها به این صورت است که یا پروژه از خارج از هاردتک دریافت و آن را انجام می دهند و یا بر اساس نیاز جامعه پروژه ای برای خود تعریف می کنند و آن را انجام می دهند.

استارتاپ ها در این مجموعه موظف به حضور ۶۰ ساعت در هفته در این مجموعه و فعالیت خالص در آن هستند. یکی از استارتاپ های موفق این شرکت، استارتاپ باینو هست که کار این مجموعه تولید یخچال هایی برای عرضه مواد غذایی تازه در اداره ها است.

نحوه کار این یخچال ها بدین صورت است که با قرار دادن کارت بر روی قسمت کارت خوان قادر به باز کردن در آن خواهید بود و زمانی که در آن بسته شد، پول از حساب شما کسر خواهد شد. نحوه محاسبه مبلغ کسر شده نیز بسیار ساده است، به این صورت که این یخچال به بخش هایی تقسیم شده است که در هر بخش یک ترازو قرار دارد و وقتی یک ماده غذایی از داخل آن برداشته شود به کمک آن ترازوها مبلغ خرید محاسبه می شود.



۲-۱ یخچال باینو [2]

### ۳-۱ استارتاپ وینگ

بخشی که این جانب در آن فعالیت می‌کردم، استارتاپ سخت‌افزاری «وینگ» نام دارد. وظیفه این استارتاپ کنترل کردن اسکوتر برقی به‌منظور استفاده عمومی افراد و کمک به کاهش استفاده از وسایلی که باعث آلودگی هوا می‌شوند، است.

اعضای این استارتاپ، شامل آقای شایان شریفی، حسن فراهانی و امیررضا فتحی دانشجویان دانشگاه صنعتی شریف هستند. این استارتاپ در حال انجام دادن تست گیری نهایی است، سپس یک تست گیری عمومی در دانشگاه امیرکبیر خواهد داشت.

سایت این استارتاپ طراحی شده است و هم‌اکنون به نشانی [wingco.ir](http://wingco.ir) قابل استفاده است.

عکسی از امیررضا فتحی و اسکوتر این استارتاپ را در زیر می‌بینید:



۳-۱ اسکوتر برقی وینگ [3]

## ۲ فصل دوم: فعالیت‌های انجام‌شده در کارآموزی

### ۲-۱ مقدمه

همان‌طور که می‌دانید یکی از معروف‌ترین framework ها در زمینه ی صنعتی، framework Arduino است. بی‌شک هر استارت‌آپ نوپایی باید ابتدا کدهای خود را تحت این زبان بنویسد چراکه یکی از غنی‌ترین community (انجمن) ها را دارد و در زمینه کار با یک میکرو کنترلر، یکی از مهم‌ترین عوامل داشتن انجمن قوی است که آردینو از این ویژگی برخوردار است.

استارت‌آپ وینگ نیز از این قاعده مستثنا نیست و امیررضا فتحی که بخش کد زنی سخت‌افزاری این استارت‌آپ را دارد، کدهای این قسمت را به کمک شایان شریفی تحت آردوینو نوشته است.

حال وظیفه من در این استارت‌آپ پیاده‌سازی این کاربر روی یک framework جدید است. یکی از دلایل این کار این است که میکرو کنترلر مورد استفاده در این استارت‌آپ، میکرو کنترلر ESP32 است که می‌تواند علاوه بر framework اختصاصی خود یعنی espressif، تحت framework معروف آردوینو نیز برنامه‌نویسی شود، حال این استارت‌آپ دنبال استفاده از framework اختصاصی این میکرو کنترلر به دلیل ثبات بالا (stability) در هنگام اجرا است.

در انجام کار با IDE جدیدی برای کد زنی آشنا شدم که به معرفی آن نیز می‌پردازم.

قابلیت‌های خاص این میکرو کنترلر که در ادامه به توضیح آن می‌پردازیم باعث شده است تا بتوانیم آن را از راه دور پروگرام کنیم و این ویژگی بسیار ارزشمندی است و دلیل آن نیز این است که مثلاً اگر مشکلی در کد پیدا شود دیگر نیاز به جمع‌آوری همه اسکوترها نیست و می‌توان از راه دور همه اسکوترها را مجدد پروگرام کرد.

## ۲-۲ معرفی میکروکنترلر ESP32 [4]

چیپ ESP32 شامل یک پردازنده دو هسته‌ای ۳۲ بیتی است که می‌تواند با فرکانس ۱۶۰ تا ۲۴۰ مگاهرتز کار کند و البته این پردازنده بسیار کم‌مصرف یا به اصطلاح (ultra low power) ULP است.

این میکرو ۴۸ پایه دارد و حافظه داخلی کوچکی دارد به همین دلیل در داخل ماژول‌های طراحی‌شده به همراه این چیپ یک حافظه فلش قرار دادند تا بتوان کدهای سنگین‌تری را نیز پشتیبانی کند. مقدار این حافظه فلش برای ماژولی که ما از آن استفاده می‌کنیم 4 MB است. برای مقایسه می‌توان میکروکنترلر ATmega128 را در نظر گرفت که صرفاً 128 KB حافظه برای کد زنی دارد.

این میکروکنترلر همچنین دارای یک ماژول وای فای و بلوتوث داخل خود است که دارای مشخصات زیر هستند:

Wi-Fi: 802.11 b/g/n

Bluetooth: v4.2 BR/EDR and BLE

قابلیت‌های داشتن حافظه فلش بالا و وای فای به ما کمک می‌کند تا بتوانیم این میکروکنترلر را از راه دور پروگرام کنیم و همچنین راه ارتباطی خوبی برای فرستادن پیام به این میکروکنترلر است.

گذرگاه‌های ارتباطی را در عکس زیر می‌بینید:

- Peripheral interfaces:
  - + 12-bit SAR ADC up to 18 channels
  - + 2 × 8-bit DACs
  - + 10 × touch sensors
  - + Temperature sensor
  - + 4 × SPI
  - + 2 × I<sup>2</sup>S
  - + 2 × I<sup>2</sup>C
  - + 3 × UART
  - + SD/SDIO/MMC host
  - + Slave (SDIO/SPI)
  - + Ethernet MAC interface with dedicated DMA and IEEE 1588 support
  - + CAN bus 2.0
  - + IR (TX/RX)
  - + Motor PWM
  - + LED PWM up to 16 channels
  - + Hall effect sensor
  - + Ultra low power analog pre-amplifier

۱-۲ گذرگاه‌های ارتباطی [5]

عکسی از میکروکنترلر:



۲-۲ میکروکنترلر ESP32

در این کارآموزی برای استفاده از پین های این میکروکنترلر از برد آموزشی زیر استفاده شده است:



۳-۲ برد آموزشی

این میکروکنترلر را می توان با راه های زیر به صورت سریال پروگرام کرد:

(۱) دستورات پایتون در محیط Command Prompt مانند زیر:

```
Esptool.py erase_flash
```

\*همه راه ها نیازمند نصب esptool هستند.

(۲) استفاده از flash\_downloader که توسط خود کمپانی سازنده میکرو ساخته شده است.

(۳) استفاده از ide هایی که از این framework پشتیبانی می کنند مانند Eclipse و Platform Io.

## ۳-۲ توضیح در مورد Platform Io IDE

دلیل استفاده از این ide به دلیل داشتن community فوق‌العاده قوی و بروز است. طبق تجربه خودم از کاربر روی میکروکنترلر های stm32 و ATmega داشتن یک community قوی یکی از شروط اصلی انتخاب کردن محیطی است که قرار است در آن کد بزنیم.

این IDE به صورت یک افزونه بر روی برنامه Visual Studio Code و یا Atom اجرا می‌شود.

سازنده این افزونه یک گروه روسی هستند و به نظر رقیبی قدرتمند برای Arduino IDE است.

بخش این افزونه را جایگزین محیط Arduino ide می‌کنم.

نحوه نصب این افزونه نیز بسیار ساده است، ابتدا باید VsCode را نصب کرد، سپس در بخش افزونه‌های آن،

کلمه Platform Io را باید سرچ کرد و بر روی دکمه نصب کلیک کرد.




در زیر عکسی از نماد این افزونه و مشاهده می‌کنید:



۴-۲ نماد Platform Io

در زیر تصویری از مقایسه سرعت build شدن پروژه‌ها روی یک میکروکنترلر از خانواده esp آورده شده است:

## Comparison of ESP8266 Project Build Times

Time to Build Project by IDE	 Arduino IDE 1.6.8	 PlatformIO IDE 1.1.1	 Arduino Eclipse Plugin V3
Entire Project	24s	15s	28s
Single File Changed	8s	3s	3s

**Tested Sketch:** WiFiWebServer.ino from <https://github.com/esp8266/Arduino> , core 2.1.0  
**Test PC:** Dell OptiPlex 780, Intel Core 2 Duo E8400 @3GHz, 12GB RAM, SSD, Windows 7 x64

۵-۲ سرعت build شدن بر روی ide های مختلف [6]

قابل مشاهده است که سرعت در Platform io بسیار بالاست و این قابل تحسین است.

این افزونه اکنون تقریباً تمامی framework های مورد استفاده در میکروکنترلرها را پشتیبانی می کند و هرروزه شاهد بهروزرسانی برای این افزونه هستیم.



## ۲-۴ ESP Iot Development Framework

همان‌طور که در مقدمه گفته شد، از framework، Espressif برای کد زنی روی ESP32 استفاده شده است. انتخاب کردن این framework در ابتدا صرفاً یک چالش بود و انتظار از من این‌طور بود که در انتها بتوانم مزیت و یا عیب این framework را پیدا کنم و طبق مشاهدات من این framework در مقابل Arduino چیزی کم ندارد و حتی documentation غنی‌تر و کاربردی‌تری دارد. بعید است که بخواهید یک عملی را با EPS32 انجام دهید ولی مثالی از آن را روی اینترنت پیدا نکنید و این مزیت بسیار بزرگی است. البته همه این مثال‌ها نیاز به تغییر دارند زیرا این framework برای Platform Io با اندکی تغییر عرضه شده است و باید کدها را اندکی تغییر داد.



۲-۶ نماد Espressif

## ۵-۲ Over-the-air programming

OTA به این معناست که شما می‌توانید دستگاهتان را از راه دور پروگرام کنید و این قابلیت بزرگی است اگر تعداد دستگاه‌ها بالا باشند و هرکدام در نقطه‌ای از شهر باشند، این مزیت بیش‌ازپیش به چشم می‌خورد.

البته این قابلیت زمانی قابل‌استفاده خواهد بود که میکروکنترلر همراه با یک ماژول wifi عرضه شود که این در مورد ESP32 صدق می‌کند و نیز ویژگی مهم‌تری که داشتن حافظه فلش بالاست، زیرا زمانی که دیتای برنامه جدید برای میکروکنترلر ارسال می‌شود باید علاوه برداشتن حافظه برای برنامه قبلی، برنامه جدید را نیز بتواند بر روی فلش ذخیره کند و در انتهای بارگذاری، میکرو را با برنامه جدید لود کند. ما این کار را برای دو حالت انجام می‌دهیم، زمانی که میکروکنترلر به یک Access point متصل است و خودش دنبال دیتای جدید می‌گردد و زمانی که یک station به Access point خود ESP32 متصل است و این دو حالت را باهم مقایسه می‌کنیم و مزیت‌ها و معایب آن‌ها را بیان می‌کنیم.

برای دسترسی تمیزتر به حافظه فلش، قابلیت partition بندی برای این میکروکنترلر ها تعریف شده است. برخلاف سایر میکروکنترلر ها که شاید این کار بسیار سخت به نظر برسد در این framework کار بسیار راحتی است.

حالا با دو مد از این میکروکنترلر آشنا می‌شویم:

### مد اول: حالت single app

اکثر برنامه‌هایی که بر روی این میکروکنترلر پیاده‌سازی می‌شوند و نیاز به ota ندارند، باید در این مد partition بندی شوند.

```
# Espressif ESP32 Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs, data, nvs, 0x9000, 0x6000,
phy_init, data, phy, 0xf000, 0x1000,
factory, app, factory, 0x10000, 1M,
```

۷-۲ single mode partition[8]

همان‌طور که در بالا مشاهده می‌کنید 1 MB حافظه برای بارگذاری کد در بخش factory در نظر گرفته شده است.

این partition به صورت یک فایل csv تشکیل می‌شود و به صورت دستی می‌توان آن را modify کرد.

معمولاً بر اساس برنامه و پروژه ای که قرار است بر روی آن انجام شود، این partition تغییر می‌کند.

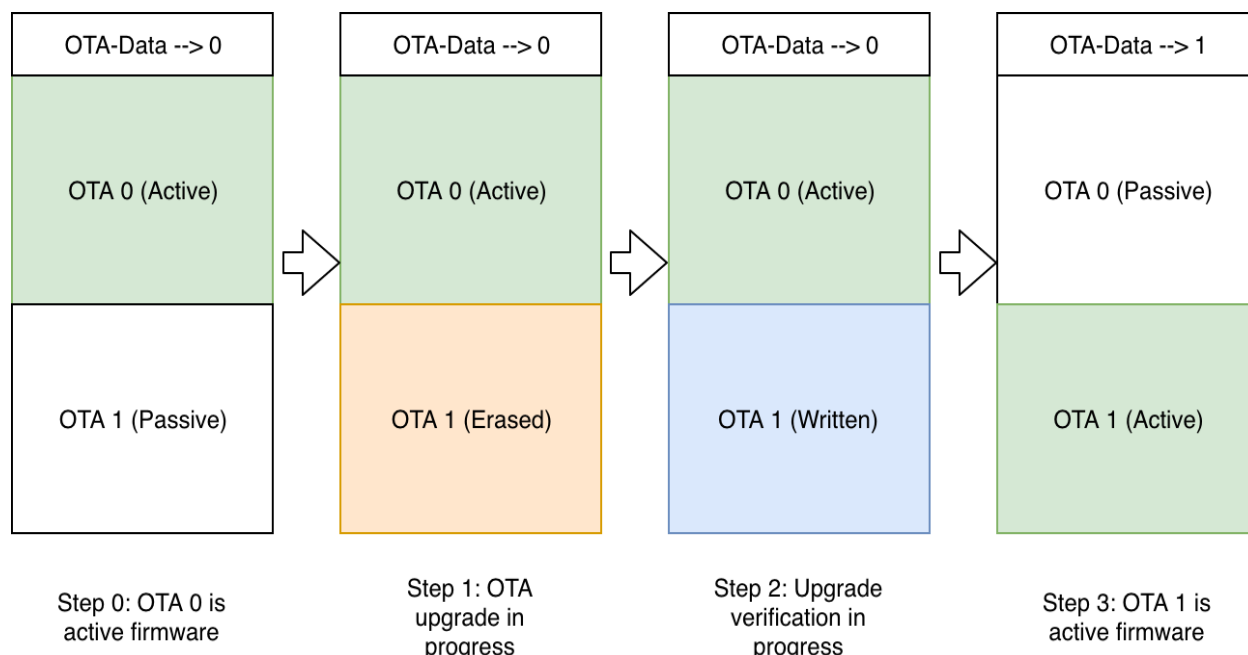
## مد دوم: حالت ota یا two apps

اگر بخواهیم قابلیت ota را در این میکروکنترلر استفاده کنیم باید partition بندی به صورت زیر باشد.

```
# Espressif ESP32 Partition Table
# Name,   Type, SubType, Offset, Size, Flags
nvs,      data, nvs,     0x9000, 0x4000,
otadata,  data, ota,        0xd000, 0x2000,
phy_init, data, phy,     0xf000, 0x1000,
factory,  0,    0,        0x10000, 1M,
ota_0,    0,    ota_0,    0x110000, 1M,
ota_1,    0,    ota_1,    0x210000, 1M,
```

ota mode partition[8] ۸-۲

همان طور که در تصویر بالا دیده می شود مانند حالت single app یک بخش factory برای کد اولیه در نظر گرفته شده است. حال دو بخش جدید به نام های ota\_0 و ota\_1 می بینیم، این دو بخش از این قرار هستند که زمانی که برنامه از روی factory در حال اجرا شدن باشد، دیتای جدید بارگذاری شده بر روی ota\_0 نوشته می شود و در انتهای دانلود شدن دیتا، برنامه از روی این بخش اجرا می شود، در ادامه با دریافت دیتای جدید، دیتای بر روی ota\_1 ذخیره می شود و بعد دانلود شدن کامل، دستگاه با دیتای دانلود شده روی این بخش اجرا می شود و از این به بعد هر دیتای جدیدی که به میکروکنترلر ارسال شود، میکرو بین دو بخش ota\_0 و ota\_1 تغییر وضعیت می دهد.



۹-۲ نحوه انجام ota [9]

حال به بررسی دوحالتی که برای پیاده‌سازی ota در نظر داریم می‌پردازیم:

### حالت اول) اتصال Esp32 به یک Access point

همراه این میکروکنترلر درون اسکوترها یک مودم LTE قرار گرفته است که به شبکه اینترنت متصل است و ما به کمک این مودم می‌توانیم دیتای جدید را به میکروکنترلر ارسال کنیم.

روش ما به این صورت است که از یک پروتکل انتقال پیام MQTT که در بخش بعد بیشتر به آن می‌پردازیم، استفاده می‌کنیم، بدین شکل که با ارسال پیام "CHECK FOR UPDATE" این میکروکنترلر یک فایل JSON را که حاوی دو دیتا است را بارگذاری می‌کند.

دیتای اول عدد نسخه firmware را نشان می‌دهد که آخرین باربر روی سرور بارگذاری شده است.

حال داخل کد میکرو نیز یک عدد تحت عنوان firmware ذخیره شده است، میکرو با مقایسه عدد نسخه خود با عدد موجود در فایل json تصمیم به آپدیت می گیرد و قاعده‌تاً زمانی این عملیات را انجام می دهد که نسخه روی سرور از نسخه خودش بروزتر باشد.

دیتای دوم فایل JSON، لینک دیتای دوم است که در صورت تشخیص آپدیت جدید، دیتا را از این لینک دانلود می کند.

این روش به این دلیل مناسب است که نیاز به چک کردن مداوم سرور توسط میکروکنترلر نیست و صرفاً با دریافت یک پیام این کار را انجام می دهد.

### حالت دوم) اتصال به Access point ایجادشده توسط Esp32

در این حالت بر روی Esp32 یک وب سرور بالا می آوریم و یک صفحه html لود می کنیم.

خودم برای این کار در file، directory، یک صفحه html ایجاد کردم و در آن یک ورودی آپلود فایل قراردادام و در ادامه می توان با هر دستگاهی به میکروکنترلر وصل شد و دیتای جدید را برای آن ارسال کرد.

البته مودم تولیدشده توسط esp32 دارای پسورد است و همه نمی توانند به آن دسترسی داشته باشند.

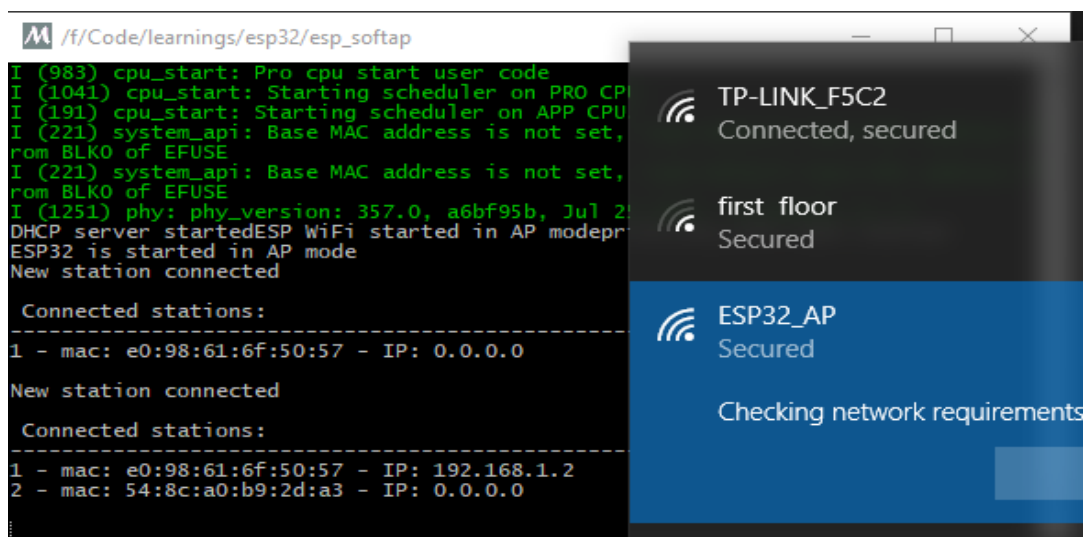
مقایسه این دو حالت:

در حالت اول سرعت دانلود دیتا به دلیل استفاده از مودم LTE بالاتر است و نیز زمانی که بخواهیم دستگاه‌های زیادی را به صورت هم‌زمان آپدیت کنیم، این روش بسیار کارآمد است.

در مقابل حالت دوم کندتر از حالت اول است، به این صورت که با انجام آزمایشی زمان بارگذاری فایل‌ها در دو حالت مختلف اندازه‌گیری شد و در حالت این زمان حدوداً ۳۰ ثانیه است و در حالت دوم حدود ۱ دقیقه است و این تفاوت بسیار چشم‌گیر است.

همچنین زمانی که بخواهیم همه دستگاه‌ها را باهم آپدیت کنیم، این روش اصلاً عملی نخواهد بود.

ولی اگر بخواهیم صرفاً یک دستگاه را آپدیت کنیم از این روش می‌توان استفاده کرد، من صرفاً به دلیل آموزشی هر دو روش را بر روی این میکرو پیاده کردم.



```
/f/Code/learnings/esp32/esp_softap
I (983) cpu_start: Pro cpu start user code
I (1041) cpu_start: Starting scheduler on PRO CPU
I (191) cpu_start: Starting scheduler on APP CPU
I (221) system_api: Base MAC address is not set,
rom BLK0 of EFUSE
I (221) system_api: Base MAC address is not set,
rom BLK0 of EFUSE
I (1251) phy: phy_version: 357.0, a6bf95b, Jul 2
DHCP server startedESP WiFi started in AP modepr
ESP32 is started in AP mode
New station connected

Connected stations:
-----
1 - mac: e0:98:61:6f:50:57 - IP: 0.0.0.0

New station connected

Connected stations:
-----
1 - mac: e0:98:61:6f:50:57 - IP: 192.168.1.2
2 - mac: 54:8c:a0:b9:2d:a3 - IP: 0.0.0.0
```

۲-۱۰ اتصال به ESP32

## ۲-۶ [10] Message Queuing Telemetry Transport

پروتکل MQTT یک پروتکل ساده است که بر روی بستر TCP/IP سوار شده است و انتقال اطلاعات را بر اساس Socket انجام می‌دهد، هدرهای مورد استفاده در MQTT حجم خیلی کمی دارند و این موضوع باعث کاهش حجم ترافیک در تبادلات شده و همچنین باعث سادگی پروتکل می‌شود،

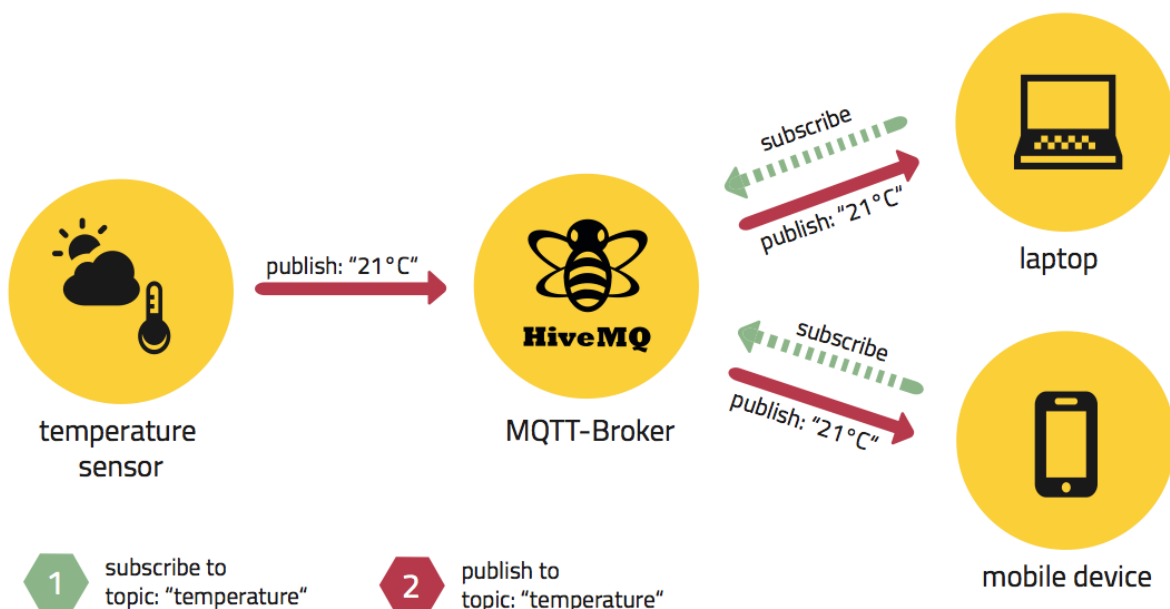
با مثالی این پروتکل را واضح‌تر می‌کنیم، فرض کنید عضو یک کانال تلگرام هستید، در این صورت مدیر کانال صرفاً یک‌بار پیام را درون کانال می‌فرستد و تمامی اعضای آن کانال، آن پیام را دریافت می‌کنند.

در این پروتکل، مفهومی بنام Broker را داریم، Broker به این صورت عمل می‌کند که رابطی است بین فرستنده پیام و گیرنده پیام، در مثال قبل کانال تلگرام نقش یک broker را بازی می‌کرد.

مدیر کانال نقش یک Publisher را به عهده دارد و شمایی که عضو کانال هستید نقش یک Subscriber را دارید. انتقال پیام در این پروتکل به این صورت انجام می‌شود که یک topic توافق می‌شود بین فرستنده و گیرنده، یعنی subscriber یک topic مشخص را subscribe می‌کند (به حالت listener)، حالت اگر یک فردی پیامی را در همان topic به همان broker، publish کند، broker پیام را به subscribe کننده ارسال می‌کند.

اگرچند دستگاه به‌طور هم‌زمان یک topic را subscribe کنند و پیامی در آن topic، publish شود، broker آن پیام را برای تمامی subscriber ها ارسال می‌کند.





۱۱-۲ مثالی از ارسال پیام [11]

مفهوم دیگری که در این پروتکل استفاده می‌شود، مفهومی بنام Quality of Service یا به اختصار QoS است، این مفهوم کمک می‌کند که بتوانیم نحوه ارسال پیام‌ها را مدیریت کنیم.

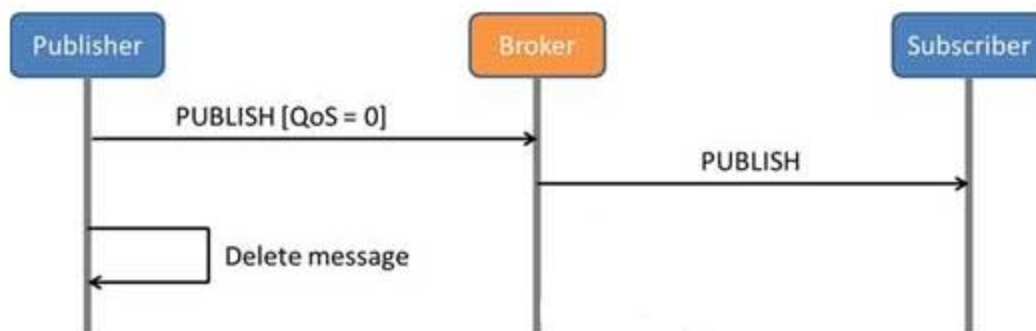
برای مثال فرض کنید بخواهید با سنسور دما، دمای یک مکانی را گزارش کنید و ممکن است دمای آن محیط زیاد مهم نباشد و به دلیل قطع شدن ارتباط از دست بروند و بتوان از آن چشم‌پوشی کرد.

حال فرض کنید بخواهید دمای کوره‌ای را گزارش کنید، در این حالت تمام داده‌ها از اهمیت خیلی بالایی برخوردار هستند و از دست رفتن داده می‌تواند خطرات زیادی به همراه داشته باشد. پس از QoS دیگری در ارسال داده‌ها استفاده می‌شود که احتمال از بین رفتن و یا گم‌شدن اطلاعات در آن‌ها خیلی پایین است.

در زیر سه حالت مختلف QoS را بررسی می‌کنیم:

#### الف) QoS Level 0

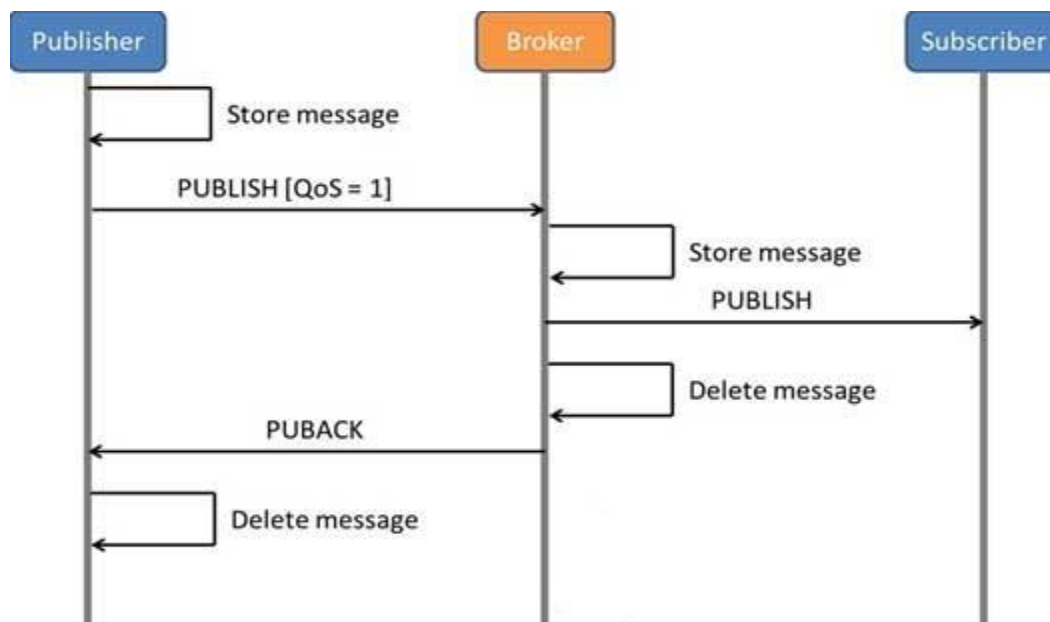
QoS Level 0 که به اختصار QoS0 هم نوشته می‌شود، ساده‌ترین حالت برقراری ارتباط با broker است که نیازی به acknowledgment ندارد، بدین معنی که به این موضوع که پیام شما توسط Broker دریافت شود یا نشود اهمیتی نمی‌دهد. همین که Ack شبکه TCP را دریافت کند ملاک را بر ارسال پیام می‌گذارد و پیام را از لیست ارسال حذف می‌کند. در این روش اگر فرض کنیم کانکشن قطع شده باشد و هنوز اینترپت آن به دستگاه نرسیده باشد، ممکن است بخشی از داده‌های ارسالی هیچ‌وقت به broker نرسند. در ارتباطات ساده این روش قابل قبول است چرا که حجم مصرفی داده کمی دارد. من نیز برای شروع از این حالت در پروژه خود استفاده کردم.



۱۲-۲ QoS0

## QoS Level 1 (ب)

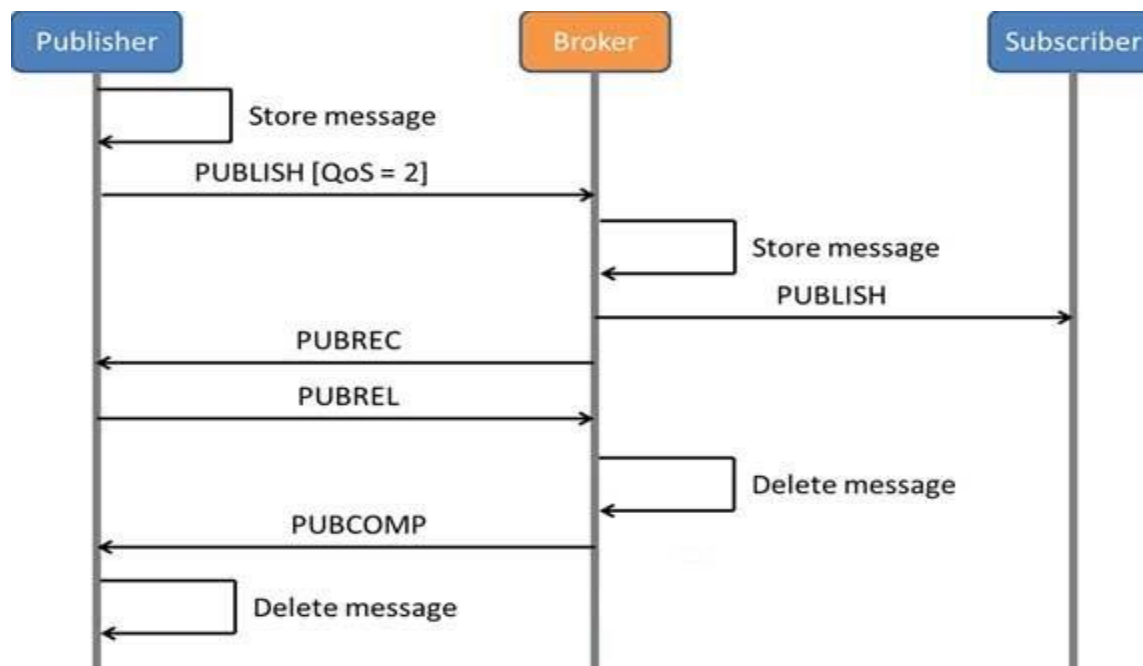
در این حالت دستگاه اطمینان حاصل می‌کند که حداقل یک پیام توسط Broker دریافت شده است. به این صورت که Broker پس از دریافت پیام، یک پیام PUBACK به دستگاه ارسال می‌کند و دستگاه با دریافت PUBACK می‌فهمد پیام به Broker رسیده است و پیام را از صف ارسال حذف می‌کند. به هر دلیلی که پیام PUBACK دریافت نشود، برنامه مجدداً بسته را ارسال خواهد کرد و این کار تا زمانی که پیام توسط Broker دریافت شود ادامه خواهد یافت.



QoS1 ۱۲-۲

## پ) QoS Level 2

در این حالت دستگاه ارسال کننده اطمینان حاصل خواهد کرد که پیام تنها یکبار توسط یک subscribe کننده دریافت شده است نه بیشتر. به این طریق در محل های حساس می شود فهمید که دستگاه پردازنده اطلاعات، داده های ارسالی را دریافت کرده است. در این روش ترافیک زیادتری بین دستگاه و برآکر ردوبدل می شود تا این اطمینان ایجاد شود که داده های ارسالی فقط یکبار توسط عضوها دریافت می شود و نه تعداد بیشتری.



QoS2 ۱۴-۲

همان‌طور که در بالا توضیح دادم broker ها پل‌های ارتباطی بین subscriber ها و publisher ها هستند. دسترسی به این Broker ها از طریق URI (URL + PORT) آن‌ها هستند.

Broker های آنلاین و رایگان زیادی وجود دارند که چند نمونه از آن‌ها را نام می‌برم:

## HiveMQ



Hive Broker ۱۵-۲



Eclipse Broker ۱۶-۲



Mosquitto Broker ۱۷-۲

مفهوم دیگری که در MQTT مطرح است، مفهومی تحت عنوان Retain است. اگر پیامی که Publish می‌شود دارای این flag باشد، broker همواره اطمینان حاصل پیدا می‌کند که آخرین پیامی که publish شده است توسط هر subscriber یکبار دریافت خواهد شد.

در واقع این پیام هر بار یک subscriber اضافه شود به آن ارسال می‌شود.

در این پروژه سعی شد که MQTT تحت چهار حالت مختلف زیر پیاده شود:

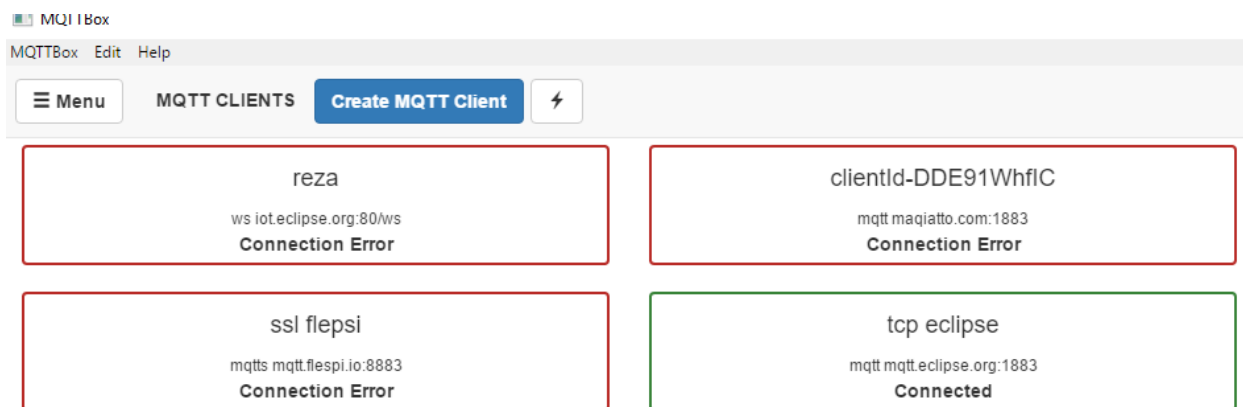
Web Socket (۱)

Web Socket Secure (۲)

TCP (۳)

SSL (۴)

در پروژه‌های بالا اکثراً از Broker رایگان Eclipse استفاده شد و همچنین نرم‌افزار رایگان MQTTBOX



۱۸-۲ محیط برنامه MQTTBOX

همچنین به صورت packet دیتای MQTT تحت TCP Connection با میکروکنترلر stm32 ارسال شد.

همچنین می توان با برنامه packet sender ارسال یک دیتای MQTT را مشاهده کرد.

همچنین می توان با نصب کردن نرم افزار Mosquitto بر روی LocalHost یک Broker راه اندازی کرد و پیام ارسال و دریافت کرد.

همچنین برای استفاده از پروتکل های wss و ssl مجبور به استفاده از certificate اختصاصی این broker

ها هستیم که به منظور دستیابی به آن ها می توان از دستورات openssl در Command Prompt کمک

گرفت و خروجی را تحت عنوان pem ذخیره و از آن استفاده کرد.

```
C:\Users\Rezaneo7>openssl s_client -host iot.eclipse.org -port 443 -showcerts
CONNECTED(000001D0)
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
verify error:num=20:unable to get local issuer certificate
---
Certificate chain
 0 s:/C=CA/ST=Ontario/L=Ottawa/O=Eclipse.org Foundation, Inc./OU=IT/CN=*.eclipse.org
 1 i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
-----BEGIN CERTIFICATE-----
MIIHZjCCBk6gAwIBAgIQBXStleLgQwEVQJT+V/d0SzANBgkqhkiG9w0BAQsFADBw
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQSW5jMRkwFwYDVQQLExB3
d3cuZGlnYWVlcnQwY29tMS8wLQYDVQQDEyZEaWdpQ2VydCBTSEYyIEhpZ2ggQXNz
dXJhbml1IFNlcnZlciBDQTAeFw0xNzAxMTcwMDAwMDBaFw0yMDAzMDIxMjAwMDBa
MHwxGzAgAjBgNVBAYTAkNBMR4wEgYDVQQtIEwDQ2VydW5kYXRpb24sIEluYy4xCzAJBgNV
d2ExJTAjBgNVBAoTHEVjbG1wc2Uub3JnIEZvdW5kYXRpb24sIEluYy4xCzAJBgNV
BAsTAKlUMRYwFAYDVQQDDA0qLmVjbG1wc2Uub3JnMIIIBjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAtiSzPmMwLHagBFfAIhE0eoeconPwKS33vmsYP1NOCvU1
Qq/Kp9I/srR/b0Ive+moXWkvXsn+rXcnfAkmKKjaczjiam86KAvrSjKprSrBAJZ7
oKcl//Dh10GZFK85Q4BRgGAPaL8RtcnV74UMYiIBl0QEt+2hS/SLyvCKJgGbbmZp
h1FFM/TenB1udTewQid0870VXh4cnaNCuuyFggvov14Uhp73dfuPCV7043T00Y
```

۱۹-۲ یافتن certificate

## ۷-۲ پخش صدا

در این قسمت می‌خواهیم با ذخیره کردن فایل‌های صوتی در داخل حافظه فلش، صدایی را به‌عنوان اخطار پخش کنیم.

ابتدا برای این کار نیاز است **partition** بندی را عوض کنیم و یک حافظه برای صدا در نظر بگیریم.

	A	B	C	D	E	F	G
1	# Name	Type	SubType	Offset	Size	Flags	
2	# Note: if make sure to change the offset in Kconfig.projbuild						
3	nvs	data	nvs		0x4000		
4	otadata	data	ota		0x2000		
5	phy_init	data	phy		0x1000		
6	factory	app	factory		1M		
7	ota_0	app	ota_0		1M		
8	ota_1	app	ota_1		1M		
9	storage	data	fat		0x9000		

## ۲۰-۲ پارتیشن جدید حافظه فلش

همان‌طور که در بالا نیز قابل‌مشاهده است بعد از بخش‌های ota، یک بخش 900 KB به نام storage برای ذخیره کردن صدا ایجاد کرده‌ایم. حال صرفاً باید صدایی که در لپ‌تاپ ذخیره کردیم را به فرمت قابل پخش توسط ESP32 درآوریم.

از نرم‌افزار Audacity برای تبدیل نرخ فایل صوتی به 16000 HZ و تبدیل آن به حالت Mono از حالت Stereo استفاده می‌کنیم.

سپس به کمک یک کد پایتون این صدا را به آرایه قابل ذخیره برای ESP32 تبدیل می‌کنیم.



کلیه مراحل بالا را در تصویر زیر (ذخیره شده در اکانت github خودم) می بینید:

build passing

- The script in tools folder provides an example of generating audio tables from .wav files. just put the .wav file in tools folder and run script
- In this example, the wav file must be in 16k/16bit mono format. you can use audacity for converting.
- The script will bundle the wav files into a single table named audio\_example\_file.h. it will be created in tools folder.
- Since the ADC can only play 8-bit data, the script will scale each 16-bit value to a 8-bit value. in function "example\_i2s\_adc\_data\_scale"
- The script will covert all signed values into unsigned values because only positive values will be output by the ADC.

\*we can use this code to record sound from adc of esp32 and then plat it back by dac. just set these flags

"RECORD\_IN\_FLASH\_EN" && "REPLAY\_FROM\_FLASH\_EN"

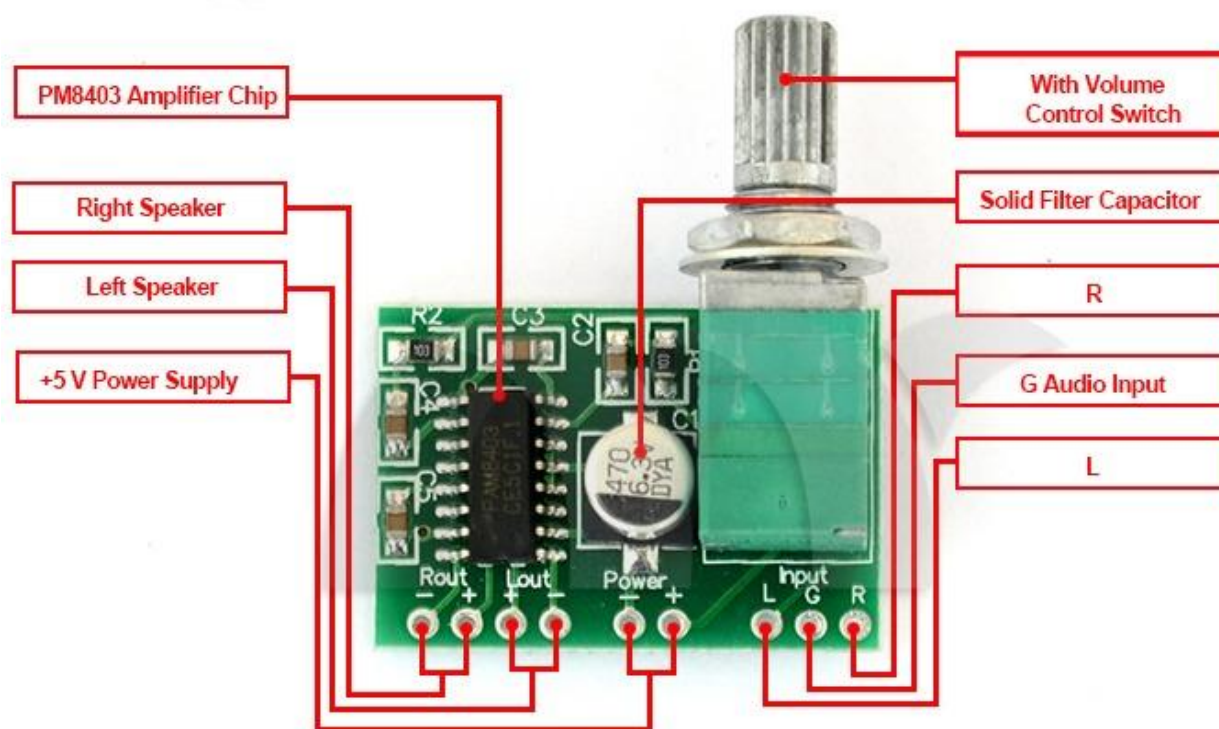
- first we should convert it to rate 16000, then convert it from stereo to mono

۲-۲۱ مراحل ویرایش فایل صوتی [12]

```
2  const unsigned char audio_table[] = {
3  0x7e, 0x80, 0x7e, 0x80, 0x80, 0x7e, 0x7e, 0x7e, 0x80, 0x7e, 0x7e, 0x7e, 0x80, 0x80, 0x80, 0x7e,
4  0x80, 0x80, 0x7e, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7e, 0x80, 0x80, 0x7e, 0x80, 0x80, 0x7e, 0x80,
5  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7e, 0x80, 0x80, 0x7d, 0x7d, 0x7e, 0x7e,
6  0x7e, 0x80, 0x7e, 0x7e, 0x80, 0x80, 0x81, 0x80, 0x80, 0x80, 0x7e, 0x7e, 0x7e, 0x7e, 0x7d,
7  0x7e, 0x7e, 0x80, 0x81, 0x80, 0x80, 0x80, 0x7e, 0x7e, 0x80, 0x80, 0x80, 0x7e, 0x80, 0x7e, 0x7e,
8  0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x7e, 0x80, 0x80, 0x80, 0x7e, 0x7d, 0x7e, 0x7e, 0x7e, 0x7d,
9  0x7e, 0x7d, 0x7e, 0x7d, 0x7e, 0x7d, 0x7e, 0x7e, 0x80, 0x7e, 0x7e, 0x7e, 0x7d, 0x80, 0x7c, 0x7e,
```

۲-۲۲ تصویری از فایل صدای ذخیره شده

حال این فایل صدا توسط پین‌های digital to analog converter به امپلی فایر pam8403 ارسال می‌شود و سپس توسط یک بلندگو پخش می‌شود.



۲-۲۳ عکس ماژول امپلی فایر [13]

البته در این پروژه صرفاً یک خروجی این امپلی فایر استفاده شده است.

## ۸-۲ MPU9250 [14]

این ماژول تقریباً تمام نیازهای شمارا برای ساخت یک سامانه اندازه‌گیری داخلی یا همان IMU با ۹ درجه آزادی و ۹ محور فراهم می‌نماید. دقت بسیار بالا و سیستم پیشرفته پردازش حرکت دیجیتال داخلی (DMP) از ویژگی‌های منحصربه‌فرد این ماژول می‌باشد.

در این پروژه صرفاً از ۳ محور شتاب سنج خطی (Accelerator) استفاده شده است. برای راه‌اندازی ماژول از پروتکل I2C استفاده شده است.

ویژگی‌های دیگر این ماژول را در زیر می‌بینید:

- سنسور جایرو دارای ۳ محور با دقت تا 131 LSBs/dps و رنج کامل از  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$  و  $\pm 2000$  dps
- شتاب سنج دارای ۳ محور با رنج کامل قابل برنامه ریزی  $\pm 8g$ ,  $\pm 4g$ ,  $\pm 2g$  و  $\pm 16g$
- قطب نمای ۳ محوره با رنج کاملی از  $\pm 1200\mu T$
- بدون اختلال در عملکرد همزمان جایرو، شتاب سنج و قطب نما
- جریان کاری جایرو: 3.6mA
- جریان جایرو و شتاب سنج: 3.8mA (توان کامل، جایرو در همه ی رنج ها و شتاب نچ در فرکانس نمونه برداری 1KHZ خواهد بود.)
- جریان کشی استفاده همزمان از چهار سنسور جایرو، شتاب، قطب نما و دما: 4.25mA (توان کامل، جایرو در همه ی رنج ها، شتاب سنج در فرکانس نمونه برداری 1kHz و قطب نما در فرکانس نمونه برداری 8Hz میباشد.)
- دارای رابط 400KHZ برای مد سریع ارتباط I2C
- دارای تایم ژنراتور بر روی تراشه با فرکانس  $\pm 1\%$
- دارای مقاومت پول آپ I2C بر روی برد
- پین های دارای فاصله ی 0.1 اینچ
- ابعاد: 25mm در 15mm
- وزن: 3 گرم

معمولاً خروجی‌های این مازول دارای خطای زیادی است، به این منظور از چند نوع فیلتر برای رفع اررور های این مازول استفاده می‌کنند که در بین این فیلترها سه فیلتر زیر معروف ترینند:

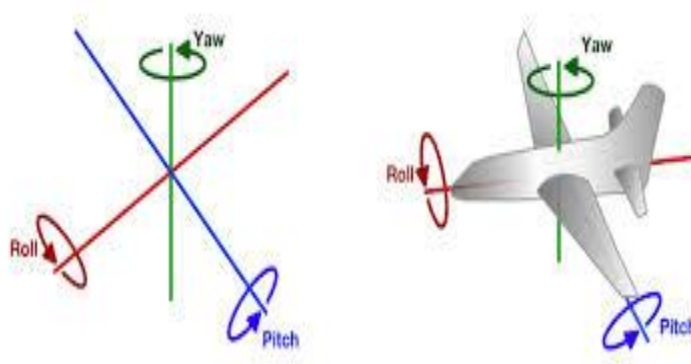
Kalman (۱)

Madgwick (۲)

Mahony (۳)

از فیلترهای بالا برای ترکیب ۹ دیتایی که از محورهای مختلف داده می‌شود، استفاده کرد و به اصطلاح روی دیتاها عمل fusion را انجام داد و میزان چرخش در مقابل سه محور را به دست آورد.

(به دست آوردن pitch,yaw,roll)



۲۵-۲ نشان دادن محور چرخش‌ها [15]

با مقایسه‌های انجام‌شده بین این سه فیلتر، Mahony از همه سریع‌تر است در محاسبات ولی دقت Madgwick از سایرین بالاتر است، به همین دلیل از فیلتر Madgwick استفاده می‌کنیم.

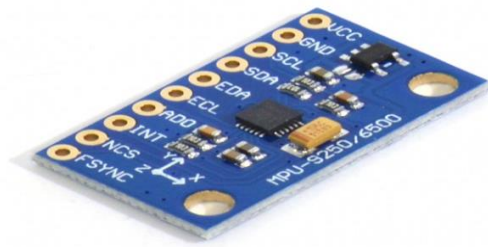
هدف استفاده از این ماژول، پی بردن به جابجایی‌هایی ناگهانی و جلوگیری از دزدی دستگاه است.

می‌خواهیم به‌وسیله این ماژول میزان جابجایی دستگاه را در دو جهت به دست بیاوریم.

حال چون ما صرفاً شتاب خطی را داریم، باید دو بار از آن انتگرال بگیریم که این به معنای ایجاد خطای بسیار بالاست. پس باید تقریب‌های دقیق‌تری بکار ببریم، از انتگرال ریمان استفاده می‌کنیم و طول بازه انتگرال‌گیری را کوچک می‌گیریم.

مراحل زیر را انجام می‌دهیم:

- (۱) ابتدا به‌منظور صفر کردن خطای حالت ایستا، از ۲۰۰ داده اول میانگین می‌گیریم و سپس داده‌های بعدی را از این میانگین (خطای حالت ایستا) کم می‌کنیم.
- (۲) حال عملیات خطی‌سازی در بازه‌های 100 ms را انجام می‌دهیم، به این منظور هر 10 ms از این بازه یک داده از mpu می‌گیریم و در انتها میانگین ۱۰ داده گرفته‌شده را به‌عنوان شتاب در آن بازه لحاظ می‌کنیم. تعداد سمپل‌هایی که mpu در هر ۱ ثانیه به ما می‌دهد برابر ۱۰۰۰ عدد است، پس در 100 ms، ما ۱۰۰ دیتا خواهیم داشت که از ۹۰ تای آن‌ها استفاده نمی‌کنیم.
- (۳) سپس شتاب‌ها را از یک فیلتر دیگر می‌گذرانیم، به این صورت که اگر شتاب پایین‌تر از یک مقدار مشخص بود، آن را صفر فرض می‌کنیم.
- (۴) سپس دو مرحله انتگرال ریمان را اعمال می‌کنیم.
- (۵) در صورتی‌که مجموع جابجایی در دو جهت از ۵ متر بیشتر بود، پیام خطاری را تحت MQTT ارسال می‌کنیم.



MPU9250 ۲۶-۲

از این دستگاه در خودروها می‌توان به‌منظور تحلیل رفتار رانندگان نیز استفاده کرد.

برای کار با این ماژول باید ابتدا رجیسترهایی را جهت تنظیم آن مقداردهی کرد. برای مثال برای تنظیم کردن Rate دیتا ارسالی، محدوده اندازه‌گیری شتاب سنج و ...

این رجیسترها در فایلی به نام register map در سایت سازنده این ماژول موجود هستند.

## ۹-۲ کارهای جانبی

### (۱) راه اندازی ماژول GPS neo 6m

از این ماژول به عنوان موقعیت یاب یک دستگاه در خارج از محیط های بسته استفاده می شود. خطای این ماژول در حدود ۱۰ متر است. دیتاهایی که از این ماژول خوانده می شود تحت زبان خاصی به نام NMEA هستند که از داخل می توان طول (longitude) و عرض (latitude) جغرافیایی را به دست آورد.



۲۷-۲ ماژول GPS NEO 6m

رابط کاربری این ماژول درگاه سریال (UART) است و Command های خاصی از قبیل PUBX و... می پذیرد.

مثالی از دیتای خروجی این ماژول:

**GGA** - essential fix data which provide 3D location and accuracy data.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Where:

GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality: 0 = invalid
	1 = GPS fix (SPS)
	2 = DGPS fix
	3 = PPS fix
	4 = Real Time Kinematic
	5 = Float RTK
	6 = estimated (dead reckoning) (2.3 feature)
	7 = Manual input mode
	8 = Simulation mode
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude, Meters, above mean sea level
46.9,M	Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field)	time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	the checksum data, always begins with *

۲-۲۸ یک مثال از خروجی ماژول GPS [16]

## ۲) تحقیق در مورد نحوه پیاده‌سازی WebGl بر روی ESP8266 [17]

به کمک این راهکار می‌توان عملیات پیچیده و یا حتی درزمینهٔ ماشین لرنینگ را از طریق وب بر

روی کارت گرافیک اجرا کرد و خروجی را با سرعت بالا بر روی میکرو کنترلر مشاهده کرد.

لینک کنفرانس مربوط به این بخش در بخش مراجع قرار داده شده است.



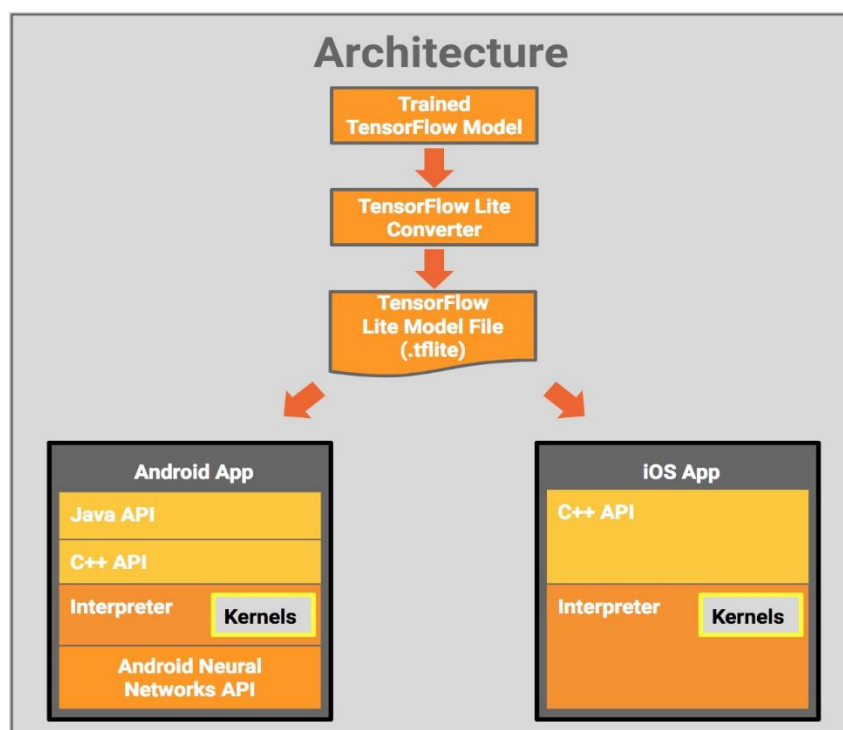
## [18] Tensorflow lite (۳)

این پکیج برای اجرای کدهای ماشین لرنینگ به صورت basic و کم حجم تر بر روی میکروکنترلر ها ساخته شده است که البته صرفاً دو دستگاه قابلیت پشتیبانی از این ویژگی را دارند.



۲-۲۹ نماد tensorflow lite

با بهینه سازی این پکیج حتی قادر به استفاده از آن بر روی پردازنده های تلفن همراه نیز هستیم:



۲-۳۰ استفاده از tensorflow بر روی گوشی

در این فصل کارهای انجام‌شده در طول مدت کارآموزی، اعم از کارهای اصلی برای پروژه کارهای جانبی که شخصاً به علت علاقه و کنجکاوی به سراغ آن‌ها رفتم، آورده شده است.

در هر بخش از پروژه اصلی تلاش شده که تمام حالت‌های موجود برای آن ویژگی تست شود و بهترین گزینه برای ادامه پروژه انتخاب شود.

فعالیت‌های جانبی نیز به صورتی انتخاب شد که در پروژه بکار بیایند. هر بخش پروژه اصلی بعد از چک شدن در github این‌جانب آپلود شده است و پروژه نهایی در دو فاز station و softAP بر روی Azure قرار گرفته است که لینک آن در بخش مراجع قرار خواهد گرفت.

## ۳ فصل سوم: جمع‌بندی، نتیجه‌گیری و پیشنهادها

### ۳-۱ جمع‌بندی

در فصل اول به معرفی کلی شرکت و استارت‌آپی که خودم در آن فعالیت داشتم و محصول این استارت‌آپ پرداختم و هدف خود را از حضور در این استارت‌آپ بیان نمودم. لازم است اینجا نیز ذکر کنم که کارگروهی و پایبندی به آن یکی از رازهای موفقیت این استارت‌آپ بوده است.

اگر تمام اعضای گروه پایبند به Integrity باشند، گروه به هیچ وجه شکست نخواهد خورد.

در فصل دوم به کارهای انجام‌شده توسط این جانب در شرکت پرداختم، معمولاً کارها به این صورت انجام می‌شد که قسمتی برای هر هفته مشخص می‌شد و در انتهای هفته جلسه‌ای با دیگر اعضای گروه تشکیل می‌شد و در آن جلسه من توضیحی در مورد فعالیت‌ها در آن هفته و نتایج آن بیان می‌کردم.

کارها اکثراً در راستای پروژه خود استارت‌آپ بود و من نیز جدا از پروژه‌ای که برای این استارت‌آپ انجام دادم، از تجربیات این گروه نیز بهره می‌بردم.

### ۳-۲ نتیجه‌گیری

در دوره کارآموزی متوجه این موضوع شدم که دنیای IOT، جای پیشرفت بسیار بزرگی دارد، بشخصه جدا از این پروژه یک پروژه دیگر نیز داشتم که مرا با این دنیا آشنا کرد. شاید در انتهای کار ادامه این مسیر را برای آینده خودم جالب نمی‌دانستم ولی تجربیاتی که در این راه به دست آوردم قطعاً در آینده به من کمک خواهد کرد.

جدا از پروژه، قرار گرفتن در یک محیط منظم کاری برای اولین بار، بسیار جالب و آموزنده بود.

محیط شرکت، محیط جدید و جذابی بود که بسیار من را به فکر ادامه فعالیت با این استارتاپ می‌انداخت و بالاینکه در انتها پیشنهاد برای کار در این استارتاپ به من داده شد، با توجه به علاقه به کاربر روی FPGA و دودل بودن بین این دو موضوع (یعنی iot و یا FPGA) این پیشنهاد را نپذیرفتم.

همچنین موارد آورده شده در این گزارش بسیار مهم هستند و در آینده یکی از منابع من برای دوره مطالب آموخته‌شده در این دوره کارآموزی خواهند بود.

### ۳-۳ پیشنهادها

طبق تجربه خودم، بهتر است محل کارآموزی یا به محل اقامت دانشجو نزدیک باشد و یا شرایط اسکان را داشته باشد.

بهتر است زمینه انتخاب‌شده برای دوره کارآموزی مرتبط با علایق دانشجو باشد تا در انتها بتواند نتیجه‌گیری کلی‌ای در مورد فعالیت‌های خود در آینده بکند.

## چند عکس از توضیحات پروژه نهایی:

### ◆ CompleteStation / README.md

ESP32 Platform Io Project

Project's features : OTA , MQTT , Audio , MPU 9250

WIFI MODE : Station

esp-idf documentation : <https://github.com/espressif/esp-idf/tree/master/examples> ↗

/\*\*\*\*\* OTA \*\*\*\*\*/

see this link : <http://www.lucadentella.it/en/2018/10/27/esp32-37-ota-via-https/> ↗

base code : <https://github.com/lucadentella/esp32-tutorial/> ↗

i changed some configs so it can be used in http mode ( without certificate )

/\*\*\*\*\* MQTT \*\*\*\*\*/

see this example : <https://github.com/espressif/esp-idf/tree/master/examples/protocols/mqtt/tcp> ↗

i just modified it for platform io.

you can set your broker by modifying esp\_mqtt\_client\_config\_t in **mqtt.c**.

and you can set your functions in its event handler (mqtt\_event\_handler\_cb)

/\*\*\*\*\* Audio \*\*\*\*\*/

The script in tools folder provides an example of generating audio tables from .wav files. just put the .wav file in tools folder and run script

In this example, the wav file must be in 16k/16bit mono format. you can use audacity for converting.

audacity: <https://www.foosshub.com/Audacity.html/audacity-win-2.3.1.exe> ↗

The script will bundle the wav files into a single table named audio\_example\_file.h. it will be created in tools folder.

Since the ADC can only play 8-bit data, the script will scale each 16-bit value to a 8-bit value. in function "example\_i2s\_adc\_data\_scale"

The script will covert all signed values into unsigned values because only positive values will be output by the ADC.

we can use this code to record sound from adc of esp32 and then plat it back by dac. just set these flags "RECORD\_IN\_FLASH\_EN" && "REPLAY\_FROM\_FLASH\_EN"

first we should convert it to rate 16000, then convert it from stereo to mono

**you should send "Play Audio" to "reza" topic. then you can hear sound.**

عکس مربوط به توضیحات نوشته شده در اکنت Azure Microsoft

/\*\*\*\*\* mpu 9250 \*\*\*\*\*/

i use madgwick filter for fusion of data.

The Madgwick filter is a glorified Complementary Filter with significant improvements to accuracy without significant markup in computation time.

if you wanna change the code, you should just modify `imu_task.c` file.

#### \*\*\* important variables \*\*\*

- `_imu` struct holds all the data we need. it will be updated by `imu_update` function.
- `accelXError` and `accelYError`, they hold the average of 200 first samples.
- `meanX`, `meanY`, they hold the linearized accelerometer value in 100 ms.
- `displacementX`, `displacementY`, they hold the displacement value in their own axis.
- `displacement`, it holds the overall displacement.
- `lastaccelX`, `lastaccelY`, they hold the linearized accelerometer value in last 100 ms.

you should send "Imu Task" to "reza" topic to initialize imu's task.

if the device moves more than 5 meter, the "displacement happend!!!" message will be published.

/\*\*\*\*\* extra notes \*\*\*\*\*/

for Secure mqtt connection see this link : <https://github.com/rezaneo7/WssMqtt>

you should convert your certificate to `.pem` file and save it into `src` folder

and you should paste line below into `platformio.ini`:

```
build_flags = -DCOMPONENT_EMBED_TXTFILES=src/ecli.pem
```

for erasing flash of esp32, after installing esptool, go to cmd and run command below :

```
esptool.py erase_flash
```

to change id and password of ap and station mode, in `wifi_functions.c` change `wifi_config_t` struct.

you can modify your own partition or you can use built in partitions, see this link :

<https://docs.platformio.org/en/latest/platforms/espressif32.html>

you can change the link of json file in `ota.c` change `esp_http_client_config_t`

you can change monitor\_speed in `platform.in`

you can change filter from madgwick to mahony by changing `USE_MADGWICK` in `imu.h`

## مراجع

- [1] <https://evand.com/organizations/%D8%B4%D8%AA%D8%A7%D8%A8-%D8%AF%D9%87%D9%86%D8%AF%D9%87-%D8%B3%D8%AE%D8%AA-%D8%A7%D9%81%D8%B2%D8%A7%D8%B1%DB%8C-%D9%87%D8%A7%D8%B1%D8%AF%D8%AA%DA%A9-53246093>
- [2] <https://www.linkedin.com/company/buynow/>
- [3] <https://icheezha.ir/%D8%A7%D8%B3%D8%AA%D8%A7%D8%B1%D8%AA%D8%A7%D9%BE-%D9%88%DB%8C%D9%86%DA%AF/>
- [4] <http://hiradbms.com/fa/news/20271/%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-%D9%BE%D8%A7%DB%8C%D9%87-%D9%87%D8%A7-%D9%88-%D8%A7%D9%85%DA%A9%D8%A7%D9%86%D8%A7%D8%AA-%D9%85%D8%A7%DA%98%D9%88%D9%84-ESP32>
- [5] <http://www.iotsharing.com/2017/05/introduction-to-esp32.html>
- [6] <https://twitter.com/krzychb2/status/719240133430636544/photo/1>
- [8] <https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/partition-tables.html>
- [9] <https://docs.espressif.com/projects/esp-jumpstart/en/latest/firmwareupgrade.html>
- [10] <https://sisoog.com/2018/09/%D9%BE%D8%B1%D9%88%D8%AA%DA%A9%D9%84-mqtt-%DA%86%DA%AF%D9%88%D9%86%D9%87-%DA%A9%D8%A7%D8%B1-%D9%85%DB%8C%E2%80%8C%DA%A9%D9%86%D8%AF%D8%9F/>
- [11] [https://www.eclipse.org/community/eclipse\\_newsletter/2014/october/article2.php](https://www.eclipse.org/community/eclipse_newsletter/2014/october/article2.php)
- [12] <https://github.com/rezaneo7/pam8403>
- [13] <http://electrobist.com/product/pam8403-mini-amplifier-5v-with-volume-6w/>
- [14] <http://thecaferobot.com/store/imu-ahrs-i2c-mpu9250>
- [15] <https://www.touringmachine.com/Articles/aircraft/6/>
- [16] <https://www.gpsinformation.org/dale/nmea.htm>

- [17] <https://www.youtube.com/watch?v=qkDg-Y9iHBA>
- [18] <https://www.youtube.com/watch?v=5J72iMSQmy8>
- [19] <https://dev.azure.com/ramazanpoorreza/CompleteStation>
- [20] [https://dev.azure.com/ramazanpoorreza/Complete\(SoftAp\\_Station\)](https://dev.azure.com/ramazanpoorreza/Complete(SoftAp_Station))